References

1. Abdelfattah, M.S., Bauer, L., Braun, C., Imhof, M.E., Kochte, M.A., Zhang, H., Henkel, J., Wunderlich, H.-J.: Transparent structural online test for reconfigurable systems. In: IEEE International On-Line Testing Symposium (IOLTS), pp. 37–42 (2012)
2. Amrouch, H., van Santen, V.M., Ebi, T., Wenzel, V., Henkel, J.: Towards interdependencies of aging mechanisms. In: IEEE/ACM International Conference on Computer-Aided Design (ICCAD), pp. 478–485 (2014)
3. Angermeier, J., Ziener, D., Glaß, M., Teich, J.: Stress-aware module placement on reconfigurable devices. In: International Conference on Field Programmable Logic and Applications (FPL), pp. 277–281 (2011)
4. Bauer, L., Shafique, M., Henkel, J.: Concepts, architectures, and run-time systems for efficient and adaptive reconfigurable processors. In: NASA/ESA Conference on Adaptive Hardware and Systems (AHS), pp. 80–87 (2011)
5. Bauer, L., Braun, C., Imhof, M.E., Kochte, M.A., Zhang, H., Wunderlich, H.-J., Henkel, J.: OTERA: Online test strategies for reliable reconfigurable architectures. In: NASA/ESA Conference on Adaptive Hardware and Systems (AHS), pp. 38–45 (2012)
6. Bauer, L., Braun, C., Imhof, M.E., Kochte, M.A., Schneider, E., Zhang, H., Henkel, J., Wunderlich, H.-J.: Test strategies for reliable runtime reconfigurable architectures. IEEE Trans. Comput. (TC) **62**(8), 1494–1507 (2013)
7. Bauer, L., Zhang, H., Kochte, M.A., Schneider, E., Wunderlich, H.-J., Henkel, J.: Advances in hardware reliability of reconfigurable many-core embedded systems. In: Many-Core Computing: Hardware and Software, pp. 395–416. Institution of Engineering and Technology (IET) (2019)
8. Cao, Y., Velamala, J., Sutaria, K., Chen, M.S.-W., Ahlbin, J., Esqueda, I.S., Bajura, M., Fritze, M.: Cross-layer modeling and simulation of circuit reliability. IEEE Trans. Comput. Aided Des. Integr. Circ. Syst. (TCAD) **33**(1), 8–23 (2014)
9. Gaisler, A.: Homepage of the Leon Processor. Online available: https://www.gaisler.com/index.php/products/processors/leon3. Accessed 13 Mar 2019
10. Guo, X., Burleson, W., Stan, M.: Modeling and experimental demonstration of accelerated self-healing techniques. In: IEEE/ACM Design Automation Conference (DAC), pp. 1–6 (2014)
11. Gupta, P., Agarwal, Y., Dolecek, L., Dutt, N., Gupta, R.K., Kumar, R., Mitra, S., Nicolau, A., Rosing, T.S., Srivastava, M.B., Swanson, S., Sylvester, D.: Underdesigned and opportunistic computing in presence of hardware variability. IEEE Trans. Comput. Aided Des. Integr. Circ. Syst. (TCAD) **32**(1), 8–23 (2013)
12. Henkel, J., Bauer, L., Becker, J., Bringmann, O., Brinkschulte, U., Chakraborty, S., Engel, M., Ernst, R., Härtig, H., Hedrich, L., Herkersdorf, A., Kapitza, R., Lohmann, D., Marwedel, P., Platzner, M., Rosenstiel, W., Schlichtmann, U., Spinczyk, O., Tahoori, M., Teich, J., Wehn, N., Wunderlich, H.-J.: Design and architectures for dependable embedded systems. In: International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS), pp. 69–78 (2011)
13. Henkel, J., Bauer, L., Dutt, N., Gupta, P., Nassif, S., Shafique, M., Tahoori, M., Wehn, N.: Reliable on-chip systems in the nano-era: lessons learnt and future trends. In: IEEE/ACM Design Automation Conference (DAC), pp. 1–10 (2013)
14. Huang, W., Ghosh, S., Velusamy, S., Sankaranarayanan, K., Skadron, K., Stan, M.R.: HotSpot: a compact thermal modeling methodology for early-stage VLSI design. IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **14**(5), 501–513 (2006)
15. Lala, P.K.: Self-checking and Fault-Tolerant Digital Design. Morgan Kaufmann, San Francisco (2001)
16. Mahapatra, S.: Fundamentals of Bias Temperature Instability in MOS Transistors: Characterization Methods, Process and Materials Impact, DC and AC Modeling. Springer Series in Advanced Microelectronics, vol. 52. Springer, New Delhi (2015)

17. Srinivasan, S., Krishnan, R., Mangalagiri, P., Xie, Y., Narayanan, V., Irwin, M.J., Sarpatwari, K.: Toward increasing FPGA lifetime. IEEE Trans. Depend. Sec. Comput. (TDSC) **5**(2), 115–127 (2008)
18. Stott, E.A., Wong, J.S., Sedcole, P., Cheung, P.Y.: Degradation in FPGAs: measurement and modelling. In: ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA), pp. 229–238 (2010)
19. Xilinx: Partial Reconfiguration User Guide, UG702 (v14.1) (2012)
20. Yang, S.: Logic synthesis and optimization benchmarks user guide: version 3.0. MCNC Technical Report, Microelectronics Center of North Carolina (MCNC). https://ddd.fit.cvut.cz/prj/Benchmarks/
21. Zhang, H., Bauer, L., Kochte, M.A., Schneider, E., Braun, C., Imhof, M.E., Wunderlich, H.-J., Henkel, J.: Module diversification: fault tolerance and aging mitigation for runtime reconfigurable architectures. In: IEEE International Test Conference (ITC), pp. 1–10 (2013)
22. Zhang, H., Kochte, M.A., Schneider, E., Bauer, L., Wunderlich, H.-J., Henkel, J.: STRAP: stress-aware placement for aging mitigation in runtime reconfigurable architectures. In: IEEE/ACM International Conference on Computer-Aided Design (ICCAD), pp. 38–45 (2015)
23. Zhang, H., Bauer, L., Kochte, M.A., Schneider, E., Wunderlich, H.J., Henkel, J.: Aging resilience and fault tolerance in runtime reconfigurable architectures. IEEE Trans. Comput. (TC) **66**(6), 957–970 (2017)

Reliability Analysis and Mitigation of Near-Threshold Voltage (NTC) Caches

Anteneh Gebregiorgis, Rajendra Bishnoi, and Mehdi B. Tahoori

1 Introduction

SRAM based memory elements have been the prominent limiting factor in the near-threshold voltage domain as the supply voltage of SRAM cells does not easily downscale, as it is done for combinational logic. The supply voltage downscaling limitation is due to the significant increase in the failure rate of SRAM cells operating at lower supply voltage values, which in turn severely affects the yield. Various state-of-the-art solutions have been proposed to address this issue. These solutions include variation tolerant SRAM cell design [3, 13, 29] and heterogeneous cache design [31], improve the robustness of cache memories. However, the improvement comes at the cost of increased area and power overheads. Moreover, these approaches mostly ignore the impact of runtime failure mechanisms, such as aging and soft error, on the reliability of memory components. Therefore, design-time reliability failure analysis and mitigation schemes are crucial for the reliable operation of near-threshold caches.

Analyzing failures based on a particular reliability failure mechanism is insufficient for estimating the system-level reliability, as the interdependence among different failure mechanisms has a considerable impact on the overall system reliability. Moreover, the running workload affects the aging and SER of memory components as it determines the SP and AVF of the memory elements. Therefore, performing a combined analysis on the reliability failure mechanisms across different layers of abstraction (as shown in Fig. 1) is crucial, and it helps designers to choose the most reliable components at each abstraction layer, and tackle the reliability challenges of NTC operation.

A. Gebregiorgis · R. Bishnoi · M. B. Tahoori (✉)
Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany
e-mail: anteneh.gebregiorgis@kit.edu; rajendra.bishnoi@kit.edu; mehdi.tahoori@kit.edu

J. Henkel, N. Dutt (eds.), *Dependable Embedded Systems*, Embedded Systems,
https://doi.org/10.1007/978-3-030-52017-5_13

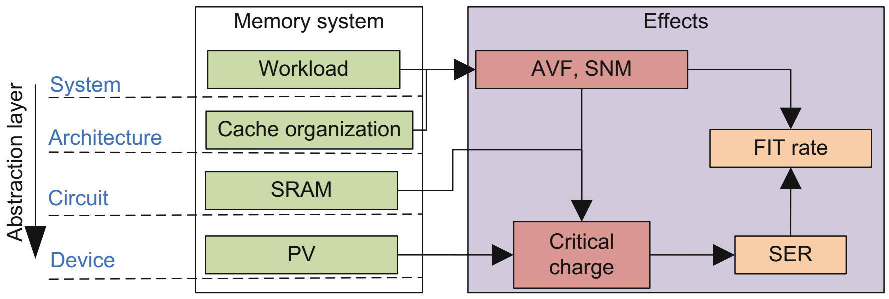


Fig. 1 Cross-layer impact of memory system and workload application on system-level reliability (Failure In-Time (FIT rate)) of NTC memory components, and their interdependence

For this purpose, a comprehensive cross-layer reliability analysis framework addressing the combined effect of aging, process variation, and soft error on the reliability of NTC cache designs is presented in this chapter. Moreover, the chapter presents the advantages and limitations of two different NTC SRAM cell designs (namely, 6T and 8T cells) in terms of reliability (SER and SNM) improvement, area, and energy overheads. The framework presented in this chapter helps to explore the cross-layer impact of different reliability failure mechanisms, and it is useful to study the combined effect of workload and cache organization on the SER and SNM of cache memories. The framework is also helpful to understand how the reliability issues change from super-threshold to the near-threshold voltage domain. Furthermore, it is important for architectural-level design space exploration to find the best cache organization for better reliability and performance trade-offs of NTC caches. Based on the comprehensive analysis using the framework, a memory failure mitigation scheme is developed to improve the energy efficiency of NTC caches.

2 Functional Failure and Reliability Issues of NTC Memory Components

The increase in sensitivity to process variation of NTC circuits affects not only the performance but also functionality. Notably, the mismatch in device strength due to process variation affects the state of positive feedback loop based storage elements (SRAM cells) [3, 10, 14]. The mismatch in the transistors makes SRAM cells to incline for one state over the other, a characteristic that leads to hard functional failure or soft timing failure [17, 20]. The variation-induced functional failure rate of SRAM cells is more pronounced in the nanoscale era as highly miniaturized devices are used to satisfy the density requirements [1]. SRAM cells mainly suffer from three main unreliability sources: (1) aging effects, (2) radiation-induced soft error, and (3) variation-induced functional failures [19]. The SRAM cell susceptibility to these issues increases with supply voltage downscaling.

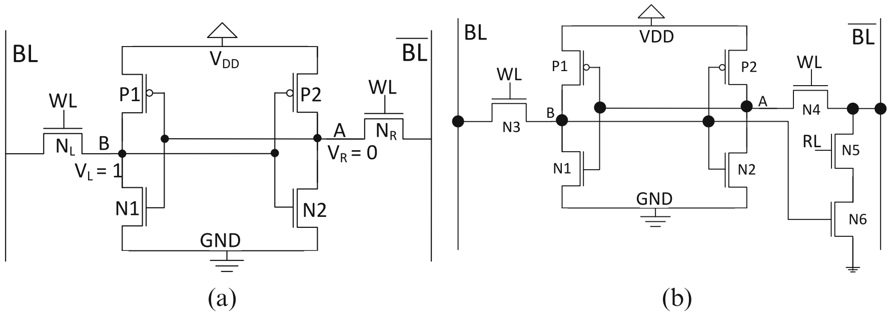


Fig. 2 Schematic diagram of 6T and 8T SRAM cell, where WL = word-line, BL = bit-line and RL = read-line. (**a**) 6T cell design. (**b**) 8T cell design

2.1 Aging Effects in SRAM Cells

Accelerated transistor aging is one of the main reliability concerns in CMOS devices. Among various mechanisms, Bias Temperature Instability (BTI) is the primary aging mechanism in nanoscale devices [18]. BTI gradually increases the threshold voltage of a transistor over a long period, which in turn increases the gate delay [18]. BTI-induced threshold voltage shift is a strong function of temperature as it has an exponential dependency. Hence, BTI-induced aging rate is higher at high operating voltage and temperature values. In SRAM cells, BTI reduces the Static Noise Margin (SNM)[1] of an SRAM cell, and makes it more susceptible to failures. BTI-induced SNM degradation is higher when the cell stores the same value for a longer period (e.g., storing “0” at node “A” of the SRAM cell shown in Fig. 2a). Hence, the effect of BTI on an SRAM cell is a strong function of the cell’s Signal Probability (SP).[2]

2.2 Process Variation in SRAM Cells

Variation in transistor parameters such as channel length, channel width, and threshold voltage results in a mismatch in the strength of the transistors in an SRAM cell, and in extreme cases it makes the cell to fail [15]. The variation-induced memory failure rate increases significantly with supply voltage downscaling, for instance, SRAM cells operating at NTC (0.5 V) have 5$\times$ higher failure rate than the cells operating at a nominal voltage [15]. Process variation affects several aspects of SRAM cells, and the main variation-induced SRAM cell failures are:

[1]SNM is the minimum amount of DC noise that leads to a loss of the stored value.

[2]Probability of storing logic “1” in the SRAM cell.

Read Failure Read failure/disturb is a phenomenon where the stored value is distorted during read operation. For example, when reading the value of the cell shown in Fig. 2a, (V_L = "1" and V_R = "0"), due to the voltage difference between the access transistor N_R and pull-down transistor N_2, the voltage at node V_R increases [21, 39]. If this voltage is higher than the trip voltage (V_{trip}) of the left inverter, then the stored value of the cell is changed. Hence, the condition for read failure is expressed as [33]:

$$\text{read failure} = \begin{cases} 1, & \text{if } V_R > V_{trip} \\ 0, & \text{otherwise} \end{cases}$$

where $V_{trip} = V_{P_1} - V_{N_1}$ (here V_{P_1} and V_{N_1} indicate the voltages of the PMOS and NMOS transistors of the left inverter shown in Fig. 2a where P_1 and N_1 are the corresponding PMOS and NMOS transistors of the inverter).

Write Failure Write failure occurs when the cell is not able to write/change its state with the applied write voltage. For example, during a write operation (e.g., writing "0" to the SRAM cell shown in Fig. 2a), the node V_L is discharged through the bit-line BL. Write failure occurs when the node V_L is not reduced to be lower than V_{trip} of the right inverter (V_R) [21, 33]. In the standard 6T SRAM cell, write failure is a challenging issue as the cell cannot be optimized without reducing its read margin [21, 33, 39]. However, this is improved with the help of read/write assist circuitries or differential read/write access as it is done in the 7T, 8T, and 10T SRAM cell designs [3, 8, 10]. In order to illustrate the write failure issue, the write margin behaviors of 6T and 8T NTC SRAM cells are studied and compared in Fig. 3. As shown in the figure, the 6T SRAM cell has a smaller write margin as it

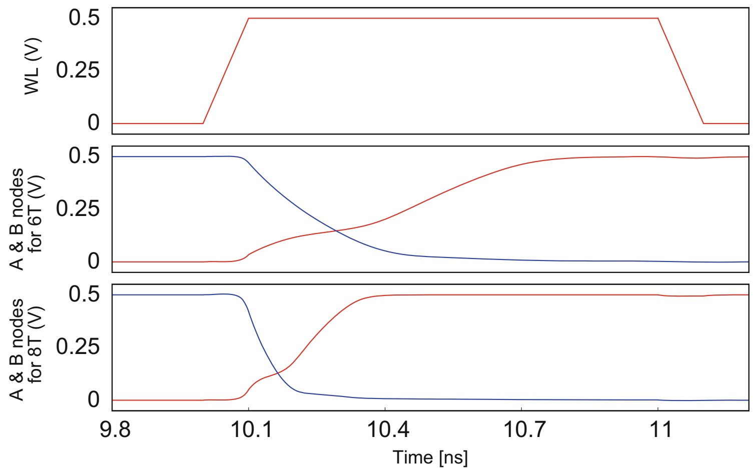


Fig. 3 Write margin (in terms of write latency) comparison of 6T and 8T SRAM cell operating in near-threshold voltage domain (0.5 V)

has longer write latency. On the other hand, the short write latency of the 8T design enables it to have a relatively larger write margin. The improvement in the write margin is because the 8T cell is optimized to improve the write operation without affecting its read operation, as the write and read operations are decoupled.

Hold Failure Hold failure commonly known as metastability issue is a reliability issue that occurs when the SRAM cell is not able to store the value for a longer period [20, 33]. This problem happens during a standby mode if the voltage at nodes V_L or V_R is smaller (smaller SNM value), then the stored value is easily destroyed by a noise voltage due to various sources such as particle strike and leakage current [20, 33].

2.3 *Soft Error Rate in SRAM Cells*

In SRAM cells, soft error is a transient phenomenon that occurs when charged particles penetrate the cell's cross junction creating an aberrant charge that changes the state of the cell [27]. The primary source of soft errors is related to cosmic ray events such as neutrons and alpha particles. Atmospheric neutrons are one of the higher flux components, and their reaction has a high energy transfer. Thus, neutrons are the most likely cosmic radiations to cause soft errors [16, 19]. Neutrons do not generate electron-hole pairs directly. However, their interaction with the Si-atoms generates secondary particles. These secondary particles produce charges/electron-hole pairs [16]. If the generated charges are larger than the *critical charge*[3] of an SRAM cell, then the internal value of the cell is inverted, this phenomenon is commonly referred to as soft error.

Radiation-induced Soft Error Rate (SER) of an SRAM cell increases significantly with decrease in the supply voltage. Previous experiments have shown that the radiation-induced SER increases by 50% for just 20% decrease in the supply voltage [40]. Moreover, the SER of NTC designs is affected by variation and aging-induced SNM degradation.

2.4 *Interdependence and Combined Effects*

Analyzing failures based on a particular reliability failure mechanism is insufficient for estimating the system-level reliability as the interdependence among different failure mechanisms (such as aging, soft error, and process variation) has a considerable impact on the overall system reliability [4, 19, 20]. Figure 4 shows how the interdependence between different reliability mechanisms (aging, SER, and process

[3]Minimum amount of charge required to upset the stored value, of an SRAM cell.

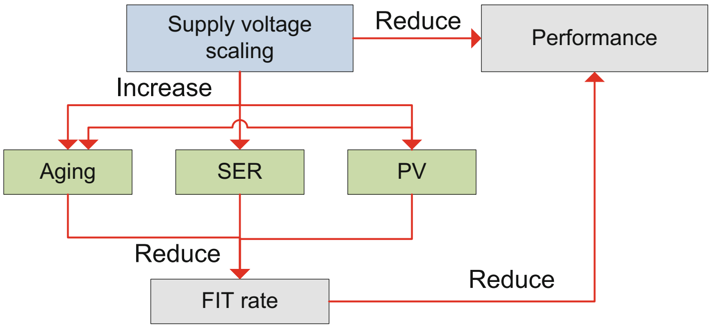


Fig. 4 Interdependence of reliability failure mechanisms and their impact on the system Failure In-Time (FIT) rate in NTC

variation) affects the overall system reliability of memory components in terms of Failure In-Time (FIT rate). As shown in the figure, variation-induced threshold voltage shift increases both aging and SER by reducing the SNM and critical charge of the cell. Similarly, aging-induced SNM degradation increases the sensitivity of SRAM cell to soft errors. The problem is more pronounced when the SRAM cell is operating at NTC domain due to the wide variation extent and higher sensitivity to aging effects [19]. It has been observed that aging has $\approx$5% SNM and critical charge degradation at NTC while process variation-induced SNM degradation reaches as high as 60% [19]. In the super-threshold voltage domain (1.0 V), however, the aging effect increases by $3\times$ to be 15% while variation effect is reduced significantly.

Moreover, the running workload affects the aging rate and SER of memory components, as it determines the signal probability and the Architectural Vulnerability Factor (AVF)[4] of the memory elements [19]. Therefore, to overcome these reliability challenges and improve the overall system reliability, combined analysis of the reliability failure mechanisms at different levels of abstraction is imperative. Besides, the cross-layer analysis should consider the impact of workload on signal probability as well as architectural vulnerability factor of memory components, and their circuit-level consequences on critical charge and SNM degradation.

2.5 Technology Scaling Effects on SRAM Reliability

Reliability has been an essential issue with the miniaturization of CMOS technology, as different design-time and runtime failures are among the limiting factors of technology scaling [24]. At smaller technology nodes, process variation increases the permanent and transient failures of memory components significantly [11, 15].

[4]AVF is the probability that an error in memory structure propagates to the data path. AVF = vulnerable period/total program execution period.

The authors in [15] show that SRAM cell failure rate increases by more than 2× with downscaling from 90 to 65 nm technology node. Similarly, the authors in [26] demonstrated that technology downscaling increases the radiation-induced soft error rate of SRAM cells significantly.

3 Cross-Layer Reliability Analysis Framework for NTC Caches

The comprehensive cross-layer reliability estimation framework that abstracts the impact of workload, cache organization, and reliability failure mechanisms at different levels of abstraction is illustrated in Fig. 5. The reliability analysis and simulation conducted in this work use the symmetric six-transistor (6T) and 8T SRAM cells shown in Fig. 2a and b. In this work, the device-level critical charge characterization is modeled according to the analytical model presented in [27].

This section presents the cross-layer reliability estimation framework in a top-down manner. The system-level *Failure In-Time* (FIT) rate and SNM extraction are described in Sect. 3.1 followed by the cross-layer SNM and SER estimation in Sect. 3.2.

3.1 System FIT Rate Extraction

The system-level FIT rate of a cache memory is the sum of the FIT rate of each row (cache line). The row FIT rate is calculated as the product of the row-wise SER (extracted based on the circuit-level SER information) and its

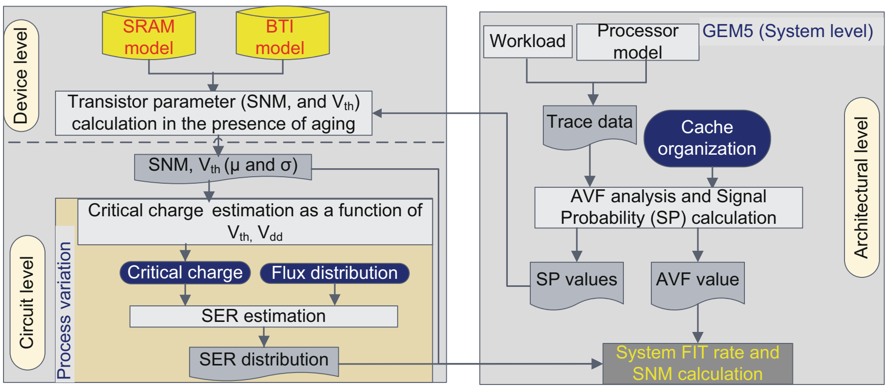


Fig. 5 Holistic cross-layer reliability estimation framework to analyze the impact of aging and process variation effects on soft error rate

Architectural Vulnerability Factor (AVF). Cache AVF is a metrics used to determine the probability that an error in a cache memory propagates to the datapath, and results in a visible error in a program's final output [38]. Equation (1) shows the system-level FIT rate calculation of cache memories.

$$\mathrm{FIT}_{\mathrm{system}} = \sum_{i=0}^{N-1} \mathrm{AVF}_i \times \mathrm{SER}_i \tag{1}$$

where N is the total number of rows in the cache.

3.1.1 Architecture-Level AVF Analysis

One step of determining the failure rate of memory (cache) due to soft errors is to determine the AVF value of the memory. AVF of a memory array is measured by the ratio of vulnerable periods, time interval in which the memory content is exposed to particle strike, to the total program execution period, and the probability of the erroneous value being propagated [38]. Hence, the vulnerability factor of a memory array is computed based on the liveness analysis commonly known as Architectural Correct Execution (ACE) analysis which is the ratio of ACE (vulnerable) cycles to the total number of operational cycles [42]. Therefore, the AVF value of a memory array with M cells is computed as shown in Eq. (2).

$$\mathrm{AVF}_{\mathrm{array}} = \frac{\sum_{i=0}^{M-1} \mathrm{ACE}_i}{T \times M} \tag{2}$$

where T is the total number of cycles.

3.1.2 Architecture-Level SNM Analysis

Aging-induced SNM degradation of an SRAM cell strongly depends on the Signal Probability (SP) of the cell. Thus, BTI-induced SNM degradation is minimized when the signal probability of the cell is balanced (close to 0.5) [18]. In order to determine the aging-induced SNM degradation, the worst-case SP of the memory row is obtained as the maximum SP distance from 0.5 ($D = |\mathrm{SP} - 0.5|$) as shown in Eq. (3). Then, the worst-case SP is used by the SNM estimation tool given in Fig. 5 to determine the corresponding aging-induced SNM degradation.

$$\mathrm{SP}_{\text{worst-case}} = \mathrm{MAX}_{i=1}^{Z} D_i \tag{3}$$

where $D_i = |\mathrm{SP}_i - 0.5|$ and Z is the total number of cells in the memory row.

In order to extract the AVF and SNM of a cache unit, first, it is necessary to extract the trace of the data stored in the cache, read-write accesses, and the duration (number of cycles) of the running workload. Once the information is available, the reliability analysis tool uses it along with the cache organization to determine the AVF and SP of the cache memory according to Eqs. (2) and (3), and generates the SNM LUT for different signal probability values.

The cache organization (size and associativity) has significant impact on the SER and SNM of the cache, as it determines the hit ratio and the duration data is stored in a cache entry. Hence, different cache size and associativity combinations result in different SER and SNM values for the same workload application. Additionally, SER and SNM are highly dependent on the running workload. In order to explore the impact of cache organization and workload, various organizations and workload applications are investigated.

3.2 Cross-Layer SNM and SER Estimation

3.2.1 SNM Degradation Estimation

Device-Level Aging Analysis

BTI-induced aging degrades the carrier mobility of CMOS transistors, and leads to transistor threshold voltage (V_{th}) shift. In an SRAM cell, the V_{th} shift reduces the noise tolerance margin of the cell, and makes it more susceptible to failures. In the reliability analysis framework, the BTI-induced threshold voltage shift of the transistors in an SRAM cell is evaluated at device-level using a Reaction-Diffusion (RD) model [28]. Then, the device-level V_{th} shift results are used to estimate the corresponding SNM degradation of an SRAM cell at the circuit-level.

Circuit-Level SNM Estimation

The SNM of an SRAM cell is extracted by conducting a circuit-level SPICE simulation. The SPICE simulation uses device-level aging and architecture-level SP results to determine the SNM of the SRAM cell. Finally, the SNM degradation of a particular SP value is obtained according to Eq. (4).

$$\mathrm{DEG}_{\mathrm{SP}} = \frac{\mathrm{SNM}_{\mathrm{SP}} - \mathrm{SNM}_{\mathrm{fresh}}}{\mathrm{SNM}_{\mathrm{fresh}}} \times 100\% \tag{4}$$

where $\mathrm{SNM}_{\mathrm{SP}}$ is the SNM of the SRAM cell for a particular signal probability value and $\mathrm{SNM}_{\mathrm{fresh}}$ is the SNM of a fresh (new) SRAM cell.

Aging and Process Variation-Induced SNM Degradation Analysis

BTI-induced SNM degradation of an SRAM cell depends not only on the cell signal probability but also on process parameters, such as channel length and

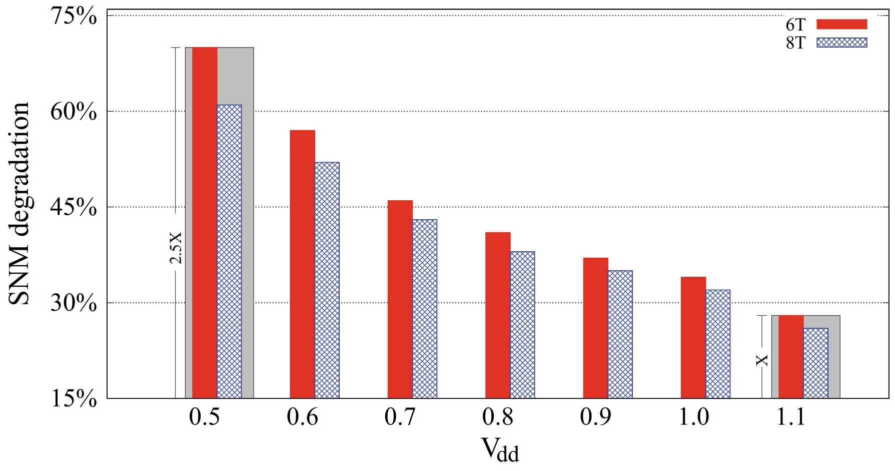


Fig. 6 SNM degradation in the presence of process variation and aging after 3 years of operation, aging+PV-induced SNM degradation at NTC is 2.5× higher than the super-threshold domain

oxide thickness, which are highly affected by manufacturing variabilities. Due to low operating temperature at NTC, aging has relatively less impact on the SNM degradation of near-threshold voltage SRAM cells. However, in combination with variation-induced threshold voltage shift, aging degrades the SNM of SRAM cells significantly.

Figure 6 shows the worst-case aging (SP = 0.0) and variation-induced SNM degradation of 6T and 8T SRAM cells after 3 years of operation for wide supply voltage range. The obtained SNM degradation confirms the analytical expectation as the SNM degradation in NTC is 2.5× higher than the degradation in the super-threshold voltage domain (as shown by the gray boxes). While the use of 8T instead of 6T SRAM cells in super-threshold voltage domain has limited improvement in SNM degradation (only 7.7%), it achieves more than 14% reduction in the SNM degradation in the near-threshold voltage domain.

3.2.2 SER Estimation

The SER of an SRAM cell depends on two main factors, the critical charge of the cell and the flux rate of the strike. To determine SRAM cell SER, first, the critical charge of an SRAM cell is obtained from a circuit-level model. Then, the SER value is calculated by combining the critical charge, flux distribution, and the area sensitive to strike.

Device-Level Critical Charge Characterization

The sensitivity of an SRAM cell to radiation-induced soft errors is determined by the critical charge (Q_{critical}) of the cell, as it determines the minimum amount of charge required to alter the state of the cell. The Q_{critical} of an SRAM cell depends on several factors such as supply voltage, threshold voltage, and strength of the transistors of the SRAM cell [9]. The critical charge of an SRAM cell is computed using analytical models or circuit simulators. An analytical model developed in [27] is used to determine the Q_{critical}.

As shown in Fig. 5, the SPICE model of an SRAM cell along with the BTI model is employed to evaluate the impact of BTI on the threshold voltage (V_{th}) of the transistors of an SRAM cell. The BTI analysis uses the SP values of the memory array from higher (architecture-level) analysis to determine the BTI-induced V_{th} shift of the running workload. In this way, the aging effect of the workload is incorporated into the framework. Once the fresh and aged V_{th} values are available, the impact of process variation is incorporated as a normal distribution ($\mu \pm 3\sigma$) of the transistor threshold voltage where μ is the mean V_{th} value and the standard deviation (σ) which is obtained using an industrial standard, measurement based, model (the "Pelgrom model") given in Eq. (5) [30]. Finally, all these parameters are used by the model given in [27] to extract the Q_{critical}.

$$\sigma \Delta V_{th} = \frac{A_{VT}}{\sqrt{L \times W}} \tag{5}$$

where L and W are the length and width of transistors, and A_{VT} is process specific parameter (the "Pelgrom coefficient").

Circuit-Level SER Analysis

The circuit-level SER analysis is conducted using the SER extraction module of the framework given in Fig. 5. First, the critical charge of the SRAM cell is extracted using the device-level model [27]. Afterward, the critical charge along with the neutron-induced flux distribution is used to determine the SER of the cell using an experimentally verified empirical model given in Eq. (6) [23]. As shown in Eq. (6), the SER of an SRAM cell has an inverse exponential relation with its critical charge (Q_{critical}). Hence, the higher the Q_{critical}, the lower the SER will be.

$$\text{SER} \propto FAe^{\left(-\frac{Q_{\text{critical}}}{Q_s}\right)} \tag{6}$$

where F is the flux in particles/cm^2-s with energy higher than 1 MeV [6]; A is the area sensitive to a strike in cm^2, and Q_S is the charge collection efficiency.

The main observations from Eq. (6) are:

- The SER of an SRAM cell has an inverse exponential relation to its critical charge. Hence, a small decrease in the $Q_{critical}$ leads to an exponential increase in the cell SER.
- For the same atmospheric neutrons, a small drift in $Q_{critical}$ leads to a significant increase in the SER. Furthermore, transistor up-sizing increases the area which is sensitive to particle strike and hence, higher SER.

SER of 6T and 8T SRAM Cells

In the conventional 6T SRAM cell, the cell must maintain the stored value and it should be stable during read/write accesses. SRAM cell stability is a challenging task when the cell is operating in the near-threshold voltage domain, as the cell mainly suffers from read-disturb. To address this issue, either a read-write assist circuitry should be employed or the pull-down (NMOS) transistors of the SRAM cell should be strengthened by transistor up-sizing [35]. However, the up-sizing also increases the area of the cell that is sensitive to soft errors. Since the read-disturb of the 6T SRAM cell is worst when it operates at lower voltage values, transistor up-sizing cannot adequately mitigate the read-disturb issue which makes the 6T design less desirable for near-threshold voltage operation.

This issue is addressed by using alternative SRAM cell designs (such as 8T [32] and 10T [8] SRAM cells). For example, the read failure issue is solved in the 8T design by decoupling the read and write lines using two additional NMOS access transistors. The decoupling allows to downsize the pull-down NMOS transistors, and reduce the area sensitive to soft errors. Therefore, alternative SRAM designs (e.g., 8T) are recommended for NTC operation, which is verified by studying the reliability and energy efficiency improvement of the 8T SRAM design over the conventional 6T design. The transistor sizing specified in [32] is used for the design of the 6T and 8T SRAM cells used in this study.

Figure 7 shows the fresh and aged SER of the 6T and 8T SRAM designs for different supply voltage values. In the super-threshold voltage domain, (0.9–1.1 V) the 6T and 8T designs have negligible differences in their SER. In NTC, however, the 6T design has higher SER than the 8T design due to the effects of transistor up-sizing which increases the area sensitive to radiation. The combined effect of aging and process variation on 6T and 8T SRAM cells is shown in Fig. 8. Figure 8 shows variation effect has severe impact at NTC, as the SER of the 6T and 8T SRAM cell designs in the near-threshold voltage domain is 4× higher than their SER in the super-threshold voltage domain.

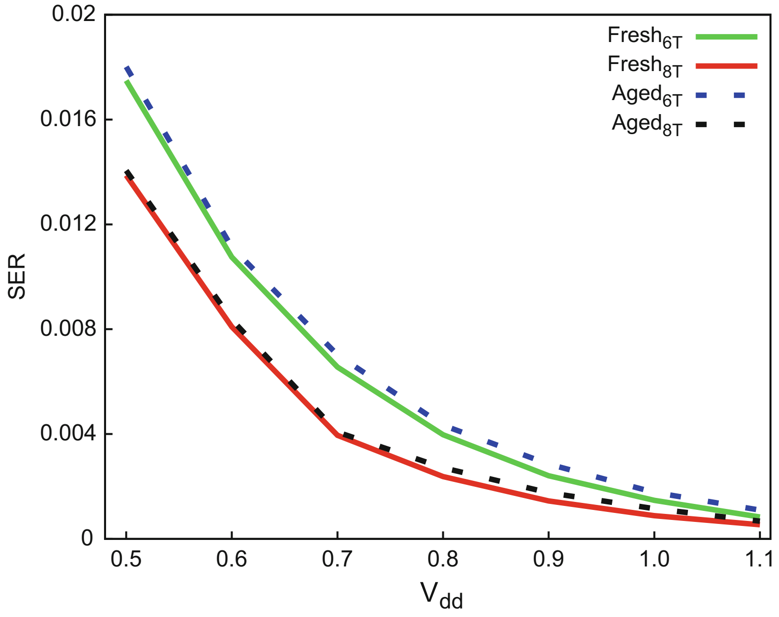


Fig. 7 SER rate of fresh and aged 6T and 8T SRAM cells for various V_{dd} values

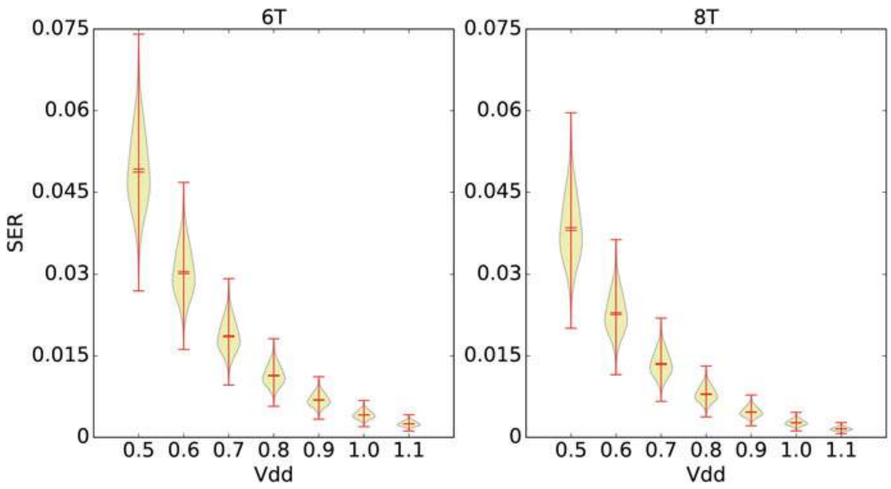


Fig. 8 SER of 6T and 8T SRAM cells in the presence of process variation and aging effects after 3 years of operation

3.3 Experimental Evaluation and Trade-Off Analysis

3.3.1 Experimental Setup

The reliability analysis is conducted using an ALPHA implementation of an embedded in-order core on the Gem5 architectural simulator [7]. Since cache memories are the main focus, various cache sizes (4–16 KB) and wide associativity range from simple directly mapped to 4-way set associative caches are assessed to perform a reliability and performance trade-off analysis. The evaluation is conducted using several workload applications from the SPEC2000 CPU benchmark suite [25]. The workload applications were executed for five million cycles by fast-forwarding to the memory intensive phases. The experimental setup used in this work is presented in Table 1.

The BTI-induced V$_{th}$ shift is extracted by assuming 10% BTI-induced aging after 3 years of operation [37]. First, the 45 nm 6T and 8T SRAM cells are modeled using the PTM model. Afterward, the BTI-induced V$_{th}$ shift LUT and the corresponding SNM degradation for various SP values (0.0–1.0) are obtained using a SPICE simulation. The impact of process variation is considered as a normal distribution of the transistor threshold voltage with a mean ($\mu = V_{th}$, 300 mV) and standard deviation (σ) obtained using the Pelgrom model given in Eq. (5).

To demonstrate the effect of soft error, neutron-induced soft errors are considered as they are the dominant soft error mechanisms at terrestrial altitudes. In order to ensure the proper functionality of both 6T and 8T SRAM cells in the near-threshold voltage domain, their transistors are sized according to the transistor sizing used to model and fabricate near-threshold 6T and 8T SRAM cells specified in [32]. It should be noted that L1 cache is used for illustration purpose only as most embedded

Table 1 Experimental setup, configuration, and evaluated benchmark applications

	Gem5	
Simulation environment	Near-threshold	Super-threshold
Core configuration		
Processor model	Embedded	Embedded
Architecture	Single in-order core	Single in-order core
ISA	ALPHA	ALPHA
Supply voltage	0.5 V	1.1 V
Frequency	100 MHz	1 GHz
Technology node	45 nm PTM	45 nm PTM
Cache configuration		
L1 Cache	Sizes = 4, 8, and 16 KB	Sizes = 4, 8, and 16 KB
	Associativity = 1, 2, and 4 way	Associativity = 1, 2, and 4 way
	Replacement policy = LRU	Replacement policy = LRU
	SRAM cells = 6T and 8T	SRAM cell = 6T
Benchmark	SPEC2000	SPEC2000

NTC processors have limited cache hierarchy. However, the framework is generic, and it is applicable to any cache levels such as L2 and L3.

3.3.2 Workload Effect Analysis

As discussed in Sect. 3.2, BTI-induced SNM degradation of SRAM cell highly depends on the cell's signal probability and the residency time of valid data which varies from one workload application to another. Similarly, the SER of memory components is dependent on the data residency period which is commonly measured using AVF. Hence, for SER analysis, the AVF of different workloads is obtained based on the workload application's data residency period. In order to show the effect of workload variation on SER and SNM degradation, the AVF and signal probabilities of the cache memory are extracted by running different workload applications from the SPEC2000 benchmark suite. Then, the corresponding SNM and SER of the cache memory are obtained using the SER and SNM models presented in Sect. 3.2.

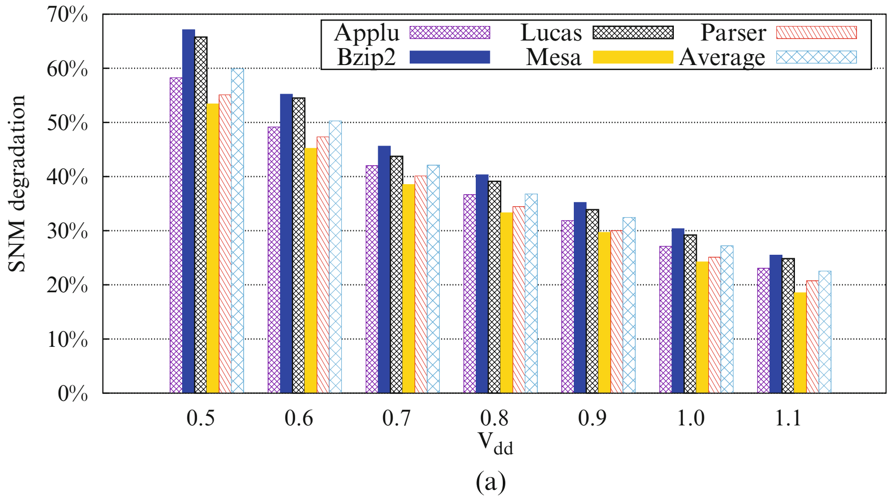
3.3.3 Aging and Variation-Induced SNM Degradation

SNM degradation affects the metastability of SRAM cells. Metastability of SRAM cell determines the stability of the stored value, and it is highly dependent on the worst-case SNM degradation [18]. Therefore, for any workload application, the aging-induced SNM degradation should be evaluated based on the first cell to fail (worst-case SNM degradation).

The impact of workload on the SNM degradation of 6T and 8T based caches across wide supply voltage range is shown in Fig. 9a and b, respectively. For both cases, the SNM degradation increases significantly with supply voltage downscaling. Although the aging rate is slower at lower supply voltage values due to the lower temperature, the wide variation extent in NTC leads to higher aging sensitivity. Hence, in NTC the impact of process variation on SNM is more severe and leads to a significant increase in the aging sensitivity of SRAM cells.

3.3.4 Soft Error Rate Analysis

In order to analyze the impact of workload variation on the soft error rate of cache memories, the architectural vulnerability factor of each workload is extracted and combined with the circuit-level information. Figure 10 shows the contribution of the SPEC2000 workload applications on the SER of the 6T SRAM based cache. As shown in the figure, for all workload applications the SER increases significantly with supply voltage downscaling. For example, the SER of all workload applications increases by five orders of magnitude when the supply voltage is downscaled from the super-threshold voltage (1.1 V) to the near-threshold voltage domain (0.5 V).



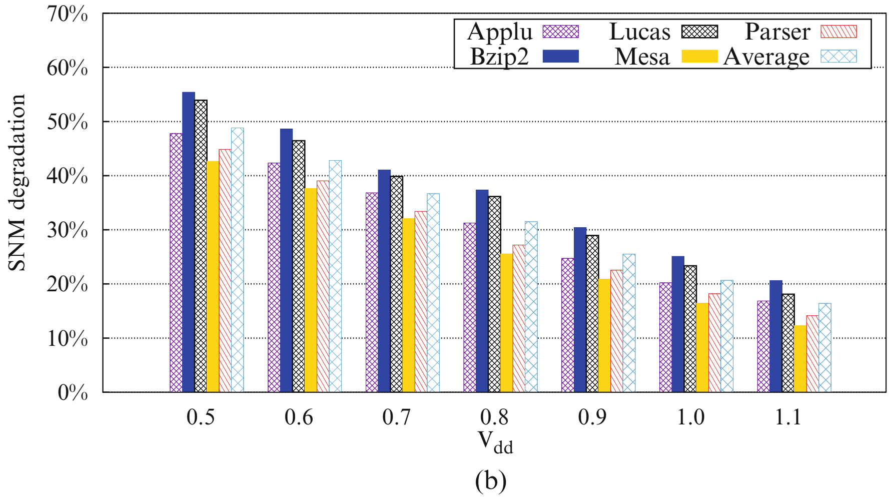


Fig. 9 Workload effects on aging-induced SNM degradation in the presence of process variation for 6T and 8T SRAM cell based cache after 3 years of operation (**a**) 6T SRAM based cache (**b**) 8T SRAM based cache

Additionally, the workload variation has a considerable impact on the soft error rate. For example, the SER of *Bzip2* is almost two orders of magnitude higher than the SER of *Mesa* and *Parser* workload applications. The workload variation impact is observed because *Bzip2* application has higher locality and hit rate which increases the data residency period when compared to the other workload applications. Although the higher hit rate of *Bzip2* leads to a better performance measured in Instructions Per Cycle (IPC), it has a significant impact on the soft error rate of the cache. Hence, it is essential to exploit the workload variation in

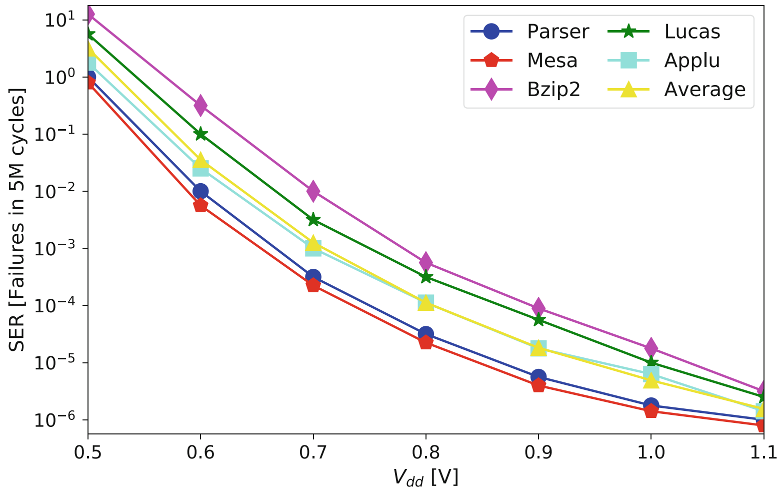


Fig. 10 Workload effect on SER rate of 6T SRAM cell based cache memory for wide supply voltage range

order to downscale the supply voltage of the cache memory in per-application bases for a given target error rate. For a given target FIT rate (e.g., 10^{-2}) the cache has to operate at 0.6 V for *Mesa* and *Parser* workload applications. However, for Bzip2 the cache has to operate at a higher voltage (0.7 V) for the same target error rate.

3.3.5 Cache Organization Impact on System FIT Rate

Cache organization has a significant impact on the performance of embedded processors [34]. Similarly, the organization has an impact on the reliability of cache units. In NTC, the reliability impact of cache organization is even more pronounced. Hence, a proper cache size and associativity selection should consider both performance and reliability as target metrics. The system failure probability (FIT rate and SNM) of a cache unit is highly dependent on the architectural vulnerability factor and the values stored in the cache as well as their residency time intervals, which is in turn is a strong function of the read-write accesses of the cache. Hence, these parameters are influenced by cache size and associativity.

The performance and reliability impacts of different cache organizations in the near and super-threshold voltage domains are evaluated using the configurations described in Table 1. For near-threshold voltage (0.5 V) the processor core frequency is set to 100 MHz, and the cache latency is set to 1 cycle as gate delay is the dominant factor in the near-threshold voltage domain [12]. In the super-threshold voltage domain, however, the cache latency and interconnect delay have a significant impact on the overall delay. Thus, the cache hit latency is set to 2 cycles for 4 and 8 K cache sizes and 3 cycles for the 16 K cache size [41].

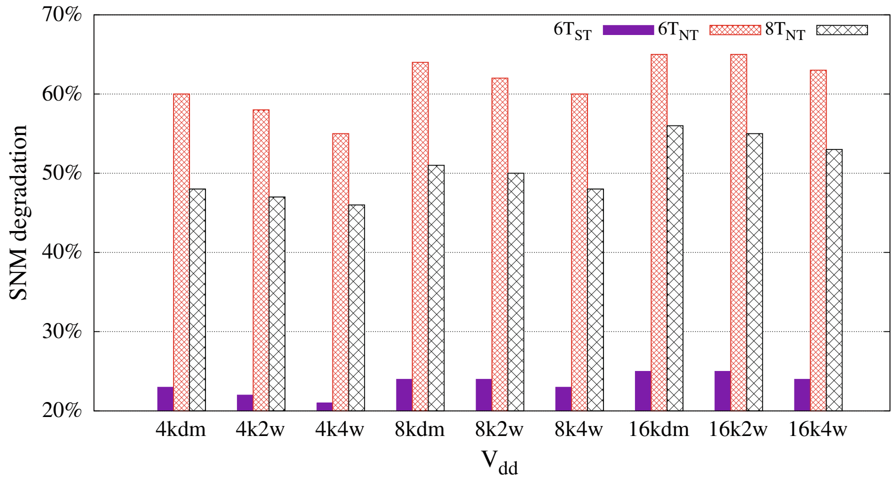


Fig. 11 Impact of cache organization on SNM degradation in near-threshold (NTC) and super-threshold (ST) in the presence of process variation and aging effect after 3 years of operation

3.3.6 Cache Organization and SNM Degradation

Since cache organization determines the data residency period, it has a direct impact on the SNM degradation. Figure 11 illustrates the impact of cache organization on the SNM degradation of near and super-threshold voltage 6T and 8T SRAM cell based memory arrays in the presence of process variation and aging effects after 3 years of operation. The figure shows smaller cache size with higher associativity (4 k-4 w) has less impact on SNM degradation as the data resides in the cache for a smaller duration.

3.3.7 Cache Organization and SER FIT Rate

The cache size and associativity also affect the ACE cycles of cache lines and their failure probabilities. The impact of the cache organization on the FIT rate and performance (IPC) varies along various supply voltage domains. In the super-threshold voltage, an increase in cache size and associativity improves the performance. However, from a FIT rate point of view, an increase in the cache size has a negative impact on FIT rate as it increases the FIT of the cache. Smaller cache sizes, however, have lower performance and better FIT rate. Figure 12 shows the design space of FIT rate and performance (IPC) impact of various cache organizations in the super-threshold voltage domain. In the figure, the FIT rate and performance optimal configuration is (8 k-4 w) as indicated by the blue italic font in Fig. 12.

In the near-threshold voltage domain, the performance is mainly dominated by the delay of the logic unit and the memory failure rate is significantly high. Therefore, it is essential to select a cache organization that gives better reliability

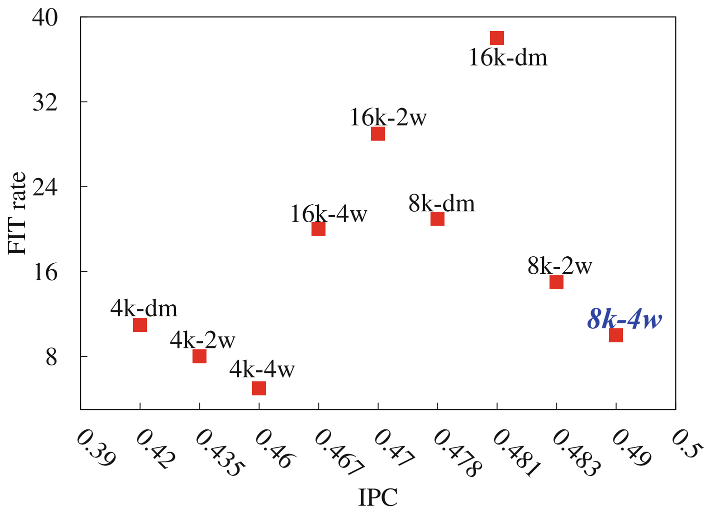


Fig. 12 FIT rate and performance design space of various cache configurations in the super-threshold voltage domain by considering average workload effect (the *blue italic font* indicates optimal configuration)

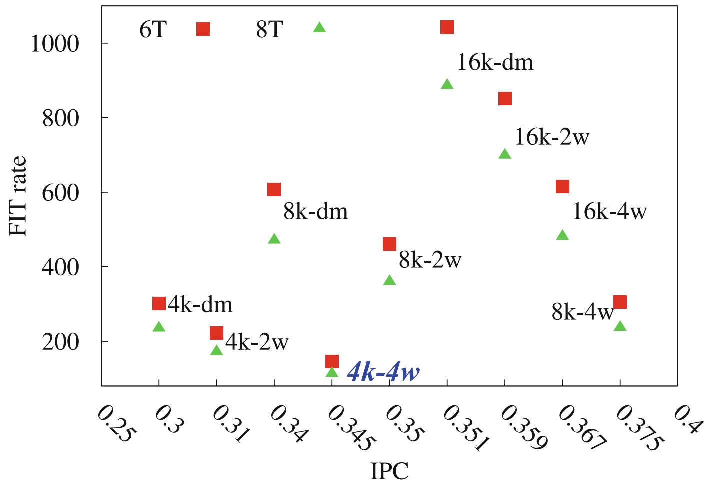


Fig. 13 FIT rate and performance design space of 6T and 8T designs for various cache configurations in the near-threshold voltage domain by considering average workload effect (the *blue italic font* indicates optimal configuration)

(FIT rate and SNM) than performance. Hence, in NTC a smaller cache size with higher associativity gives the best reliability and performance trade-off. Figure 13 shows the design space for the FIT rate and performance trade-off for 6T and 8T designs in NTC.

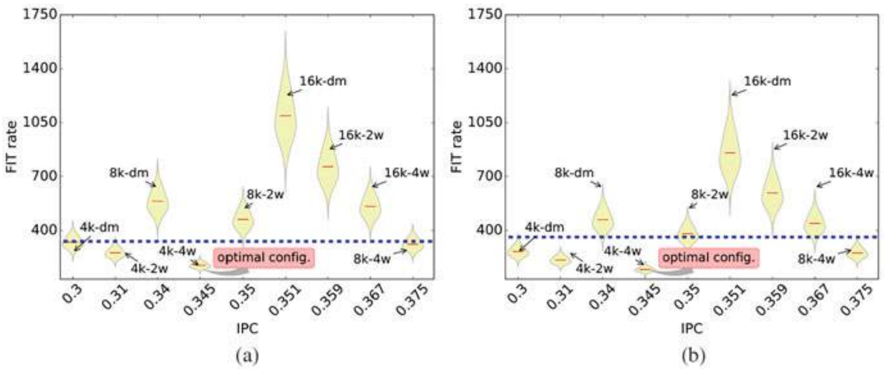


Fig. 14 FIT rate and performance trade-off analysis of near-threshold 6T and 8T caches for various cache configurations and average workload effect in the presence of process variation and aging effects. (**a**) Near-threshold 6T. (**b**) Near-threshold 8T

3.3.8 Reliability-Aware Optimal Cache Organization

The experimental results reported in Figs. 11, 12, 13, and 14 show an increase in the cache associativity improves the performance and reliability (both FIT rate and SNM). Hence, in the super-threshold voltage domain, medium cache size (e.g., 8 KB) with higher associativity has a better reliability and performance trade-off. In NTC, however, smaller cache sizes with higher associativity are preferable for two main reasons: (1) The performance is mainly dominated by the processor core, not by the cache units and hence, cache latency is not an important issue. (2) The soft error rate and SNM degradation are higher in NTC than in the super-threshold voltage domain. Hence, the cache size is reduced by half to obtain a better reliability and performance trade-off in NTC.

In the NTC domain, the selection of an optimal cache organization for the 6T SRAM cell based caches is different from the 8T based caches, depending on the FIT rate and performance requirement. For example, for a target tolerable FIT rate of 350 at NTC (as shown by the dotted line in Fig. 14a and b), only 4 KB 4-way associative cache organization is within the acceptable zone for the 6T-based cache. In the 8T-based cache, however, three additional cache organizations (4 K-dm, 4 k-2 w, and 8 k-4 w) are within the acceptable zone. Hence, the 8 k-4 w cache is used in the 8T-based cache to get ≈10% performance improvement without violating the reliability constraint.

To implement the suggested cache organizations for a specific supply voltage value (only near-threshold or super-threshold) is straightforward. For caches that are expected to operate in both super and near-threshold voltage domains, the reliability-performance optimum cache organization in the super-threshold voltage (e.g., 4-way 8 KB in this case) is preferable. Then, when switching to the near-threshold voltage domain, some portion of the cache is disabled (power gated) in order to maintain the reliability-performance trade-off at NTC.

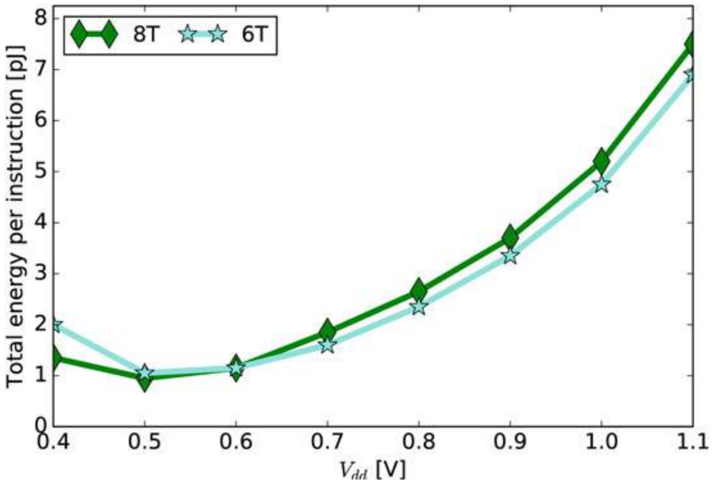


Fig. 15 Energy consumption profile of 6T and 8T based 4 K 4-way cache for wide supply voltage value ranges averaged over the selected workloads from SPEC2000 benchmarks

3.3.9 Overall Energy-Saving Analysis of 6T and 8T Caches

The energy-saving potential of supply voltage downscaling is evaluated by extracting the average energy consumption profile of the 4 K-Byte 4-way set associative cache (i.e., the reliability-performance optimal cache configuration) using 6T and 8T implementations. The energy consumption of the cache memory consists of three different components. These components are peripheries, row and column decoders, and bit-cell array energy consumptions. Since the energy consumption of the periphery and row/column decoder is independent of the bit-cell used, they are assumed to be uniform for both 6T and 8T based caches. Hence, the energy-saving comparison is done based only on the energy consumption of the bit-cell array.

Figure 15 compares the total energy consumption of the 6T and 8T based cache memories for a wide supply voltage range. As shown in the figure, the 8T based cache has slightly higher energy consumption in the super-threshold voltage domain (0.7–1.1 V) than the 6T based cache. The slightly higher energy consumption is because of the additional transistors used for read/write decoupling. However, due to the increase in the failure rate in the near-threshold domain, the 6T based cache consumes more energy than the corresponding 8T based implementation. The energy cost of the higher failure rate is considered as an increase in the read/write latency of the cache. This shows addressing the failures of the 6T cache in NTC results in additional energy cost which makes it less attractive for operating at lower supply voltage values (e.g., below 0.6 V).

3.3.10 Reliability Improvement and Area Overhead Analysis of 8T Based Caches

In a near-threshold voltage SRAM design, the 8T cell improves the soft error rate in the presence of aging and variation effects by up to 25%. Similarly, the SNM is improved by ≈15% using 8T SRAM cells in NTC caches. However, it is expected that the 8T SRAM design has 30% area overhead than the 6T design due to the two additional access transistors. In practice, however, the overhead is much less. Since the 6T SRAM has to be up-sized to increase its read stability, the up-sizing increases the cell area of the 6T design to the extent of being larger than the area of 8T design, as experimentally demonstrated in [32].

4 Voltage Scalable Memory Failure Mitigation Scheme

As shown in the analysis presented in Sect. 3, process variation has a significant impact on the failure rate of memory components operating in the near-threshold voltage domain. Hence, addressing variation-induced memory failures plays an essential role in harnessing NTC benefits. One way of mitigating variation-induced memory failures is by determining the voltage downscaling potential of cache memories without surpassing the tolerable/correctable error margins. For this purpose, the operating voltage of caches should be gracefully reduced so that the number of failing bits due to permanent and transient failures remains tolerable.

This section presents a BIST based voltage scalable mitigation technique to determine an error-free supply voltage downscaling potential of caches at runtime. In order to reduce the runtime configuration complexities, the cache organizations such as size, associativity, and block size are determined during design time. In this work, the block size is considered as the smallest unit used to transfer data to and from the cache. Then, a BIST based runtime cache operating voltage downscaling analysis is performed for a given cache organization. To illustrate the impact of block size selection, the voltage downscaling potential of two block sizes is studied.

4.1 Motivation and Idea

Due to the wide variation extent in NTC, different memory cells have different SNM values; as a result, their minimum operating voltages for a proper functionality vary significantly. The cells with smaller SNM values need to operate at a higher supply voltage than the cells with larger SNM values. Therefore, the supply voltage of some cells (cells with smaller SNM value) should be scaled down more conservatively than the cells with larger SNM in order to maintain the overall reliability. This idea is exploited in order to minimize the effect of process variation and determine error tolerant/error-free voltage downscaling potential of near-threshold caches. Since

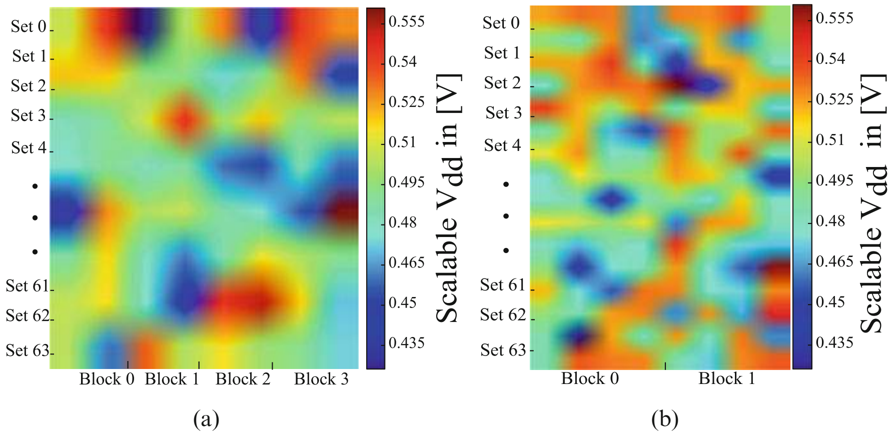


Fig. 16 Error-free minimum operating voltage distribution of 8 MB cache, Set size = 128 Byte (**a**) block size = 32 Bytes (4 blocks per set) and (**b**) block size = 64 Bytes (two blocks per set), the cache is modeled as 45 nm node in CACTI

cache memories are divided into several blocks, block size selection has a significant impact on the supply voltage downscaling potential of cache memories. Hence, one needs to analyze the impact of process variation and supply voltage downscaling potential of cache memories in a per block bases.

Cache block size has a substantial impact on the miss rate and miss penalty of caches at the same time. In order to reduce the cache miss rate and its associated penalty, a larger block size is preferable as it improves locality and reduces the miss rate. From a reliability point of view, however, larger block sizes have wide variation extent, and as a result more failing cells in NTC, which makes the entire block fail. These failures force the cache memory to operate at a much higher voltage (i.e., more conservative scaling) leading to a significant reduction in the energy efficiency. However, this is addressed by decreasing the cache block size in order to reduce cache operating voltage as the variation extent is minimal in comparison to larger block sizes.

To exploit this fact, the impact of block size selection on the supply voltage downscaling potential of a near-threshold voltage 8 KB cache is evaluated as shown in Fig. 16. The cache is modeled in CACTI [36] with 128 Byte set size and two different block sizes, and the impact of process variation is modeled using the threshold voltage variation model given in Eq. (5). As shown in the figure, the smaller block size (Fig. 16a) has narrow variation extent, and hence, it has more supply voltage downscaling potential than its larger block size counterpart (Fig. 16b) at design time. During operation time, the supply voltage downscaling potential of the larger block size cache is reduced further due to various runtime factors such as aging-induced SNM degradation and SER. Moreover, smaller block sizes have lower multiple bit failure rates, and hence, simpler ECC schemes are adopted at a minimum cost [2]. Table 2 shows the ECC overhead comparison for 64 and 32 Byte

Table 2 ECC overhead analysis of different block sizes and correction capabilities

ECC schemes	Block size = 64 Byte			Block size = 32 Byte		
	Area overhead	Storage overhead	Latency overhead	Area overhead	Storage overhead	Latency overhead
SECDED	13k gates	11 bits	2 cycle	≈4k gates	10 bits	1 cycle
DECTED	>50k gates	21 bits	4 cycles	≈10k gates	19 bits	2 cycle
4EC5ED	≈60k gates	41 bits	15 cycles	≈50k gates	37 bits	9 cycle

block sizes according to [2]. The table shows dividing the cache into smaller blocks has an advantage in terms of ECC overhead. Therefore, appropriate cache block size selection should consider both performance and reliability effects at the same time in order to achieve maximum performance while operating within the tolerable reliability margin. Once the cache block size is determined, the cache supply voltage should be tuned at runtime to incorporate the runtime reliability effects such as aging. For this purpose, a BIST based supply voltage tuning is used, and its concept is discussed in the following subsection.

4.2 Built-In Self-Test (BIST) Based Runtime Operating Voltage Adjustment

Built-In Self-Test (BIST) is a widely used technique to test VLSI system on chip [22]. Since memory components occupy majority of the chip area, BIST plays a significant role in testing large and complex memory arrays easily [5, 22]. In order to determine the runtime supply voltage downscaling potential of caches, it is essential to assume a cache memory is equipped with BIST infrastructure to test the entire memory.

In a conventional BIST, the BIST controller generates the test addresses and test patterns (finite number of read/write operations). Then, the test is performed, and the test result is compared with the expected response to determine the failing cells [5]. In this case, however, since the BIST module has to determine the minimum scalable voltage of each block, the test controller has to be modified in order to iteratively test and generate the minimum scalable voltages of each block. The goal is first to determine the error-free minimum scalable voltage of each cache block with/without error correction hardware. Then, the cache operating voltage is determined based on the block with higher operational voltage as shown in Eq. (7), such that the runtime memory failure is minimized.

$$\mathrm{V}_{dd}^{\mathrm{cache}} = \max_{0 \le i \le N-1} \mathrm{V}_{dd}^{B_i} \tag{7}$$

where N is the total number of cache blocks, and $\mathrm{V}_{dd}^{B_i}$ is the runtime minimum scalable voltage of block B_i obtained using the iterative BIST.

Algorithm 1 Runtime cache operating voltage adjustment

```
1: function CACHE-V$_{dd}$-SCALING ($C_s$, $V_{dd}$, $B_s$, $F_m$ ){$C_s$=cache size, $V_{dd}$= operating voltage, $B_s$=block size,
   F$_m$=tolerable margin of failing bits}
2:     B$_t$ $\leftarrow \frac{C_s}{B_s}$;{ B$_t$=total number of cache blocks}
3:     for block i$\leftarrow$1 to B$_t$ do
4:         F$_c$ $\leftarrow$0; {F$_c$=failing cells counter}
5:         V$_{dd}^{new}$ $\leftarrow V_{dd}$; { V$_{dd}^{new}$=voltage used to perform BIST}
6:         while F$_c$ $\leq$F$_m$ do
7:             Perform BIST using V$_{dd}^{new}$;
8:             F$_c$ $\leftarrow$ failing cells;{total number of failing cells per block}
9:             V$_{dd}^{new}$ $\leftarrow$V$_{dd}^{new}$-$\Delta V_{dd}$;{reduce operating voltage by $\Delta V_{dd}$}
10:        end while
11:    end for
12:    V$_{dd}^{cache}$ $\leftarrow \max_{1\leq i\leq B_t} V_{dd_i}^{new}$;{ V$_{dd_i}^{new}$ new operating voltage of block$_i$ }
```

Algorithm 1 presents the iterative BIST technique used to determine the minimum scalable voltage of cache memory by considering permanent and runtime memory failures. The algorithm takes cache size (C_s), operating voltage (V$_{dd}$), block size (B_s), and tolerance margin (F_m) as its input. Then, the number of cache blocks is determined by dividing the cache size by the block size (Step 2). Afterward, the minimum scalable voltage of each block is obtained by gradually reducing the operating voltage, and conducting block-level BIST to determine the total number of failing bits at each operating voltage level (Steps 3–10). It should be noted that, the supply voltage is reduced as long as the number of failing bits per block is within the tolerable/correction capability of the adopted error correction scheme. For example, a cache memory equipped with a Single Error Correction Double Error Detection (SECDED) infrastructure tolerates two failing bits per block (hence $F_m = 2$) as SECDED corrects only one bit and detects two erroneous bits at a time. Hence, whenever two failing bits are detected the error-free version is loaded from the lower-level memory which makes SECDED sufficient solution for tolerating two failing bits per block. Finally, the algorithm determines the operating voltage of the cache based on the block with the highest voltage as shown in Step 12.

The overall flow of the cache access control logic along with the BIST infrastructure as well as mapping logic is presented in Fig. 17. The cache controller first decodes the address and identifies the requested block. Then, it determines if the requested block is functional or failing block for the specified operating voltage. If the requested block is functional, then a conventional block access is performed. In case the requested block is a failing one, the error tolerant block mapping scheme is employed to redirect the access request.

Since this approach considers the effect of permanent and transient failure mechanisms, it is orthogonal with different dynamic cache mitigation schemes such as block disabling [1, 43] and strong ECC schemes [2]. For energy-critical systems, block disabling technique is applied in combination with this approach to downscale the cache operating voltage aggressively by disabling the failing blocks at lower operating voltages at the cost of performance reduction (increase in miss rate).

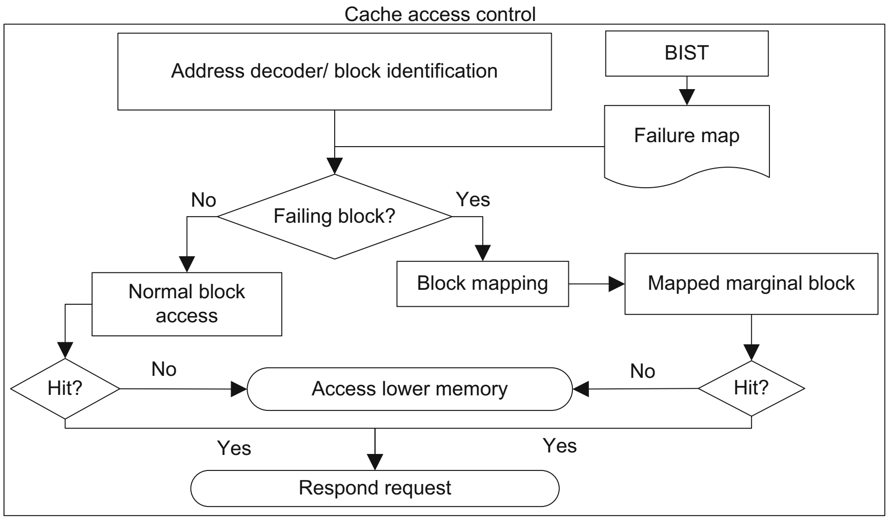


Fig. 17 Cache access control flowchart equipped with BIST and block mapping logic

4.3 Error Tolerant Block Mapping

Once the minimum scalable voltages of the cache blocks are determined, the next task is to disable the failing blocks, and map their read/write accesses to the corresponding non-failing blocks in order to ensure reliable cache operation. Additionally, in order to reduce the vulnerability to runtime failures (such as noise and soft errors), the non-failing blocks are stored in a stack frame sorted by their minimum scalable voltage values. Since the marginal blocks (blocks with less voltage downscaling potential) are more sensitive to runtime failures, they are stored at the top of the stack. Then, access to a disabled block is mapped to the marginal blocks in the stack. The mapping enables to reduce soft error vulnerability of the marginal blocks by reducing their data residency period. Since a stack is a linear data structure in which the insertion and deletion operations are performed at only one end commonly known as "top," the marginal blocks need to be at the top (upper half) of the stack to ensure their fast replacement.

The mapping process is illustrated in Fig. 18 by using an illustrative example. As shown in the figure, the cache blocks are divided into three categories: (1) red blocks are failing blocks. (2) yellow blocks are marginal blocks (non-failing but with limited supply voltage downscaling potential). (3) blue blocks are robust blocks (i.e., non-failing with higher supply voltage downscaling potential). Hence, the marginal blocks are stored at the top of the stack frame. Then, when a disabled (failing) block is requested (e.g., B5) its access request is mapped to a marginal block at the top of the stack frame (e.g., B4), and the stack pointer is updated to point to the next element in the stack. This process continues until all the disabled blocks are mapped. It should be noted that once a block is mapped, it is removed from the mapping stack

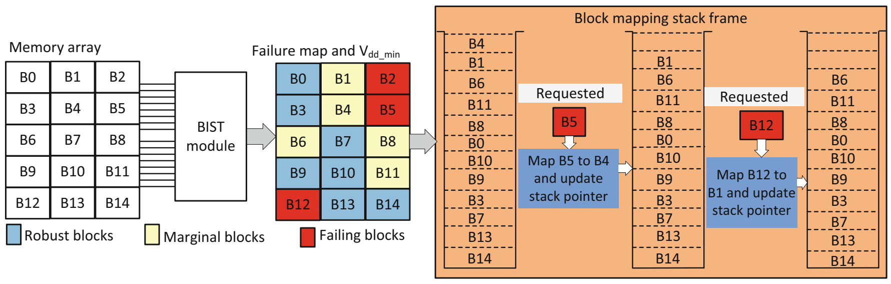


Fig. 18 Error tolerant cache block mapping scheme (mapping failing blocks to marginal blocks)

Table 3 Minimum scalable voltage analysis for different ECC schemes

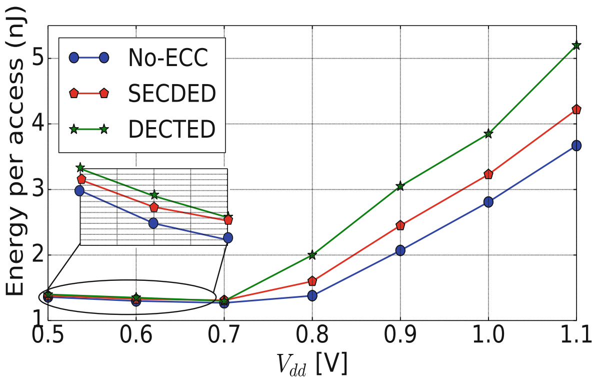
	Minimum scalable voltage in [V]		
ECC-Scheme	Block size = 16Byte	Block size = 32Byte	Block size = 64Byte
No-ECC	0.50	0.53	0.54
Parity	0.47	0.51	0.53
SECDED	0.43	0.48	0.50

when updating the stack pointer. For example, when block B5 is mapped to block B4, then, block B4 is removed from the stack as shown by the empty slot in Fig. 18.

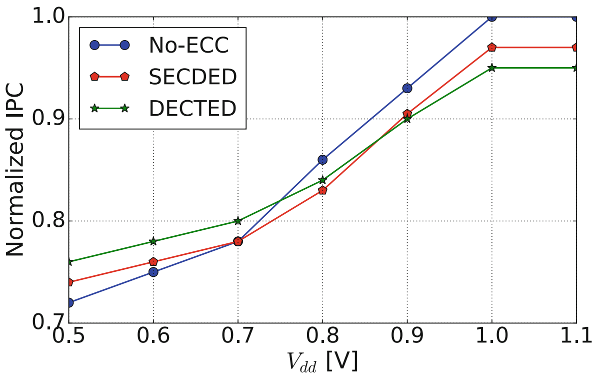
4.4 Evaluation of Voltage Scalable Mitigation Scheme

4.4.1 Variation-Aware Voltage Scaling Analysis

The supply voltage scalability of three different block sizes (16, 32, and 64 Byte) with different error correction schemes is compared in order to analyze the impact of block size selection on the supply voltage downscaling potential of cache memories with and without error correction schemes. The error-free (correctable error) minimum voltage of three block sizes is studied for 8 KB cache memory without ECC, parity, and Single Error Correction Double Error Detection (SECDED) configurations. Table 3 shows the supply voltage downscaling potential of the studied block sizes. For all ECC schemes (given in Table 3), the cache operating voltage has to be downscaled more conservatively when the block size is larger (64 Bytes). However, larger block sizes help to reduce the cache miss rate that results in a better cache performance. Therefore, for an aggressive supply voltage downscaling, the block size should be selected as small as possible by making performance and energy-saving trade-off analysis.



(a)



(b)

Fig. 19 Comparison of voltage downscaling in the presence of block disabling and ECC induce overheads for gzip, parser, and mcf applications from SPEC2000 benchmark (**a**) energy consumption comparison (**b**) Performance comparison in terms of IPC

4.4.2 Energy and Performance Evaluation of Voltage Scalable Cache Different ECC Schemes

The average energy reduction and performance comparison of voltage scaled cache memory with and without ECC are given in Fig. 19a and b by running selected workloads (*gzip*, *parser*, and *mcf*) from the SPEC2000 benchmark. The energy results in Fig. 19a are extracted from CACTI by considering block disabling, and ECC induced delay and energy overheads. As shown in the figure, supply voltage downscaling improves the energy efficiency significantly. However, the overheads of this scheme, namely ECC energy overhead, block disabling induced cash miss rate, and ECC encoding/decoding delay overhead outweigh the energy gain of supply voltage downscaling when the cache operating voltage is below 0.7 V. Therefore, the energy per access of Double Error Correction Triple Error Detection (DECTED) is higher than SECDED when the supply voltage is scaled down to 0.7 V or below. Similarly, Fig. 19b shows the cache performance (IPC) is reduced significantly with the supply voltage downscaling as more blocks are disabled for reliable operation.

5 Conclusion

Embedded microprocessors, particularly for battery-powered mobile applications, and energy-harvested Internet of Things (IoT) are expected to meet stringent energy budgets. In this regard, operating in the near-threshold voltage domain provides better performance and energy efficiency trade-offs. However, NTC faces various challenges among which increase in functional failure rate of memory components is the dominant issue. This chapter analyzed the combined effect of aging, process variation, and soft error on the reliability of cache memories in super and near-threshold voltage domains. It is observed that the combined effect of process variation and aging has a massive impact on the soft error rate and SNM degradation of NTC memories. Experimental results show process variation and aging-induced SNM degradation is $2.5\times$ higher in NTC than in the super-threshold voltage domain while SER is $8\times$ higher. The use of 8T instead of 6T SRAM cells reduces the system-level SNM and SER by 14% and 22%, respectively. Additionally, workload and cache organization have a significant impact on the FIT rate and SNM degradation of memory components. This chapter demonstrated that the reliability and performance optimal cache organization changes when going from the super-threshold voltage to the near-threshold voltage domain.

References

1. Agarwal, A., Paul, B.C., Mahmoodi, H., Datta, A., Roy, K.: A process-tolerant cache architecture for improved yield in nanoscale technologies. IEEE Trans. Very Large Scale Integr. Syst. **13**, 27–38 (2005)
2. Alameldeen, A.R., Wagner, I., Chishti, Z., Wu, W., Wilkerson, C., Lu, S.L.: Energy-efficient cache design using variable-strength error-correcting codes. In: ACM SIGARCH Computer Architecture News (2011)
3. Aly, R.E., Faisal, M.I., Bayoumi, M.A.: Novel 7T sram cell for low power cache design. In: Proceedings of the 2005 IEEE International SOC Conference (2005)
4. Amrouch, H., van Santen, V.M., Ebi, T., Wenzel, V., Henkel, J.: Towards interdependencies of aging mechanisms. In: Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (2014)
5. Au, A., Pogiel, A., Rajski, J., Sydow, P., Tyszer, J., Zawada, J.: Quality assurance in memory built-in self-test tools. In: 17th International Symposium on Design and Diagnostics of Electronic Circuits & Systems (2014)
6. Autran, J.L., Serre, S., Munteanu, D., Martinie, S., Semikh, S., Sauze, S., Uznanski, S., Gasiot, G., Roche, P.: Real-time soft-error testing of 40 nm SRAMS. In: Proceedings of IEEE International Reliability Physics Symposium (IRPS) (2012)
7. Binkert, N., Beckmann, B., Black, G., Reinhardt, S.K., Saidi, A., Basu, A., Hestness, J., Hower, D.R., Krishna, T., Sardashti, S., et al.: The gem5 simulator. ACM SIGARCH Comput. Archit. News **39**, 1–7 (2011)
8. Calhoun, B.H., Chandrakasan, A.: A 256kb sub-threshold SRAM in 65nm CMOS. In: 2006 IEEE International Solid State Circuits Conference-Digest of Technical Papers (2006)
9. Cazeaux, J.M., Rossi, D., Omana, M., Metra, C., Chatterjee, A.: On transistor level gate sizing for increased robustness to transient faults. In: 11th IEEE International On-Line Testing Symposium, IOLTS (2005)

10. Chang, L., Montoye, R.K., Nakamura, Y., Batson, K.A., Eickemeyer, R.J., Dennard, R.H., Haensch, W., Jamsek, D.: An 8T-SRAM for variability tolerance and low-voltage operation in high-performance caches. IEEE J. Solid-State Circ. **43**, 956–963 (2008)
11. Chatterjee, I., Narasimham, B., Mahatme, N., Bhuva, B., Reed, R., Schrimpf, R., Wang, J., Vedula, N., Bartz, B., Monzel, C.: Impact of technology scaling on SRAM soft error rates. IEEE Trans. Nuclear Sci. **61**, 3512–3518 (2014)
12. Chen, H., Manzi, D., Roy, S., Chakraborty, K.: Opportunistic turbo execution in NTC: exploiting the paradigm shift in performance bottlenecks. In: Proceedings of the 52nd Annual Design Automation Conference (2015)
13. Chen, Y.H., Chan, W.M., Wu, W.C., Liao, H.J., Pan, K.H., Liaw, J.J., Chung, T.H., Li, Q., Lin, C.Y., Chiang, M.C., et al.: A 16 nm 128 Mb SRAM in high-κ metal-gate FinFET technology with write-assist circuitry for low-VMIN applications. IEEE J. Solid-State Circuits **50**, 170–177 (2015)
14. Dreslinski, R., Wieckowski, M., Blaauw, D.S., Mudge, T.: Near threshold computing: Overcoming performance degradation from aggressive voltage scaling. In: Proceedings of the Workshop Energy-Efficient Design (2009)
15. Dreslinski, R.G., Wieckowski, M., Blaauw, D., Sylvester, D., Mudge, T.: Near-threshold computing: reclaiming Moore's law through energy efficient integrated circuits. Proc. IEEE **98**, 253–266 (2010)
16. Ebrahimi, M., Evans, A., Tahoori, M.B., Seyyedi, R., Costenaro, E., Alexandrescu, D.: Comprehensive analysis of alpha and neutron particle-induced soft errors in an embedded processor at nanoscales. In: Proceedings of the conference on Design, Automation & Test in Europe (2014)
17. Gebregiorgis, A., Tahoori, M.B.: Reliability and performance challenges of ultra-low voltage caches: A trade-off analysis. In: IEEE 24th International Symposium on On-Line Testing and Robust System Design (IOLTS) (2018)
18. Gebregiorgis, A., Ebrahimi, M., Kiamehr, S., Oboril, F., Hamdioui, S., Tahoori, M.B.: Aging mitigation in memory arrays using self-controlled bit-flipping technique. In: 20th Asia and South Pacific Design Automation Conference (ASP-DAC) (2015)
19. Gebregiorgis, A., Kiamehr, S., Oboril, F., Bishnoi, R., Tahoori, M.B.: A cross-layer analysis of soft error, aging and process variation in near threshold computing. In: Design, Automation & Test in Europe Conference & Exhibition (DATE) (2016)
20. Gebregiorgis, A., Bishnoi, R., Tahoori, M.B.: A comprehensive reliability analysis framework for NTC caches: a system to device approach. IEEE Trans. Comput. Aided Design Integr. Circuits Syst. **38**, 439–452 (2018)
21. Grossar, E., Stucchi, M., Maex, K., Dehraene, W.: Read stability and write-ability analysis of SRAM cells for nanometer technologies. IEEE J. Solid-State Circuits **41**, 2577–2588 (2006)
22. Hamdioui, S., Al-Ars, Z., Gaydadjiev, G.N., Van de Goor, A.: Generic march element based memory built-in self test. US Patent 8,910,001, 2014
23. Hazucha, P., Svensson, C.: Impact of CMOS technology scaling on the atmospheric neutron soft error rate. IEEE Trans. Nuclear Sci. **47**, 2586–2594 (2000)
24. Henkel, J., Bauer, L., Dutt, N., Gupta, P., Nassif, S., Shafique, M., Tahoori, M., Wehn, N.: Reliable on-chip systems in the nano-era: Lessons learnt and future trends. In: Proceedings of the 50th Annual Design Automation Conference (2013)
25. Henning, J.L.: SPEC CPU2000: Measuring CPU performance in the new millennium. Computer **33**, 28–35 (2000)

26. Hubert, G., Artola, L., Regis, D.: Impact of scaling on the soft error sensitivity of bulk, FDSOI and FinFET technologies due to atmospheric radiation. Integration **50**, 39–47 (2015)
27. Jahinuzzaman, S.M., Sharifkhani, M., Sachdev, M.: An analytical model for soft error critical charge of nanometric SRAMs. IEEE Trans. Very Large Scale Integr. Syst. **17**, 1187–1195 (2009)
28. Jeppson, K.O., Svensson, C.M.: Negative bias stress of MOS devices at high electric fields and degradation of MNOS devices. J. Appl. Phys. **48**, 2004–2014 (1977)
29. Jiang, C., Zhang, D., Zhang, S., Wang, H., Zhuang, Z., Yang, F.: A yield-driven near-threshold 8-T SRAM design with transient negative bit-line scheme. In: 7th IEEE International Conference on Electronics Information and Emergency Communication (ICEIEC) (2017)
30. Kuhn, K.J., Giles, M.D., Becher, D., Kolar, P., Kornfeld, A., Kotlyar, R., Ma, S.T., Maheshwari, A., Mudanai, S.: Process technology variation. IEEE Trans. Electr. Devices **58**, 2197–2208 (2011)
31. Maric, B., Abella, J., Valero, M.: Adam: An efficient data management mechanism for hybrid high and ultra-low voltage operation caches. In: Proceedings of the Great Lakes Symposium on VLSI (2012)
32. Morita, Y., Fujiwara, H., Noguchi, H., Iguchi, Y., Nii, K., Kawaguchi, H., Yoshimoto, M.: An area-conscious low-voltage-oriented 8T-SRAM design under DVS environment. In: IEEE Symposium on VLSI Circuits (2007)
33. Mukhopadhyay, S., Mahmoodi, H., Roy, K.: Modeling of failure probability and statistical design of SRAM array for yield enhancement in nanoscaled CMOS. IEEE Trans. Comput. Aided Design Integr. Circuits Syst. **24**, 1859–1880 (2005)
34. Olorode, O., Nourani, M.: Improving performance in sub-block caches with optimized replacement policies. ACM J. Emerg. Technolo. Comput. Syst. **11**, 1–22 (2015)
35. Seok, M., Chen, G., Hanson, S., Wieckowski, M., Blaauw, D., Sylvester, D.: CAS-FEST 2010: Mitigating variability in near-threshold computing. IEEE J. Emer. Sel. Topics Circuits Syst. **1**, 42–49 (2011)
36. Shivakumar, P., Jouppi, N.P.: Cacti 3.0: An integrated cache timing, power, and area model. Technical Report, Compaq Computer Corporation (2001)
37. Siddiqua, T., Gurumurthi, S., Stan, M.R.: Modeling and analyzing NBTI in the presence of process variation. In: 12th International Symposium on Quality Electronic Design (ISQED) (2011)
38. Sridharan, V., Kaeli, D.R.: Using hardware vulnerability factors to enhance AVF analysis. ACM SIGARCH Comput. Archit. News **38**, 461–472 (2010)
39. Takeda, K., Hagihara, Y., Aimoto, Y., Nomura, M., Nakazawa, Y., Ishii, T., Kobatake, H.: A read-static-noise-margin-free SRAM cell for low-VDD and high-speed applications. IEEE J. Solid-State Circuits **41**, 113–121 (2006)
40. Tonfat, J., Azambuja, J.R., Nazar, G., Rech, P., Frost, C., Kastensmidt, F.L., Carro, L., Reis, R., Benfica, J., Vargas, F., et al.: Analyzing the influence of voltage scaling for soft errors in SRAM-based FPGAs. In: 14th European Conference on Radiation and Its Effects on Components and Systems (RADECS) (2013)
41. Understanding CPU caching and performance (2015). http://http://arstechnica.com/gadgets/2002/07/caching/2/

42. Wilkening, M., Sridharan, V., Li, S., Previlon, F., Gurumurthi, S., Kaeli, D.R.: Calculating architectural vulnerability factors for spatial multi-bit transient faults. In: Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture (2014)
43. Wilkerson, C., Gao, H., Alameldeen, A.R., Chishti, Z., Khellah, M., Lu, S.L.: Trading off cache capacity for reliability to enable low voltage operation. In: ACM SIGARCH Computer Architecture News (2008)

Part IV
Cross-Layer from Physics to Gate- and Circuit- Levels

Mehdi Tahoori

The rapid shrinking of device geometries in the nanometer regime has led to the need for new system-on-chip (SoC) design methodologies at various levels of abstraction. Improvements in chip manufacturing technology and system integration have propelled an astonishing growth of computing systems which are integrated into almost all aspects of our daily lives. However, this trend is facing serious challenges, both at device and system levels. At the device level, as the minimum feature size continues to shrink, a host of vulnerabilities influence the robustness and reliability of computing systems. Some of these factors are caused by the stochastic nature of the nanoscale manufacturing process (e.g., process variability, sub-wavelength lithographic inaccuracies), while other factors appear because of high operating frequencies and intrinsic nanoscale features (e.g., RLC noise, on-chip temperature variation, increased sensitivity to radiation and device aging). Therefore, the reliability of systems on chip is not only limited to the technology parameters and hardware design, it is highly influenced by the runtime environment and executed workload, which can aggravate hardware stress and cause failures.

The chapters presented in this section are looking at the reliability issues from the circuit, logic, and physical design perspective, while linking the issues to the technology from one side and higher abstraction level (architecture, system, and workload) from the other end, therefore they represent a cross-layer angle on these reliability challenges.

The first chapter in this section addressed transistor aging in system bistables, mainly flip-flops. It presents various methods to improve the reliability of gate-level digital circuits by addressing the timing degradation of flip-flops under severe aging and voltage-drop. This is achieved through selective flip-flop optimization. The idea presented in this chapter is to find timing-critical flip-flops under high aging and/or voltage-drop impact, and selectively re-optimize them for operating under such

M. Tahoori
Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany
e-mail: mehdi.tahoori@kit.edu

stress by appropriately sizing their transistors. Therefore, instead of replacing all flip-flops in the netlist with the hardened versioned, only the ones which are subject to extreme stress are replaced by hardened version. This is achieved by a cross-layer analysis which based on the architecture and analyzing the running workload, the flip-flops under severe aging and voltage fluctuation stress are identified. This effectively improves the reliability and lifetime of circuits without imposing much overhead, because these flip-flops constitute a small portion of all flip-flops.

The second chapter in this section focuses on electromigration (EM) which is the major interconnect aging effect in advanced technology nodes. It provides techniques for power grid network sizing while considering electromigration reliability. This chapter starts with an overview power grid network and electromigration fundamentals. The main issue addressed in this chapter is EM immortality and aging effects, used as EM constraints when formulating the optimization problems. When an interconnect line is below a critical stress, the void nucleation cannot occur and hence the metal wires become immortal and will not fail. The chapter first shows that the new Power/ground (P/G) optimization problem, subject to the voltage IR drop and new EM constraints, can still be formulated as an efficient sequence of linear programming (SLP) problem. The new optimization will ensure that none of the wires fail if all the constraints are satisfied. However, requiring all the wires to be EM immortal can be over-constrained. To mitigate this problem, the improvement is to consider the aging effects of interconnect wires in P/G networks.

The third chapter is devoted to various monitoring circuitry for improving cross-layer resiliency. The role of monitor circuits is to establish a bridge between the hardware and other layers by providing information about the devices and the operating environment in runtime. This chapter explores delay-based monitor circuits for design automation with the existing cell-based design methodology. The chapter reviews several design techniques to monitor parameters of threshold voltage, temperature, leakage current, critical delay, and aging. The chapter then demonstrates a reconfigurable architecture to monitor multiple parameters with small area footprint. Finally, an extraction methodology of physical parameters is discussed for model-hardware correlation.

The last chapter discusses yield and aging in scaled technologies. For the robustness of VLSI design methodology and cycles, reliability and yield need to be accurately modeled, systematically optimized, and seamlessly integrated into the existing design flow. This chapter will survey critical aging and yield issues, and then review the state-of-the-art techniques to tackle them, including both modeling and optimization strategies which reside across the Physics and Circuit/Gate layers as part of the overall dependability scheme. The strategies often involve synergistic cross-layer optimization due to the complicated VLSI design procedures nowadays. Novel modeling techniques leveraging machine learning are analyzed along with analytical optimization approaches.

Selective Flip-Flop Optimization for Circuit Reliability

Mohammad Saber Golanbari, Mojtaba Ebrahimi, Saman Kiamehr, and Mehdi B. Tahoori

1 Introduction, Motivation, and Contributions

VLSI circuits are influenced by several sources of process and runtime variabilities [16]. Among them, supply voltage fluctuation and transistor aging due to BTI are the most important factors [2, 30, 36]. They degrade the performance of VLSI circuits by increasing the delay, and consequently deteriorate lifetime.

The impacts of both voltage-drop and aging are significant on sequential elements such as flip-flops and latches. Due to particular aspects of flip-flops, such as the internal feedback structure, degradation of the transistors of a flip-flop as well as supply voltage fluctuation may lead to serious timing degradation or even functional failure (inability to capture the input independent of timing) [24]. Furthermore, many flip-flops are on the critical paths of a circuit because logic synthesis tools balance the delays of circuit paths to achieve the best performance, area, and power. Therefore, it is necessary to employ design-time mitigation techniques to consider and control such gradual degradation, e.g. by adding appropriate timing margins (*aging and voltage-drop guardband*) [20, 28].

Our analysis shows that in a typical digital design such as a microprocessor, based on the functionality of different components, some flip-flops operate under static or near-static BTI stress, irrespective of the workload. These flip-flops experience large timing degradation because the flip-flop input Signal Probability (SP) is very close to 0.0 or 1.0. Being subject to severe BTI stress, the aforementioned flip-flops degrade faster, imposing a large *aging guardband* to the entire circuit. Flip-flops also experience a large temporally localized voltage-drop, because they are synchronized with the clock edge and supposedly operate at the same time (at

M. S. Golanbari · M. Ebrahimi · S. Kiamehr · M. B. Tahoori (✉)
Karlsruhe Institute of Technology, Karlsruhe, Germany
e-mail: golanbari@kit.edu; mehdi.tahoori@kit.edu

J. Henkel, N. Dutt (eds.), *Dependable Embedded Systems*, Embedded Systems,
https://doi.org/10.1007/978-3-030-52017-5_14

clock edge), hence, drawing substantial current leading to a significant voltage-drop over Power Delivery Network (PDN) [22]. Moreover, recent studies have shown that the voltage-drop impact gets more severe by technology scaling [2, 21, 38]. Therefore, in a conventional design flow, costly *voltage-drop timing guardband* is considered for reliable circuit operation [22].

In this chapter, we explore methods to improve circuit reliability by addressing the timing degradation of flip-flops under severe aging[1] and voltage-drop, i.e. *selective flip-flop optimization*. The idea is to find timing-critical flip-flops under high aging and/or voltage-drop impact, and selectively re-optimize them for operating under such stress by appropriately sizing their transistors. This effectively improves the reliability and lifetime of circuits without imposing much overhead, because these flip-flops constitute a small portion of all flip-flops.

Simulation results obtained by applying the proposed method to a processor show that the flip-flops optimized with the proposed method exhibit much less delay degradation, while imposing less than 0.1% leakage power overhead to the processor. As a result, the required timing guardband of the processor using the proposed method is significantly less compared to the original processor. Therefore, given a specific clock period, the optimized processor design with the proposed method has 36.9% longer lifetime and better reliability compared to the original processor design.

2 Variability Impact on Flip-Flops

2.1 Flip-Flop Timing

Flip-flop timing metrics such as setup-time (U), hold-time (H), clock-to-q (D_{CQ}), and data-to-q (D_{DQ}) are well discussed in [31, 34]. When the setup-time is large enough, the clock-to-q value is almost constant, but further reduction of the setup-time will increase the clock-to-q value monotonously until a value after which the flip-flop is unable to capture and latch the input [31]. Based on this, the *optimum setup-time* is defined as the setup-time value which causes the clock-to-q value to increase by 10% from its minimum value [32]. Moreover, each flip-flop has two internal paths; one for transferring the input state “zero” to the output i.e. High-to-Low (HL) input transition, and the other for transferring the input state “one” to the output i.e. Low-to-High (LH) input transition. Basically, the timing parameters for these two internal paths can be different [24] as shown in Fig. 1, meaning that there are two sets of timing parameters for internal LH and HL paths of a flip-flop:

$$\{U_{LH}, D_{CQ_{LH}}, D_{DQ_{LH}}\} \qquad \text{for LH transition,}$$

[1]We consider the impact of Negative Bias Temperature Instability (NBTI) on PMOS transistors, and Positive Bias Temperature Instability (PBTI) on NMOS transistors.

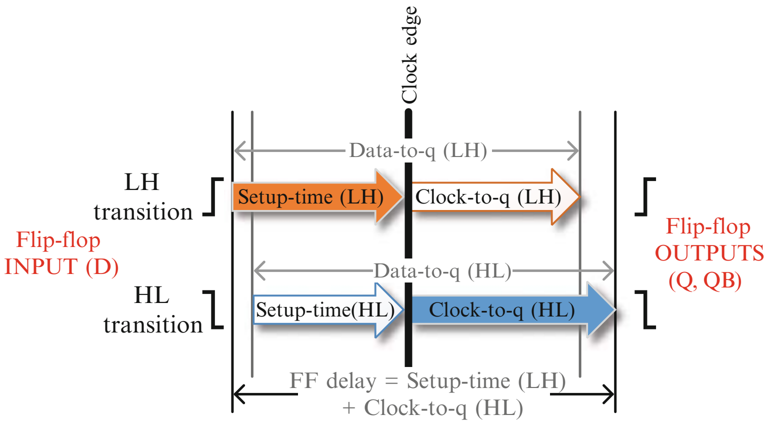


Fig. 1 Different flip-flop timing parameters. The correct functionality is guaranteed by considering the flip-flop delay as illustrated

$$\{U_{HL}, D_{CQ_{HL}}, D_{DQ_{HL}}\} \qquad \text{for HL transition.}$$

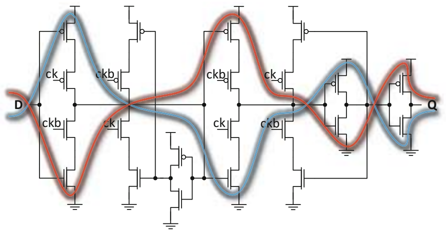
Flip-flop delay should be defined such that the correct functionality of the flip-flop will be guaranteed, disregard of the transition. Therefore, we define the flip-flop delay as the *summation of the worst setup-time and the worst clock-to-q of both transitions* as shown in Fig. 1.

$$\text{delay} = \max\{U_{LH}, U_{HL}\} + \max\{D_{CQ_{LH}}, D_{CQ_{HL}}\}. \tag{1}$$

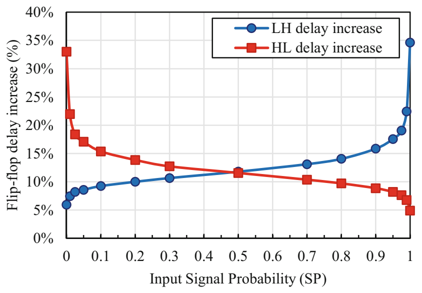
This guarantees that in both transitions the input signal is correctly captured and propagated to the flip-flop output.

2.2 Runtime Variation Impacts on Flip-Flops

Several parameters such as supply voltage, workload, and temperature affect the performance of flip-flops in a circuit. Parameters such as temperature and supply voltage affect all the transistors of a flip-flop in the same way, whereas the impact of the input SP is different for the transistors of a flip-flop [23]. This results in an asymmetric aging of transistors according to their stress duty cycles. Therefore, the delay degradation of internal LH and HL paths inside an aged flip-flop depends on the input SP [24].



(a)



(b)

Fig. 2 Separate internal LH (red)/HL (blue) paths of a C2MOS flip-flop [31] (**a**), and delay of internal LH/HL paths of an aged C2MOS flip-flop after 5 years (optimized for Power Delay Product (PDP) in the fresh state) for different input SPs (**b**)

In the C2MOS flip-flop[2] depicted in Fig. 2a, the internal LH and HL paths consist of two separate groups of transistors, which makes the aging of these two paths independent according to the input SP. Figure 2b illustrates the delay of LH and HL transitions of an aged C2MOS flip-flop [31] for different input SPs. When the flip-flop is aged under input SP $= 0.0$ (SP0), the worst delay degradation happens on the flip-flop HL path; however, the delay of the flip-flop LH path is only slightly affected. On the other hand, an aging under input SP $= 1.0$ (SP1) greatly degrades the delay of the flip-flop LH path while slightly affecting the delay of the flip-flop HL path. For moderate aging condition, i.e. $0.1 < \text{SP} < 0.9$, the delay degradation of both LH and HL paths is moderate. The reason is that under SP0 and SP1 conditions, *Static BTI* (S-BTI) asymmetrically alters the threshold voltages leading to unbalanced aging of LH and HL paths of the flip-flop as the stress duty cycle of some transistors is 1.0, i.e., always under BTI stress. However, in moderate aging condition, the transistors can partially recover as the stress duty cycle is less than 1.0.

The impact of supply voltage fluctuation on the flip-flops of a circuit depends on the workload variation and dynamic power consumption of the circuit. Therefore, each flip-flop may experience a specific amount of voltage-drop. A voltage-drop causes performance degradation of the flip-flops, which is typically larger than the degradation of simpler combinational gates in the standard cell library. Figure 3 compares the impact of a voltage-drop up to 10% on the delay of an aged flip-flop and an aged inverter. Compared to a no-voltage-drop condition, the delay of the flip-flop increases by 23.6% whereas the delay of the inverter is increased by 15%.

Moreover, the flip-flops of a circuit generally experience higher amount of voltage-drop compared to combinational gates [37]. As a result of temporally

[2]A C2MOS flip-flop design is a master–slave flip-flop built of two connected C2MOS latches. It is one of the commonly used flip-flops in modern processor designs [31].

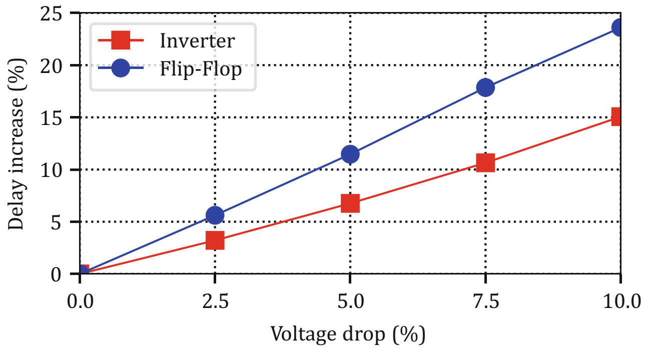
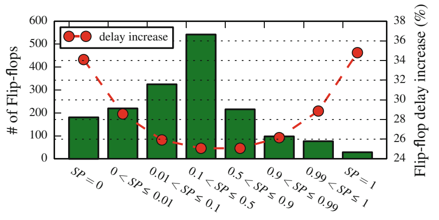


Fig. 3 Comparison between the voltage-drop induced delay degradation of a flip-flop and an inverter, which are aged under same condition (Aging under SP1 for 5 years)

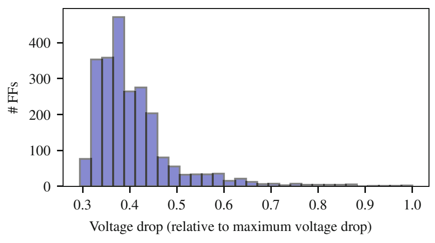
localized switching of flip-flops at the positive (or negative) edge of clock signal, the instantaneous current drawn from PDN at the synchronized clock edge is comparatively high. This leads to high voltage-drop at the clock edge, when the flip-flops are processing their input signals. This peak current consumption is damped over the rest of the clock period, when the combinational cells are active. Therefore, in this work we focus on dealing with the impact of voltage fluctuation on the flip-flops.

Temporal and spatial temperature variations can also affect the circuit performance. The temporal temperature change could be rather high and has been the subject of research since it affects the reliability of the VLSI circuits. It is demonstrated in [17] that the circuit performance can be changed by up to 10% for 110 °C temperature variation. Therefore, in order to meet the reliability constraints, the circuit timing should be adjusted according to the worst temperature corner, which is typically at high temperature. On-chip spatial temperature gradient puts different stress on circuit components across a chip. The amount of on-chip spatial temperature difference (only on cores) based on simulation [3, 7], sensor measurements [33], and thermal camera [3] is reported to be up to ∼30°C. Since the delay change is approximately 4% for every 40 °C [17, 29], the overall difference between the delay degradation of core flip-flops due to such spatial temperature gradient is expected to be less than 3%, and hence, much smaller compared to voltage-drop variation [11].

The combined impact of voltage-drop and aging significantly degrades the performance of flip-flops. As an example, the delay of a fresh flip-flop optimized with balanced HL/LH delay increases from 98.5 ps to 165.7 ps due to the combined impact of voltage-drop (10%) and S-BTI (5-years under SP0). This is equivalent to 68% delay increase. If such a flip-flop is in a critical path of the circuit, a large timing guardband is required for timing closure considering the reliability constraints. Therefore, it is necessary to find such flip-flops at design-time and optimize them for operating under such conditions.



(a) The average flip-flop SPs during the execution of some MiBench workloads on Leon3, and the corresponding delay degradation in 5 years. In order to be fair in the analysis, the flip-flops which are not exercised by the workloads are excluded from this analysis.



(b) The distribution of voltage-drop on the Leon3 flip-flops. The voltage-drop values are normalized to the maximum voltage-drop across the processor.

Fig. 4 (**a**) Input signal probabilities and (**b**) voltage-drop analysis of Leon3 flip-flops executing MiBench Workloads

2.3 Significance of Flip-Flops in Circuit Reliability

In a properly designed circuit, the timing of circuit paths are balanced during the synthesis process. Therefore, many flip-flops are *timing-critical* as they lie on the circuit critical paths. Studies [12, 37] have shown that in VLSI circuits, some flip-flops are under severe static BTI leading to a large timing degradation over time. Furthermore, the impact of voltage-drop on flip-flops could be very high as a result of localized power consumption at a specific time (e.g. positive clock edge) or at a specific location on the circuit layout.

The large impact of S-BTI and voltage-drop on flip-flops has a significant impact on the reliability of a circuit when such flip-flops are timing-critical. In order to investigate the likelihood of having such a scenario in a typical digital design, we use the flow presented in Sect. 4 to extract the voltage-drop and the aging of the Leon3 flip-flops by executing six MiBench workloads [15] namely stringsearch, qsort, basicmath, bitcount, fft, and crc32 on Leon3 processor [10]. In order to be fair, we excluded the flip-flops belonging to the parts which are not exercised by the employed workloads such as interrupt handler, timers, and UART controller. The synthesized netlist of the Leon3 processor has 2352 flip-flops, but the results demonstrated in this section contain only 1686 flip-flops belonging to the parts which are exercised by all employed workloads.

Figure 4a demonstrates the input SP distribution of the aforementioned 1686 flip-flops. The results show that 181 flip-flops always experience input SP0, whereas 29 flip-flops are under input SP1. Our analysis shows that the flip-flops with such behavior typically belong to either the error checking and exception handling registers or higher bits of address registers which are constant due to temporal and spatial locality of the executed instructions. Besides, the SP of a considerable number of flip-flops is very close to either 0.0 or 1.0. Please note that the results reported in Fig. 4a are the average of six employed workloads, and hence, the flip-flops with SP = 0 or SP = 1 have such SP across all executed workloads. Similar experiment has been carried out in [18] to study the impact of workload in real systems, which shows that some flip-flops are always under S-BTI across different workloads.

Figure 4b shows the distribution of the maximum voltage-drop impacting the flip-flops of Leon3 processor compared to the peak voltage-drop across all the executed workloads. Please note that it is necessary to consider the maximum voltage-drop over the execution of all workloads, because it eventually impacts the flip-flop characteristics. A significant portion of flip-flops experience on average 41% of the maximum amount of voltage-drop; however, there are flip-flops at the right side tail of the distribution which experience large voltage-drop comparable to the maximum voltage-drop in the circuit.

According to the observations in Fig. 4, there are flip-flops experiencing both S-BTI and high voltage-drop which leads to high-degradation. If such flip-flops are on a critical path of the processor (i.e. timing-critical flip-flops), the degradation of the flip-flops should be reflected in the timing guardband of the circuit. Timing-critical flip-flops can be categorized into different groups based on the impact of voltage-drop and aging as follows:

- low voltage-drop and low aging,
- low voltage-drop but S-BTI aging (SP0/SP1)*
- high voltage-drop but typical aging*
- high voltage-drop and S-BTI aging (SP0/SP1)*

Therefore, we propose to generate flip-flops specifically optimized for such high-degradation conditions (marked by *) and add them to the standard cell library. Using the proposed flow in Sect. 4, we determine such high-degradation and timing-critical flip-flops and replace them with the optimized versions to improve the timing and reliability of the circuit.

3 Reliability-Aware Flip-Flop Design

In a typical reliability-aware circuit design, one should consider the delay of the elements under variation impacts to ensure the correct functionality of the circuit during the expected lifetime. Therefore, higher delay degradation of timing-critical flip-flops imposes a large timing guardband. In our proposed methodology, we create optimized versions of the flip-flops for different stress conditions based on aging and voltage fluctuations, and use these optimized versions only when a flip-flop is timing-critical and subject to such stress conditions to avoid unnecessary over design. This means that in the cell library, we add the following resilient versions of the flip-flops:

- Aging-resilient flip-flops, optimized for different aging corners (SP0 and SP1),

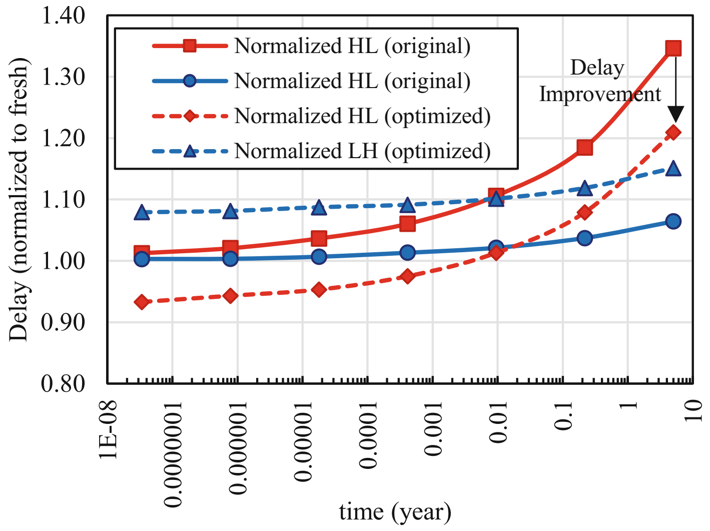


Fig. 5 Delay of a C2MOS flip-flop which is aged under SP = 0 over 5 years for LH/HL transitions, compared to the flip-flop optimized for SP = 0 showing how the unbalanced aging of internal LH/HL paths worsens the degradation in original flip-flop

- Voltage-drop resilient flip-flops, optimized to have lower performance degradation under voltage fluctuation,
- Aging and voltage-drop resilient flip-flops.

3.1 Aging-Resilient Flip-Flop Design

When the fresh delays of internal paths of a flip-flop (i.e., LH and HL paths) are designed to be similar (depicted as solid lines in Fig. 5), the internal path with higher degradation rate eventually becomes dominant and determines the total delay of the flip-flop. In this case, a significant aging in flip-flop characteristics is observed over time (corresponding to the internal path with higher degradation). On the other hand, if the internal path with higher degradation rate is initially faster (by design) than the internal path with lower degradation rate, the dominant internal path would be the slower one, and hence the higher degradation rate of the faster internal path is masked. Consequently, the overall aging of the flip-flop would be rather small. The delay of the optimized flip-flop, shown in Fig. 5 by dashed lines, exhibits such characteristics. The post-aging delay of the optimized flip-flop would increase by ∼10 ps, which is much lower than ∼40 ps increase in the delay of the original flip-flop. We exploit this method for designing aging-resilient flip-flops.

In order to decrease the overall BTI-induced aging inflicted to a flip-flop, our proposed method balances the delay of internal HL and LH paths of the flip-flop for *post-aging state of the flip-flop*, by resizing the transistors of internal HL and LH paths. In other words, the proposed method increases the fresh delay ($t = 0$) of the flip-flop internal path which has lower degradation rate in order to compensate the overall degradation of the flip-flop after aging. Although the fresh delay of the optimized flip-flop might be slightly larger compared to the fresh delay of the original flip-flop, the overall delay of the optimized flip-flop considering the aging-induced timing margin would be smaller than those of the original flip-flop since the aging rate is much smaller.

Please note that this method reduces the degradation for a given SP, but inevitably worsens the aging at the other corners of SP. For example, if we optimize the flip-flop for SP0, the degradation would be much higher if the optimized flip-flop operates at SP1. Nevertheless, these flip-flops under S-BTI will not operate at other SP corners, because their SP is determined by the circuit structure and functionality. Therefore, we only optimize for the given SP corner. This means that we intentionally sacrifice other corners, which never occur due to the specific functionality of the circuit, to gain a larger improvement.

3.2 Voltage-Drop Resilient Flip-Flop Design

Other than aging, which affects each flip-flop transistor based on the input signal probability, a drop in the supply voltage of the flip-flop slows down all flip-flop transistors in the same way. However, a slight upsizing of specific transistors can compensate the degradation in the flip-flop timing. Therefore, we evaluate the delay of the flip-flop when operating under the impact of voltage-drop, and optimize the flip-flop with the goal of improving the delay. Consequently, the optimized flip-flop would have better timing at the cost of higher power consumption.

3.3 Aging and Voltage-Drop Resilient Flip-Flop Design

The degradation in the flip-flop timing due to both S-BTI and voltage-drop is very large. Such timing degradation may not be effectively compensated by resizing the transistors within a flip-flop area without upsizing the entire flip-flop. Therefore, in addition to targeting for better timing under the impact of the aging and voltage-drop, we allow the optimization algorithm to increase the area of the flip-flop by a small percentage. Please note that an extra Engineering Change Order (ECO) might be needed to replace the original flip-flop with the optimized version in this case. However, since there exist only a few flip-flops under such degradation it would not be an issue to perform an ECO on placement.

3.4 Problem Formulation for Flip-Flop Resiliency Optimization

The delay of a flip-flop under a specific working condition (including temperature, voltage, and input SP) can be presented as a function of the transistors' widths:

$$\text{delay} = f(W), \quad W = [w_i], \tag{2}$$

where $[w_i]$ is a vector containing the width of flip-flop transistors. Here, $delay$ is the delay (Data-to-q) of the flip-flop, according to Eq. (1), under variation impact, which could be S-BTI stress, voltage-drop, or both depending on the optimization approach.

The delay function f is a complicated function of transistors' widths. Our experimental results for flip-flops with different sizing show that f cannot be presented with any general linear function. Therefore, we use Sequential Quadratic Programming (SQP) which is a non-linear programming technique [19]. In SQP, the problem is converted into quadratic sub-problems and solved in order to find a better sizing in each iteration. For this purpose, we follow an iterative approach in order to minimize the delay of Eq. (2). Given an initial sizing, the delay function f is approximated with a quadratic function:

$$f(W) \approx f(W_0) + \triangledown f(W_0)^T \cdot (W - W_0) + \frac{1}{2}(W - W_0)^T \cdot H_f(W_0) \cdot (W - W_0), \tag{3}$$

where $\triangledown f(W)$ and $H_f(W)$ are the gradient and the Hessian of the delay function f, respectively. Minimizing the quadratic approximation of Eq. (3), with respect to some constraints, which will be discussed later in this section, yields an optimized transistor sizing. Thereafter, the obtained sizing is used as the initial sizing, and a new iteration is launched. This cycle continues until the optimization reaches the required precision, i.e. the difference between the optimized delays of two consecutive iterations becomes smaller than a predefined threshold ϵ_{delay}. Therefore, the solver continues by checking the precision of the resulting delay:

$$\left|\text{delay}_{i-1} - \text{delay}_i\right| < \epsilon_{\text{delay}} \tag{4}$$

where delay_i represents the delay of ith iteration.

Another reason to use the quadratic approximation is that the optimum result of a linear problem always lies on the boundaries, while the optimum result of a

Table 1 Flip-flop optimization method summary

Parameters	$W = (w_1, \ldots, w_n)$	
Initial guess	$W_0 =$ optimized W for PDP (fresh)	
Constraints	$w_i \geq w_{\min}$	
	$\sum_1^n w_i \leq (1 + \lambda) \sum W_0$	
	power$(W) \leq (1 + \beta)$ power(W_0)	
Target	minimize: delay $= f(W)$	
Constants	$w_{\min}$	Minimum size
	λ	Acceptable excessive area
	β	Acceptable excessive leakage

quadratic problem can be any point within the boundaries as well as the boundaries themselves. In Sect. 5, we demonstrate that the optimum result does not necessarily lie on the boundaries, and hence a non-linear programming technique is needed to find a better result. Table 1 summarizes the optimization problem.

Several constraints are applied to the optimization problem, relating to transistors size, flip-flop area, and leakage. The first constraint shown in Table 1 limits the minimum size of transistors. The second constraint limits the area of the optimized flip-flop. In case of optimizing for S-BTI or voltage-drop, we consider $\lambda = 0$ to keep the flip-flop area within the area of the original flip-flop which also facilitates keeping the aspect ratio almost equal to the aspect ratio of the original flip-flop. This way, the optimized flip-flop can easily replace the original flip-flop without any layout modifications at the circuit-level. This is achieved by limiting the summation of transistor widths w_i. However, for flip-flops under S-BTI and voltage-drop, we assume a $\lambda > 0$ value to compensate the delay degradation better. The third constraint sets an upper limit for the excessive leakage of the flip-flop by parameter β. This constraint is applied to the optimization problem to limit the leakage power of the optimized flip-flops within an acceptable range. The initial guess of optimization W_0 is the optimum sizing for minimum PDP in the fresh state.

3.5 Reliability-Aware Flip-Flop Optimization Flow

Figure 6 presents our proposed reliability-aware optimization flow. For a given input SP, the SP of all transistors are once calculated using SPICE simulations. Afterwards, based on the extracted SP for transistors and the operating corner of the flip-flop (temperature, supply voltage, etc.), the BTI-induced threshold voltage shifts of all transistors (ΔV_{th}) are obtained. Then, the ΔV_{th} values are back-annotated into the original flip-flop SPICE netlist, and the SPICE netlist of aged flip-flop is generated.

In each SQP iteration, the quadratic sub-problems are created and solved to generate further improved flip-flop sizing. Subsequently, the new sizing is back-annotated into the aged flip-flop netlist extracted before, and a new aged flip-flop with the given sizing is generated. Then, Cadence Virtuoso Liberate [6] is used

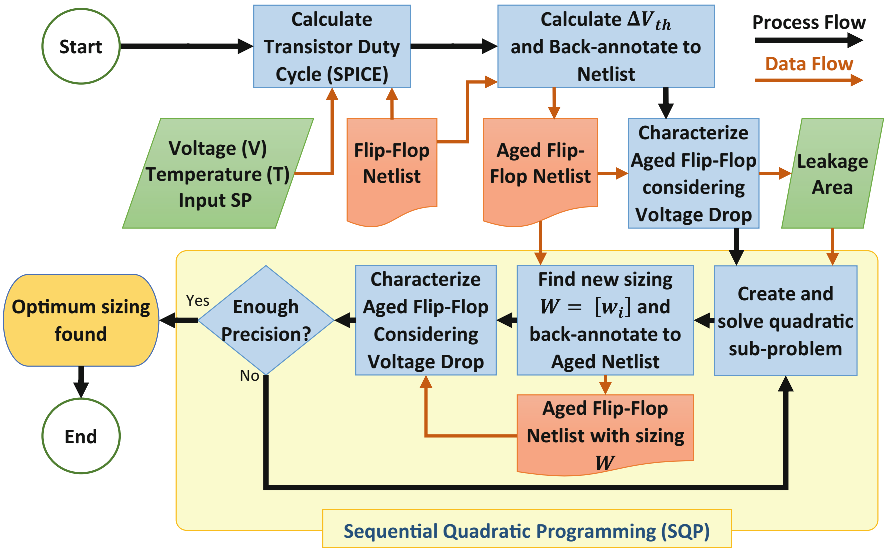

Start
Calculate Transistor Duty Cycle (SPICE)
Calculate ΔV_{th} and Back-annotate to Netlist
Process Flow
Data Flow
Voltage (V) Temperature (T) Input SP
Flip-Flop Netlist
Aged Flip-Flop Netlist
Characterize Aged Flip-Flop considering Voltage Drop
Leakage Area
Optimum sizing found
Yes
Enough Precision?
No
Characterize Aged Flip-Flop Considering Voltage Drop
Find new sizing $W = [w_i]$ and back-annotate to Aged Netlist
Create and solve quadratic sub-problem
Aged Flip-Flop Netlist with sizing W
End
Sequential Quadratic Programming (SQP)

Fig. 6 Overall flow to find the optimum flip-flop sizing for under S-BTI stress and voltage-drop at a specific working corner (voltage, temperature)

to characterize the new flip-flop and extract its delay and power consumption. When the improvement is small enough and the condition in Eq. (4) is met, the SQP method terminates and returns the last sizing as the optimum solution for the problem.

As the process is executed at a specific supply voltage (V_{dd}), it can inherently be used to optimize for a voltage-drop as well, when the given supply voltage includes the impact of the voltage-drop. We can also create voltage-drop resilient version of a flip-flop for typical aging, by considering input SP of 0.5. Therefore, we execute the flow presented in Fig. 6 for these conditions in order to create variation-resilient versions of the flip-flop, assuming a supply voltage of V_{dd} and a maximum voltage-drop of $R\%$:

	Supply voltage (V)	Aging condition
Aging	V_{dd}	S-BTI (SP0, SP1)
Voltage-drop	$(1 - \frac{R}{100})V_{dd}$	Typical aging (SP = 0.5)
Aging and voltage-drop	$(1 - \frac{R}{100})V_{dd}$	S-BTI (SP0, SP1)

After optimization process, it is necessary to re-characterize the flip-flops for different supply voltage ($(1 - \frac{R}{100})V_{dd}$ to V_{dd}) and aging conditions ($0 \leq \text{SP} \leq 1$). The characterization results are then used to obtain overall circuit timing under supply voltage fluctuation and aging impacts.

4 Selective Flip-Flop Optimization

This section explains how the optimized flip-flops in Sect. 3 can be employed to improve the reliability of a circuit. The idea is to find the flip-flops affected by S-BTI and/or voltage-drop impacts which are also influential on circuit reliability, i.e. the timing-critical flip-flops, and replace them with the optimized versions. The reason for this *selective flip-flop optimization* is that the reliability-aware flip-flop optimization is costly in terms of leakage overhead per flip-flop. Therefore, flip-flop replacement should be done only for the timing-critical flip-flops which experience S-BTI and/or large voltage-drop to be cost-effective. Since they constitute a small subset of the all flip-flops in the design, the proposed method is able to reduce the overall timing guardband in a cost-effective way.

The overall flow of the proposed selective flip-flop optimization methodology is presented in Fig. 7. The flow uses the results of the Synthesis and Place & Route steps of a VLSI design flow and is composed of *(I) Aging and Voltage-Drop Analysis* and *(II) Selective Flip-flop Replacement* steps. The optimization flow updates the gate-level netlist and the circuit layout to improve the reliability of the circuit under voltage-drop and aging impacts. The outputs of the optimization method can be further used in the rest of the VLSI design flow. Therefore, the proposed method is transparent to the VLSI design flow and can be easily integrated into it.

4.1 Aging and Voltage-Drop Analysis

In this step, the results of Synthesis and Place & Route steps of the VLSI design flow are used to discover the flip-flops which are *aging-critical*, *voltage-drop critical*, and *timing-critical*.

Aging-critical flip-flops are those flip-flops which experience large impact of aging, i.e. flip-flops under S-BTI. To find the aging-critical flip-flops we need to extract the SP of the flip-flops. Therefore, we perform a gate-level simulation running some representative workloads. The representative workloads are pieces of workloads which are typically executed on the circuit. The result of the gate-level simulation is the Voltage Change Dump (VCD) of all nets inside the circuit. Based on this information we can collect SP of all flip-flops and determine the aging-critical flip-flops.

Dynamic power profiles of circuit components can be extracted from the VCD reports. We estimate the dynamic voltage-drop in the circuit based on the power profiles and the layout and packaging of the circuit. This accounts for the resistive and inductive components of the voltage fluctuation. We generate a *voltage-drop map* of the circuit by evaluating the *maximum voltage-drop* of each cell (gates, flip-flops, etc.) over the time and over different workloads. As a result, we find the maximum amount of voltage-drop that each flip-flop experiences over time.

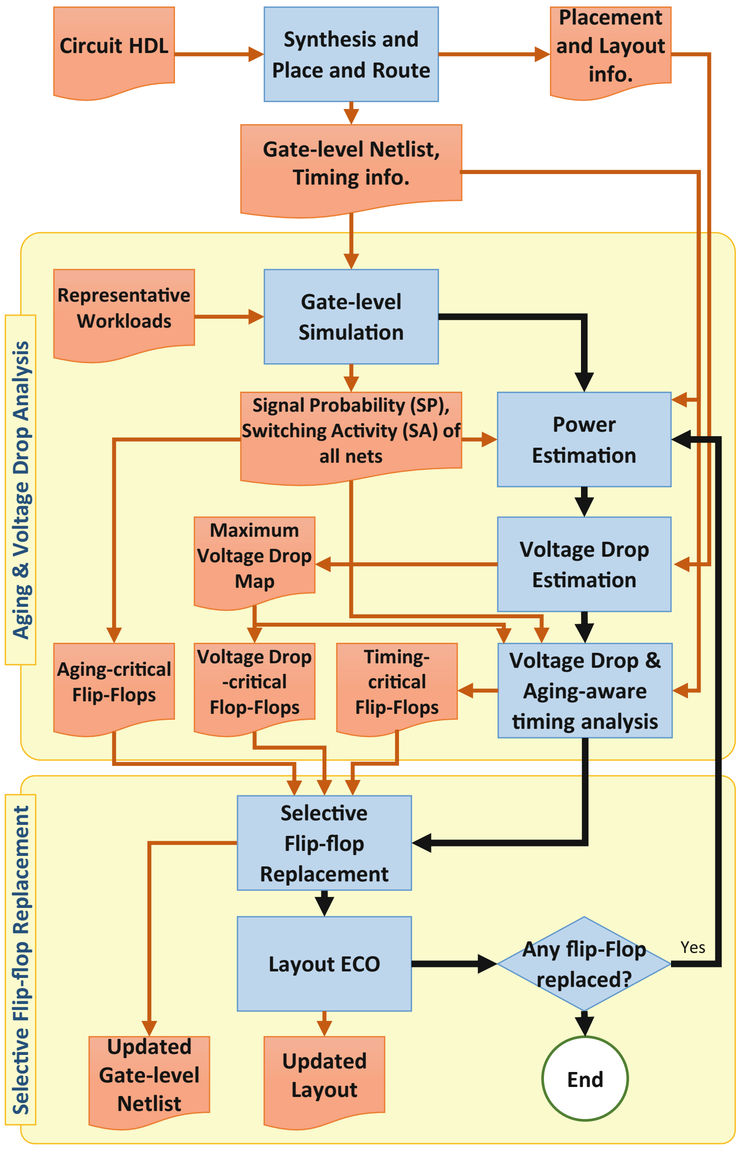

Circuit HDL
Synthesis and Place and Route
Placement and Layout info.
Gate-level Netlist, Timing info.
Aging & Voltage Drop Analysis
Representative Workloads
Gate-level Simulation
Signal Probability (SP), Switching Activity (SA) of all nets
Power Estimation
Maximum Voltage Drop Map
Voltage Drop Estimation
Aging-critical Flip-Flops
Voltage Drop -critical Flop-Flops
Timing-critical Flip-Flops
Voltage Drop & Aging-aware timing analysis
Selective Flip-flop Replacement
Selective Flip-flop Replacement
Layout ECO
Any flip-Flop replaced?
Yes
Updated Gate-level Netlist
Updated Layout
End

Fig. 7 Circuit optimization flow using the proposed selective flip-flop optimization method

Accordingly, the flip-flops which experience a large amount of voltage-drop are extracted.

Furthermore, the gate-level simulation results are used to perform a *voltage-drop and aging-aware timing analysis* which obtains the delay of circuit paths under variability impacts. We extended the aging-aware timing analysis in [8] by considering voltage-drop related information. This is done by characterizing the cells at two different supply voltages: the nominal V_{dd} and the supply voltage considering the maximum drop $(1 - \frac{R}{100})V_{dd}$. Then, for each gate/flip-flop in the gate-level netlist, based on the amount of voltage-drop on the gate, we perform a linear interpolation among the standard cell library entries for two supply voltages and find the corresponding timing information. The linear interpolation is a valid method under the assumption of limited change in the supply voltage, as shown in Fig. 3. For a more aggressive voltage fluctuation, it could be necessary to characterize the standard cell libraries for a few intermediate supply voltage values and employ a PCHIP method. Accordingly, we find the timing-critical flip-flops, which are parts of the critical and near-critical paths of the circuit considering the impact of variations.

4.2 Selective Flip-Flop Replacement

In the selective flip-flop replacement step, we replace the flip-flops which are timing-critical, aging-critical, or voltage-drop-critical with their optimized counterparts for such aging and/or voltage-drop conditions. Although a small portion of the flip-flops are replaced during the flip-flop replacement process, the circuit layout, timing, and power properties change since the replaced flip-flops are timing-critical and may have different area and power characteristics. Therefore, the proposed flip-flop replacement is an iterative process which replaces a number of flip-flops with the optimized versions in each iteration. The iterative process continues until no flip-flop needs to be replaced by an optimized version anymore.

In iteration i of the method, we assume that the circuit delay is D^i based on timing analysis results, and d_j^i is the maximum delay of the paths terminating at flip-flop j (including the delay of the flip-flop as well). Therefore, in each iteration:

1. We choose the timing-critical flip-flops with a timing slack value of less than $k\%$ of the circuit delay, i.e. when:

$$d_j^i \geq \left(1 - \frac{k}{100}\right) D^i, \quad j : \text{index of flip-flops.}$$

2. Among these flip-flops, those which are also included in the aging-critical and/or voltage-drop-critical flip-flops, are replaced with the optimized versions.
3. A trial voltage-drop and aging-aware timing analysis is performed and the circuit delay (D^{i++}) is determined considering the replaced flip-flops.
4. We keep the optimized flip-flops only when the corresponding path delay of the flip-flops *before optimization* is larger than a percentage of the evaluated circuit delay (D^{i++}):

$$d_j^i > r \times D^{i++} \Longrightarrow FF_j \rightarrow FF_{j,opt}. \tag{5}$$

The rest of the updated flip-flops in this iteration are rolled back to the original versions. Please note that we also consider a ratio $r < 1$ into Eq. (5) to compensate for the calculation errors due to simulation.

5. The layout and gate-level netlist of the circuit are updated. The layout is only updated if a cell with larger area is used (particularly applicable to the flip-flops under both aging and voltage-drop as explained in Sect. 3).
6. In case any flip-flop is replaced by an optimized version during this iteration, we need to start a new iteration because the timing and power specification of the circuit are modified. This is done by re-executing the aging and voltage-drop analysis, as explained in Sect. 4.1. The gate-level simulation, which is a time consuming process, does not need to be repeated as its results are not affected by the flip-flop replacement.

The above flow replaces minimum number of flip-flops with the optimized versions and impose minimum amount of overhead to the circuit. In our simulations the flow is terminated within a few iterations, since the changes in the circuit layout, power, and timing are not extensive.

5 Results and Discussions

In this section, we evaluate the efficiency of the proposed selective flip-flop optimization based on simulation results.

5.1 Simulation Setup

We applied the method to several flip-flop topologies, namely C2MOS latch, Dynamic/Static Single Transistor Clocked latch (DSTC/SSTC), and Semi-Dynamic flip-flop (SDFF) [31]. The flip-flops are implemented using 45 nm Bulk CMOS Predictive Technology Model (PTM) transistors [39]. All flip-flops are initially optimized for the minimum PDP in the fresh state (original design). The aging parameters of the model proposed in [4] are tuned so that the post-aging delay of a Fan-Out 4 (FO4) inverter increases by 10% at $\mathrm{SP} = 0.5$ over 5 years. For delay and leakage measurements, the output load of flip-flops is set to FO4, and the cells are characterized at room temperature and at different supply voltages, ranging from 80 to 100% of the nominal supply voltage of the technology node.

We used Leon3 processor as a case study for our proposed method. We used Nangate 45 nm open cell library for combinational logic, and aging assumptions are the same as described at the beginning of this section. The processor is synthesized using Synopsys Design Compiler and placement and routing is done using Cadence EDI [5].

We executed various MiBench workloads on the synthesized Leon3 processor and extracted the VCD files. Based on the VCD files, the SP of each node of the synthesized circuit is calculated and the power consumption of the gates and flip-flops is calculated using Synopsys Power Compiler. The voltage-drop map of the processor is also extracted using VoltSpot tool [38], which is able to extract the voltage-drop caused by both resistive and inductive components.

Please note that the proposed technique is not restricted to a specific working condition or flip-flop topology. We proceed with presenting detailed results and analysis for a C2MOS flip-flop. Then, we discuss the results for other types of flip-flops concisely. Afterwards, the dependency of the improvement achieved by the proposed method to the excessive leakage will be investigated. At the end of this section, the impact of using optimized flip-flops on a Leon3 processor lifetime will be demonstrated.

5.2 *Detailed Optimization Results of C2MOS Flip-Flop*

We apply the proposed optimization flow presented in Sect. 3.5 (see Fig. 6) to C2MOS flip-flop design to create optimized flip-flops for aging and voltage-drop resilience. In order to create the aging-resilient versions of the C2MOS flip-flop, we let the optimizer to consider designs with up to 25% more leakage compared to the original flip-flop by setting the coefficient β in Table 1 to 0.25. At this point, we limit the area of the flip-flop to the area of the original flip-flop, i.e. $\lambda = 0$. Please note that the total overhead of the leakage power for the entire circuit would be negligible since the number of optimized flip-flops in the design would be limited. For example, if according to Sect. 2.3, 12.45% of flip-flops are working under S-BTI, and the leakage overhead of an optimized flip-flop would be less than 25%, the leakage overhead imposed on the flip-flops would be *at most* 3.11% (much less overhead when considering the entire processor design). The aging and voltage-drop resilient version of the C2MOS flip-flop can be created by assuming an extra area up to 20% and more leakage overhead. For this, we assume $\lambda = 0.2$, $\beta = 1$. Using the extra area, the optimizer is able to find a better design for those flip-flops which are timing-critical and are under large impact of aging and voltage-drop. Since these flip-flops are very rare, but have significant impact on the overall processor lifetime and reliability, it is effective to spend more area for large reliability and lifetime gains.

Table 2 compares the characteristics of an original and optimized C2MOS flip-flop (such as setup-time (U), clock-to-q (D_{CQ}), data-to-q (D_{DQ}), delay, and leakage) in three different optimization scenarios:

Scenario 1 *post-aging PDP*, optimized for PDP in post-aging.

Scenario 2 The proposed method (optimized for aging), in which the flip-flop is optimized for aging resiliency, by minimizing its delay for post-

Table 2 C2MOS flip-flop characteristics for (1) Original flip-flop (Optimized for PDP in the fresh state), (2) Optimized flip-flop for PDP in post-aging [1], and optimized by the proposed method for (3) only aging, and (4) for aging + vdrop

	Original (optimized for fresh PDP)			Post-aging PDP (similar to [1])			Proposed method					
							Optimized for aging			Optimized for aging + vdrop		
				Scenario 1			Scenario 2			Scenario 3		
Parameters[c]	Fresh	Aged	Aged+vdrop[b]	Fresh	Aged	Aged+vdrop	Fresh	Aged	Aged+vdrop	Fresh	Aged	Aged+vdrop
U_{LH} (ps)	20.2	22.6	29.2	25.6	30.0	32.9	23.0	24.6	30.3	30.0	30.0	37.7
$D_{CQ,LH}$ (ps)	78.3	101.8	123.3	91.8	97.6	117.5	83.3	88.8	103.9	72.5	77.5	92.2
$D_{DQ,LH}$ (ps)	98.5	124.4	152.5	117.4	125.6	150.4	106.3	113.4	134.2	102.5	107.6	129.9
U_{HL} (ps)	16.6	30.8	42.4	13.7	28.2	39.6	16.0	30.6	40.6	11.1	24.6	30.6
$D_{CQ,HL}$ (ps)	78.1	100.5	121.0	82.9	97.9	117.2	75.9	88.5	106.4	65.8	75.6	91.7
$D_{DQ,HL}$ (ps)	94.7	131.3	163.4	96.6	126.1	156.8	91.9	119.1	147.0	76.9	100.2	122.3
Delay (ps) (Sect. 2.1)	**98.5**	132.6	165.7	117.4	126.1	157.1	106.3	119.4	147.0	102.5	107.6	129.9
Leakage (nW)	44.3	30.7	15.5	42.0	27.9	14.1	46.4	31.4	15.8	67.8	46.1	23.1
PDP	4368	4074	2573	4936	3525	2215	4928	3744	2318	6952	4963	3000
Delay degradation, Eq. (6)[a]	–	35%	68%	–	28%	60%	–	21%	49%	–	9.2%	32%
Excessive leakage	–			−5.2%			4.7%			53%		

The results are reported for “fresh”, “aged,” and “aged + vdrop” states and under SP0[d] aging
[a]The reference for calculating the delay degradation is 98.5 ps (Original, fresh flip-flop)
[b]Measurements are done under 10% delay degradation assumption (Sect. 5.1) and 10% voltage-drop
[c]Dynamic power is not reported because it is irrelevant for flip-flops under S-BTI as these flip-flops do not change state frequently
[d]Optimization results for SP1 are much better. For example, the delay degradation of the proposed method for aging is only 11% (for SP0 it is 21%)

aging. The acceptable excessive area and leakage are 0% and 25%, respectively ($\beta = 0.25$, $\lambda = 0$).

Scenario 3 The proposed method (optimized for aging + vdrop), in which the flip-flop is optimized for aging and voltage-drop resiliency, by minimizing its delay for post-aging and under voltage-drop impact. The acceptable excessive area and leakage are 20 and 100%, respectively ($\beta = 1$, $\lambda = 0.2$).

The optimization results in Table 2 are reported for "fresh" state (no aging or voltage-drop), for "aged" state (under S-BTI aging SP0 for 5 years), and for "aging + vdrop" state (when the flip-flop is aged under S-BTI for 5 years, and when the supply voltage is dropped by 10%). Setup-time, clock-to-q, and data-to-q values are presented for LH/HL transitions and the delay is calculated according to Sect. 2.1. The delay degradation is the *relative post-aging delay increase of a design compared to the fresh delay of the original design* (marked as bold in the table):

$$\text{delay degradation} = \frac{\text{delay}_{\text{opt.,aged}} - \text{delay}_{\text{orig.,fresh}}}{\text{delay}_{\text{orig.,fresh}}}. \tag{6}$$

Since the optimized flip-flop will replace the corresponding flip-flop in the design, the delay degradation is compared to the fresh delay of the original flip-flop in order to give a better understanding of how close the aged delay of the optimized flip-flop is to the fresh delay of the original design.

Basically, scenario 1 is similar to the methods proposed in many flip-flop optimization methods such as [1, 13] in the sense that they consider a multiplication of energy and delay (e.g., the PDP or the Energy Delay Product (EDP)) as the optimization target. Scenario 1 is able to effectively reduce the PDP by increasing the delay and reducing the leakage, but this may result in an unacceptable timing for S-BTI corners. Table 2 shows that due to not considering the flip-flop delay as the optimization target, the PDP methods cannot find the optimum aging-resilient sizing for S-BTI corners.

As presented, for the original flip-flop, the fresh delay of LH and HL paths is almost identical (see $D_{DQ,LH}$ and $D_{DQ,HL}$), but after aging HL path is much slower than LH path. This leads to 35% delay degradation due to only aging and about 68% when aging and voltage-drop affect the flip-flop. When this flip-flop is optimized for scenario 1, the delay is not reduced well enough because the main concern is PDP not delay. On the other hand, in scenario 2 (proposed method, only for aging), the optimizer alters the sizing to equalize the post-aging delay of the LH/HL paths to achieve the smallest possible post-aging delay with respect to the constraints (119.4 ps). In this case, the post-aging delay is increased by 21% compared to the fresh delay of the original flip-flop. Also the leakage overhead is limited to 4.7%. Since the flip-flop operates in S-BTI zone, the switching rate of the flip-flop is very small. This means that its dynamic power is almost negligible. Therefore, the total power in of flip-flops under S-BTI is determined by the leakage power.

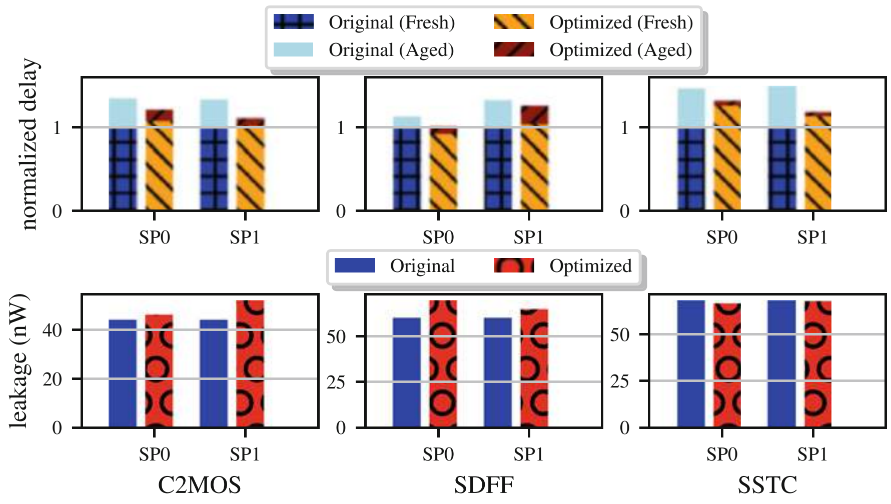


Fig. 8 Performance of the original flip-flop vs. the flip-flop optimized by the proposed method at SP0 and SP1, before and after aging (5 years)

Even though scenario 2's design is much better for flip-flops which are only under the aging impact compared to the original and the state-of-the-art [1] flip-flop designs, the impact of 10% voltage-drop is significant on the delay, i.e. 49% delay degradation. The flip-flop optimization results for scenario 3 show that such flip-flops are more resilient against both aging and voltage-drop impacts. These flip-flops consume about 53% more leakage; however, the delay degradation is only 32% under both aging and voltage-drop. Please note that the number of flip-flops under such condition is very small. Therefore, using flip-flops optimized by scenario 3 has negligible impact on the overall processor power consumption.

5.3 *Optimization Results for Other Flip-Flops*

Figure 8 provides the optimization results for a set of representative flip-flops. It compares the delay and the leakage of the original and optimized flip-flops, for both fresh and post-aging states. All delay values are normalized to the fresh delay of the corresponding original flip-flops (which are 114.8 ps for C2MOS, 28.5 ps for SDFF, and 71.0 ps for SSTC).

For C2MOS flip-flop, the proposed method reduces the delay degradation in Eq. (6) to 21%, while the delay degradation of the original design is 35% (14% improvement). This flip-flop has a symmetric structure, which means it can have balanced timing for LH/HL transitions (shown in Fig. 2b), while some flip-flop topologies such as SDFF, always have an unbalanced timing for LH/HL transitions due to their internal structure. For example, in an SDFF, the delay of HL transition is

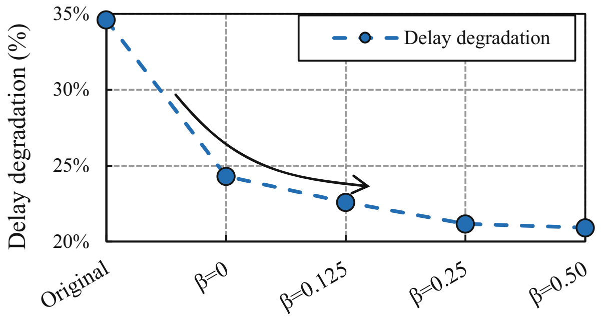


Fig. 9 Delay of C2MOS flip-flops optimized for SP0 aging using extra leakage (scenario 2). Delay degradation saturates as β increases (after $\beta = 0.25$)

always smaller than the LH transition. The reason is that, an intermediate precharged node in this flip-flop should be discharged in LH transition in order to transfer the input "one" to the output, while for the HL transition no such discharging is required. Hence, the slower path is always the LH path. This may worsen the aging if it is coupled with unbalanced aging. For these flip-flops, the optimizer minimizes the delay of the slower path by taking as much area as it can from the faster path, and giving the area to the slower path. For SDFF, this is attained with 15.8% additional leakage at SP0, but it leads to better S-BTI resiliency.

5.4 Delay-Leakage Trade-Off

In order to understand the trade-off between additional leakage and delay, we optimized a C2MOS flip-flop with several excessive leakage amounts ranging from 0 to 50% (i.e. $\beta \in \{0, 0.125, 0.25, 0.5\}$). As shown by Fig. 9, lower delay degradation can be achieved by allowing the optimization method to design flip-flops with higher leakage. However, the improvement saturates as β increases. Hence, providing extra leakage to the optimizer is only beneficial until about 25%, because the improvement in the delay is not significant. Please note that the designed flip-flops with looser leakage constraints, i.e. higher β, do not necessarily have very high leakage. As shown in Table 2, the optimized flip-flop in scenario 2 (only aging) has only 4.7% extra leakage while providing much better resiliency against S-BTI aging compared to the original flip-flop and scenario 1 (state-of-the-art work).

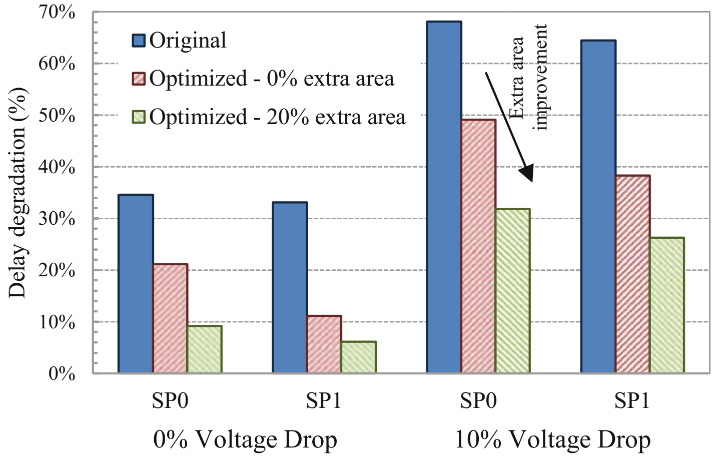


Fig. 10 Comparison of the aging-induced delay degradation under impact of voltage-drop, for original flip-flop, optimized flip-flop with 0% extra area allowance (scenario 2), and optimized flip-flop with 20% extra area allowance (scenario 3). The voltage-drop induced delay increase may be compensated by 20% upsizing of the flip-flop cell during the optimization

5.5 Delay-Area Trade-Off

The impact of a small amount of extra area on the resiliency of the flip-flops against both aging and voltage-drop impacts is studied by changing parameter *excessive area overhead* λ (see Table 1). We run the optimization flow in Sect. 3 for $\lambda \in \{0, 0.2\}$ values and compare the results to the original flip-flop design. Based on the results shown in Fig. 10, the flip-flop designs with no extra area, i.e. scenario 2, exhibit good resiliency against aging; however, under the impact of 10% voltage-drop it has up to 49% delay degradation. Under the impact of voltage-drop, the flip-flop designed with 20% extra area exhibits much better characteristics with maximum 32% delay degradation. This observation confirms that using flip-flops with 20% extra area can be beneficial for the cases when both aging and voltage-drop impacts are severe.

5.6 Circuit-Level Results

The proposed selective flip-flop optimization method presented in Sect. 4 is applied to Leon3 processor with the setup presented in Sect. 5.1 to evaluate the overall impact on the processor timing and reliability. The "original flip-flop" designs are optimized for different output loads for minimum PDP in the fresh state, while the "optimized flip-flop" designs for "aging" and "aging + vdrop" are obtained by applying the proposed method. Therefore, per each original flip-flop design for a

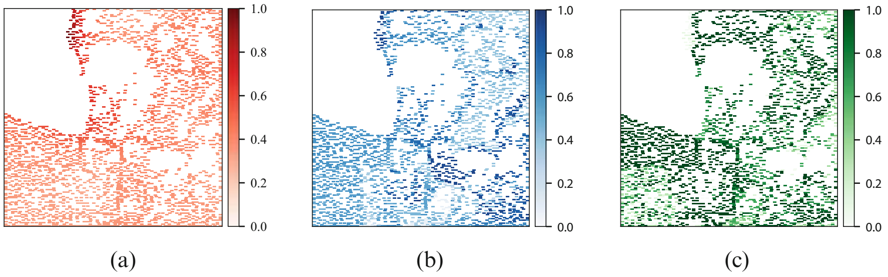


Fig. 11 The layout map of the Leon3 flip-flops during the execution of some MiBench workloads on Leon3, showing relative voltage-drop criticality, timing criticality, and aging-criticality of different flip-flops. Values close to "1" correspond to higher criticality, and values closer to "0" represent the non-critical parts. The top-left part of the processor layout is filled by combinational gates. (**a**) Relative voltage-drop criticality of flip-flops. (**b**) Relative timing criticality of flip-flops. (**c**) Relative aging-criticality of flip-flops

specific output load, there are different optimized designs for S-BTI corners SP0 and SP1 as well as no-vdrop and max-vdrop conditions (according to Sect. 3.5).

The timing of Leon3 processor is evaluated using the "aging and voltage-drop analysis" step of the proposed flow (see Fig. 7). This incorporates using an improved version of an aging-aware timing analysis tool [8] which also considers the impact of supply voltage variation as explained in Sect. 4.1. This timing analysis determines the processor delay under runtime variation impacts.

Figure 11 illustrates the timing of Leon3 flip-flops on the processor layout as well as the calculated impacts of voltage-drop and aging on the processor timing. The presented plots are all normalized to the maximum values (maximum voltage-drop, maximum delay, maximum aging) for better visualization. Therefore, higher values (darker colors) represent a critical situation. Figure 11a presents voltage-drop of the flip-flops extracted using the "aging and voltage-drop analysis" step. The voltage-drop values are normalized to the maximum voltage-drop value extracted during the simulations. As shown, many flip-flops experience at least a moderate voltage-drop during the workload execution. However, the flip-flops on the top-left corner of the layout experience heavy voltage-drop. The timing criticality of the flip-flops is also shown in Fig. 11b. The flip-flops with lower timing slack have values closer to 1.0 in this figure (darker). Interestingly, some of the flip-flops on the top-left corner are also timing-critical. Additionally, the aging-criticality of the flip-flops is presented in Fig. 11c. It is shown that many flip-flops which are under S-BTI are also timing-critical. Most importantly, a few timing-critical flip-flops are affected by both aging and voltage-drop impacts.

Table 3 presents processor delays obtained in fresh state, i.e. no aging or voltage-drop, and when under aging and voltage-drop impacts. We compare the delay of original processor (before applying the proposed method) with the delay of the optimized processors, under runtime variation impacts (aging and voltage-drop) after 7 years. The results are reported for:

Table 3 Processor delay comparison when (1) using only original flip-flops, and (2) using proposed method

	Processor delay in fresh state	Processor delay after 7 years + voltage-drop	Delay degradation	Guardband reduction	Equivalent lifetime improvement
Using original flip-flops	1389.6 ps	1528.2 ps	9.97%	–	–
Proposed (only aging)	1391.3 ps	1494.8 ps	7.44%	33.4 ps	30.8%
Proposed (aging + voltage-drop)	1379.7 ps	1486.7 ps	7.75%	41.5 ps	36.9%

1. "Original processor": using only original flip-flops,
2. "Optimized processor for aging": when only the impact of aging is considered during optimization,
3. "Optimized processor for aging and voltage-drop": when the impacts of aging and voltage-drop are considered during optimization.

The "original processor" is synthesized using the original flip-flops designs in Table 2. Then, we apply the proposed selective flip-flop optimization in two modes: (I) when only aging is considered, and (II) when both aging and voltage-drop are considered. This obtains two versions of the optimized processor, i.e. "Optimized processor for aging" and "Optimized processor for aging and voltage-drop." In the optimization flow presented in Sect. 4.2, we assume $k = 0.15$. Therefore, all flip-flops with a slack value less than 15% of the processor delay are assumed as timing-critical flip-flops. Additionally, we assume $r = 0.95$, which means up to 5% calculation error guardband in the timing analysis method is acceptable. In fact, r value depends on the accuracy of the timing analysis method. After replacing the critical flip-flops according to the proposed method, the processor delay is obtained again using the "aging and voltage-drop analysis" step.

According to the table, delay of the "original processor" is increased by 9.97% after 7 years. This translates into 138.6 ps timing guardband for 7 years of circuit operation, i.e. $T_{clk} \geq 1528.2$ ps. The "optimized processor for aging" has better delay 1494.8 ps under the impacts of aging and voltage-drop which reduces the required timing guardband by 33.4 ps for 7 years of operation, hence optimizing the performance. Therefore, the degradation rate of this optimized processor is such that it can operate for 9.2 years (30.8% lifetime improvement), if it is used with the timing margins of $T_{clk} = 1528.2$ ps. Finally, the required timing guardband of "Optimized processor for aging and voltage-drop" is further reduced by 41.5 ps compared to the original processor. Therefore, the lifetime of the processor is improved by 36.9% (9.6 years).

The reason for the achieved improvements in Table 3 is explained by Fig. 12. Here, we only plotted the delay of timing-critical flip-flops with a slack smaller

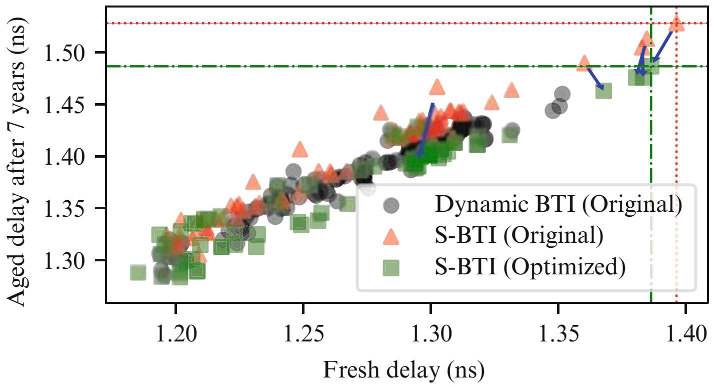


Fig. 12 Fresh delay (no aging, no voltage-drop) vs. increased delay (aged and 10% voltage-drop) of critical paths of Leon3 processor. The proposed selective flip-flop optimization method replaces the original flip-flops under S-BTI (red) with the optimized flip-flops (green) and suppresses the aging and voltage-drop degradation of the most critical paths

than 15% of the processor delay (under aging and voltage-drop impacts). With this assumption, there are 261 timing-critical flip-flops. Among the timing-critical flip-flops, 92 flip-flops are under S-BTI impact (i.e. $0 \leq SP < 0.01$ or $0.99 < SP \leq 1$), 235 flip-flops experience at least 33% relative voltage-drop. After applying the selective flip-flop optimization method, 96 flip-flops are replaced with optimized versions, from which 39 flip-flops are upsized (due to both aging and voltage-drop impact).

As the optimized flip-flops constitute about 4% of all flip-flops in Leon3, the overall leakage overhead with this method is 0.22% according to power analysis results using Synopsys Design Compiler. Moreover, there is virtually no dynamic power overhead because the replaced flip-flops are mostly under S-BTI impact and they rarely switch. The additional area overhead is also very negligible because only 39 flip-flops are replaced by the upsized versions (less than 0.1% area overhead). The ECO process easily fits these flip-flops into the existing layout by slightly moving other cells. Please note that the impact of the voltage-drop and aging on the driving logic paths is much less compared to the flip-flops. Therefore, these paths are degraded at a much lower rate.

6 Comparison with the Related Work

Various methods have been proposed to address the impact of aging and voltage-drop on flip-flops [1, 13, 23, 25]. For example, [1] proposes a method to improve flip-flop reliability for a set of corners with different working conditions such as temperatures and voltages by altering the sizing of transistors. These studies mostly optimize flip-flops for dynamic BTI stress condition, and flip-flops under static BTI are mostly overlooked. As explained, the traditional optimization techniques such as optimization for the PDP, or EDP cannot effectively address the delay increase of flip-flops under such stress. There are techniques to reduce the overall impact of voltage-drop on VLSI circuits by skewing the clock input of the flip-flops at design-time in order to reduce the peak current at clock edge [9, 35]. However, these methods are not applicable to flip-flops with zero (or close to zero) timing slack on the critical paths. The techniques at high abstraction level by software-guided thread scheduling [27] or by voltage emergency prediction [26] also impose additional overhead at another abstraction level than circuit-level, in order to address a circuit-level problem.

7 Summary

In many cases, NTC circuits are required to operate over a wide voltage range in order to achieve energy efficiency and satisfy performance constraints as needed. Therefore, an NTC circuit may be exposed to reliability issues such as aging and voltage-drop which are significant in the super-threshold region.

In this chapter, we discussed that a non-negligible portion of circuit flip-flops may be under severe aging or large voltage-drop impact, which leads to timing and functional failures. Therefore, these flip-flops need to be treated separately and specific stress-tolerant designs should be used in order to improve the reliability and lifetime. Accordingly, we propose a method to selectively optimize the flip-flops operating under severe aging stress and/or voltage-drop conditions. The proposed optimization flow resizes the flip-flop transistors to obtain the variability-resilient cells. Then, flip-flops which are under the impact of aging and/or voltage-drop are determined using a variation-aware static timing analysis tool, and are replaced by the optimized flip-flops which can withstand aging and voltage-drop impacts much better. Simulation results show that the proposed selective flip-flop optimization method can reduce Leon3 processor timing guardband, and improve the lifetime of the processor by 36.9%, with negligible power and area overhead.

References

1. Abrishami, H., Hatami, S., Pedram, M.: Multi-corner, energy-delay optimized, NBTI-aware flip-flop design. In: International Symposium on Quality Electronic Design (ISQED), pp. 652–659 (2010). https://doi.org/10.1109/ISQED.2010.5450509
2. Ajami, A.H., Banerjee, K., Mehrotra, A., Pedram, M.: Analysis of IR-drop scaling with implications for deep submicron P/G network designs. In: International Symposium on Quality Electronic Design (ISQED), pp. 35–40 (2003)
3. Amrouch, H., Ebi, T., Schneider, J., Parameswaran, S., Henkel, J.: Analyzing the thermal hotspots in FPGA-based embedded systems. In: International Conference on Field Programmable Logic and Applications (FPL), pp. 1–4 (2013)

4. Bhardwaj, S., Wang, W., Vattikonda, R., Cao, Y., Vrudhula, S.: Predictive modeling of the NBTI effect for reliable design. In: Custom Integrated Circuits Conference (CICC), pp. 189–192. IEEE, Piscataway (2006)
5. Cadence Encounter Timing System. http://www.cadence.com
6. Cadence Virtuoso Liberate Characterization Solution. http://www.cadence.com/products/cic/liberate/pages/default.aspx
7. Denney, J., Ramsey, C.: Comparison of finite-difference and spice tools for thermal modeling of the effects of nonuniform power generation in high-power CPUs. Hewlett-Packard J. **50**, 37–45 (1998)
8. Ebrahimi, M., Oboril, F., Kiamehr, S., Tahoori, M.B.: Aging-aware Logic Synthesis. In: International Conference on Computer-Aided Design (ICCAD), pp. 61–68 (2013)
9. Fishburn, J.P.: Clock skew optimization. IEEE Trans. Comput. **39**(7), 945–951 (1990)
10. Gaisler, A., Göteborg, S.: Leon3 multiprocessing CPU core. Aeroflex Gaisler (2010)
11. Gnad, D.R.E., Oboril, F., Kiamehr, S., Tahoori, M.B.: An experimental evaluation and analysis of transient voltage fluctuations in FPGAs. IEEE Trans. Very Large Scale Integr. VLSI Syst. **26**(10), 1817–1830 (2018)
12. Golanbari, M.S., Kiamehr, S., Ebrahimi, M., Tahoori, M.B.: Aging guardband reduction through selective flip-flop optimization. In: IEEE European Test Symposium (ETS) (2015)
13. Golanbari, M.S., Kiamehr, S., Tahoori, M.B., Nassif, S.: Analysis and optimization of flip-flops under process and runtime variations. In: International Symposium on Quality Electronic Design (ISQED) (2015)
14. Golanbari, M.S., Kiamehr, S., Ebrahimi, M., Tahoori, M.B.: Selective flip-flop optimization for reliable digital circuit design. IEEE Trans. Very Large Scale Integr. VLSI Syst. **39**(7), 1484–1497 (2020)
15. Guthaus, M.R., Ringenberg, J.S., Ernst, D., Austin, T.M., Mudge, T., Brown, R.B.: MiBench: a free, commercially representative embedded benchmark suite. In: IEEE International Workshop on Workload Characterization, pp. 3–14. IEEE, Piscataway (2001)
16. International Technology Roadmap for Semiconductors (ITRS). http://www.itrs2.net
17. Kaul, H., Anders, M.A., Mathew, S.K., Hsu, S.K., Agarwal, A., Krishnamurthy, R.K., Borkar, S.: A 320 mv 56 μw 411 GOPS/watt ultra-low voltage motion estimation accelerator in 65 nm CMOS. IEEE J. Solid State Circuits **44**(1), 107–114 (2009)
18. Kiamehr, S., Ebrahimi, M., Firouzi, F., Tahoori, M.B.: Extending standard cell library for aging mitigation. IET Comput. Digit. Tech. **9**(4), 206–212 (2015)
19. Kraft, D.: A software package for sequential quadratic programming. Forschungsbericht-Deutsche Forschungs- und Versuchsanstalt fur Luft- und Raumfahrt **88–28**, 1–33 (1988)
20. Krishnan, A.T., Cano, F., Chancellor, C., Reddy, V., Qi, Z., Jain, P., Carulli, J., Masin, J., Zuhoski, S., Krishnan, S., et al.: Product drift from NBTI: Guardbanding, circuit and statistical effects. In: International Electron Devices Meeting, pp. 4–3 (2010)
21. Mezhiba, A.V., Friedman, E.G.: Scaling trends of on-chip power distribution noise. IEEE Trans. Very Large Scale Integr. VLSI Syst. **12**(4), 386–394 (2004)
22. Nithin, S., Shanmugam, G., Chandrasekar, S.: Dynamic voltage (IR) drop analysis and design closure: Issues and challenges. In: International Symposium on Quality Electronic Design (ISQED), pp. 611–617 (2010)
23. Nunes, C., Butzen, P.F., Reis, A.I., Ribas, R.P.: BTI, HCI and TDDB aging impact in flip-flops. Microelectron. Reliab. **53**(9–11), 1355–1359 (2013)
24. Ramakrishnan, K., Wu, X., Vijaykrishnan, N., Xie, Y.: Comparative analysis of NBTI effects on low power and high performance flip-flops. In: International Conference on Computer Design (ICCD), pp. 200–205 (2008)
25. Rao, V.G., Mahmoodi, H.: Analysis of reliability of flip-flops under transistor aging effects in nano-scale CMOS technology. In: International Conference on Computer Design (ICCD), pp. 439–440 (2011)
26. Reddi, V.J., Gupta, M.S., Holloway, G., Wei, G.Y., Smith, M.D., Brooks, D.: Voltage emergency prediction: using signatures to reduce operating margins. In: International Symposium on High-Performance Computer Architecture (HPCA), pp. 18–29 (2009)

27. Reddi, V.J., Kanev, S., Kim, W., Campanoni, S., Smith, M.D., Wei, G.Y., Brooks, D.: Voltage smoothing: Characterizing and mitigating voltage noise in production processors via software-guided thread scheduling. In: International Symposium on Microarchitecture, pp. 77–88 (2010)
28. Reddy, V., Carulli, J., Krishnan, A., Bosch, W., Burgess, B.: Impact of negative bias temperature instability on product parametric drift. In: International Conference on Test, pp. 148–155 (2004)
29. Sato, T., Ichimiya, J., Ono, N., Hachiya, K., Hashimoto, M.: On-chip thermal gradient analysis and temperature flattening for SoC design. IEICE Trans. Fundam. Electron. Commun. Comput. Sci. **88**(12), 3382–3389 (2005)
30. Schlunder, C., Aresu, S., Georgakos, G., Kanert, W., Reisinger, H., Hofmann, K., Gustin, W.: HCI vs. BTI? - Neither one's out. In: IEEE International Reliability Physics Symposium (IRPS), pp. 2F.4.1–2F.4.6 (2012)
31. Stojanovic, V., Oklobdzija, V.G.: Comparative analysis of master-slave latches and flip-flops for high-performance and low-power systems. IEEE J. Solid State Circuits **34**(4), 536–548 (1999)
32. Sundareswaran, S.: Statistical characterization for timing sign-off: from silicon to design and back to silicon. Ph.D. Thesis, UT Austin (2009)
33. Tradowsky, C., Cordero, E., Deuser, T., Hübner, M., Becker, J.: Determination of on-chip temperature gradients on reconfigurable hardware. In: International Conference on Reconfigurable Computing and FPGAs (ReConFig), pp. 1–8 (2012)
34. Unger, S.H., et al.: Clocking schemes for high-speed digital systems. IEEE Trans. Comput. **C-35**(10), 880–895 (1986)
35. Vittal, A., Ha, H., Brewer, F., Marek-Sadowska, M.: Clock skew optimization for ground bounce control. In: International Conference on Computer-Aided Design (ICCAD), pp. 395–399 (1996)
36. Wang, W., Yang, S., Bhardwaj, S., Vrudhula, S., Liu, F., Cao, Y.: The impact of NBTI effect on combinational circuit: modeling, simulation, and analysis. IEEE Trans. Very Large Scale Integr. VLSI Syst. **18**(2), 173–183 (2010). https://doi.org/10.1109/TVLSI.2008.2008810
37. Wu, J.K., Wu, T.Y., Lu, L.Y., Chen, K.Y.: IR drop reduction via a flip-flop resynthesis technique. In: International Symposium on Quality Electronic Design (ISQED), pp. 78–83 (2008)
38. Zhang, R., Wang, K., Meyer, B.H., Stan, M.R., Skadron, K.: Architecture implications of pads as a scarce resource. In: International Symposium on Computer Architecture (ISCA), pp. 373–384 (2014)
39. Zhao, W., Cao, Y.: New generation of predictive technology model for sub-45nm design exploration. In: International Symposium on Quality Electronic Design (ISQED), pp. 585–590. IEEE Computer Society, Washington (2006)

EM Lifetime Constrained Optimization for Multi-Segment Power Grid Networks

Han Zhou, Zeyu Sun, Sheriff Sadiqbatcha, and Sheldon X.-D. Tan

1 Introduction

On-chip power supply or power-ground (P/G) networks provide power to the circuit modules in a chip from external power supplies. Since power grid wires experience the largest current flows on a chip, they are more susceptible to long-term reliability issues and functional failures. These reliability issues and failures typically come from metal electromigration (EM), excessive IR drops, and ΔI (Ldi/dt) noise along with recently emerging back end of line time-dependent dielectric breakdown (TDDB) [2, 3, 6].

As technology scales into smaller features with increasing current densities, EM-induced reliability deteriorates, the EM lifetime was projected to be reduced by half for each new technology node by ITRS 2015 [19]. As a result, EM still remains one of the top killers of copper based damascene interconnects for technologies in the sub-10 nm realm. This introduces additional challenges for designing robust power supply networks to satisfy the demanding design requirements.

An important step for power supply synthesis in the typical EDA design flow is sizing the wire width of the power grid stripes, after the topology of the power supply network has been determined, so that the minimum amount of chip area will be used while avoiding potential reliability failures due to electromigration and excessive IR drops. Numerous works have been proposed for the power supply network optimization in the past, primarily based on nonlinear or sequence of linear programming (SLP) methods [8–10, 13, 26, 27, 31].

To satisfy the EM reliability, all the existing methods use the current density of individual wires as the constraint, which is mainly based on the Black's EM

H. Zhou · Z. Sun · S. Sadiqbatcha · S. X.-D. Tan (✉)
Department of Electrical and Computer Engineering, University of California, Riverside, CA, USA
e-mail: hzhou012@ucr.edu; zsun007@ucr.edu; ssadi003@ucr.edu; stan@ece.ucr.edu

J. Henkel, N. Dutt (eds.), *Dependable Embedded Systems*, Embedded Systems,
https://doi.org/10.1007/978-3-030-52017-5_15

model. However, this constraint is too conservative for modern power grid networks. Furthermore, all existing power supply optimization methods fail to consider the aging effects. With recent advancements in physics-based EM models and numerical analysis techniques such as three-phase EM model [12, 24, 28, 33], it is possible to provide more accurate time to failure (TTF) estimation for multi-segment interconnects.

In this chapter, we present two new P/G network sizing and optimization techniques, which were first introduced in [35, 36]. We will summarize the key contributions and major computing steps from the P/G optimization technique considering the new physics-based EM models. The chapter is organized as follows: Sect. 2 describes the power grid network and its models. Section 3 presents the fundamentals of EM and the voltage-based EM immortality check method for general multi-segment interconnect wires. Section 4 outlines a physics-based three-phase EM model and a fast EM lifetime estimation method. Section 5 introduces the EM immortality constrained P/G network optimization problem and its programming-based solution. Section 6 presents the EM lifetime constrained P/G optimization method, which deals with the EM-induced aging effect. Section 7 summarizes this chapter.

2 Power Grid Modeling

Practical VLSI interconnects (especially the global networks such as power supply and clock networks) have many multi-segment wires as shown in Fig. 1. A multi-segment interconnect wire consists of continuously connected high-conductivity metal within one layer of metallization.

Figure 2 shows a typical mesh-structured P/G network with multi-layer power grids. The modeling assumptions for later optimization are listed as follows. Firstly, because of the concern with the long-term average effects of the current, we focus on the steady state (DC) problem, which means we are only interested in the resistance of the power grid networks. Secondly, the P/G network is composed of an orthogonal mesh of wires and contains multiple segments/branches, which is the typical P/G structure. Lastly, to simplify the problem, the circuits are modeled with shorted vias, which means the via resistance is ignored and vias will not be sized. Figure 3 shows the equivalent circuit of the power grid network in Fig. 2.

As a result, the power grid systems are linear and driven by the DC effective currents [17]. For a power grid network with n nodes,

$$G \times V = I \tag{1}$$

Fig. 1 Example of a multi-segment wire

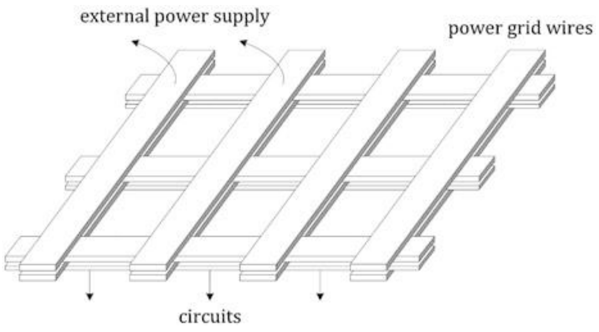


Fig. 2 A small portion of a typical power supply network [22]

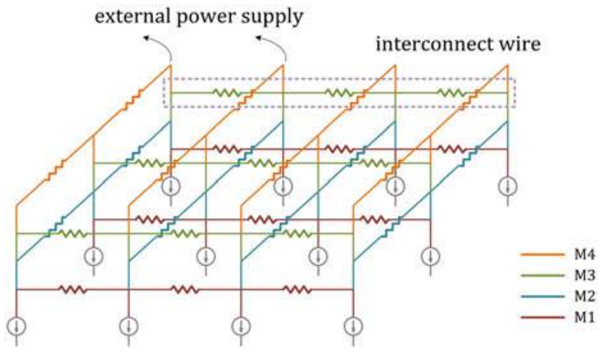


Fig. 3 Equivalent circuit of a small portion of a typical power grid

where G is a $n \times n$ conductance matrix; I is the current source vector; V is the corresponding vector of nodal voltages.

3 Electromigration Fundamentals

3.1 Electromigration Introduction

EM is a physical phenomenon of material migration caused by an electrical field. Wind force, which is produced by current flowing through a conductor, acts in the direction of the current flow and is the primary cause of EM [21]. During the migration process, hydrostatic stress is generated inside the metal wire due to momentum transfer between lattice atoms. Void and hillock formation are caused by conducting electrons at the opposite ends of the wire. The void may lead to early failure or late failure of the wire [1].

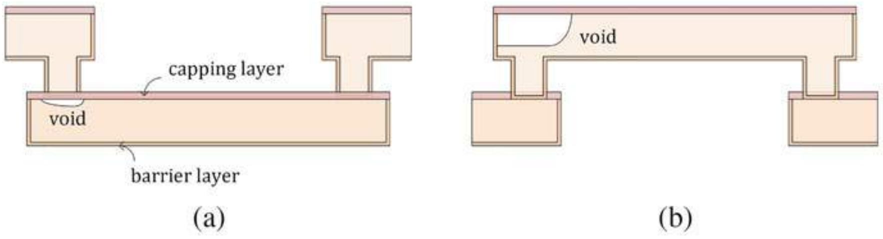


Fig. 4 Side-view of void formation: (**a**) void in a via-above line (early failure mode); (**b**) void in a via-below line (later failure mode)

Early failure typically happens in a via-to-via structure as shown in Fig. 4a. When the void forms in a via-above line and reaches critical size [16, 34], which equals the via's diameter, the via will be blocked by the void and thus the connection to the upper layer will also be blocked. This is because the capping layer is fabricated with dielectrics such as Si_3N_4 which will block the current flow. On the contrary, late failure typically happens in a via-below structure as shown in Fig. 4b. Since the barrier layer is fabricated with Ta whose resistivity is much higher than Cu, when the void reaches critical size, current can still go through the barrier layer. Sometimes early failure can happen in a via-below structure and late failure can happen in a via-above structure. Although the void can grow at these positions, the possibility is very low.

When the compressive stress at the anode continues to be built up, hillocks or extrusion may be formed, which will lead to a resistance decrease [30] and can potentially cause short-circuit failure. However, the void nucleation is still the dominant EM failure effect [15].

3.2 Steady State EM-Induced Stress Modeling

Steady state EM-induced stress modeling helps find the immortality information of the interconnect wire quickly as no complex calculations are required. For these kinds of models, stress on the cathode at steady state (σ_{steady}), which is the maximum stress the node experiences, is compared with critical stress (σ_{crit}). If σ_{steady} is lower than σ_{crit}, the wire is considered as immortal. One of the well-known steady state analysis method is *Blech product* [4], but it is only suitable for a single (i.e., one-segment) wire. Recently, a voltage-based EM immortality analysis method for multi-segment interconnect structures has been proposed [23, 24]. In this method, an *EM voltage* (V_E) is calculated as

$$V_E = \frac{1}{2A} \sum_{k \neq g} a_k V_k \tag{2}$$

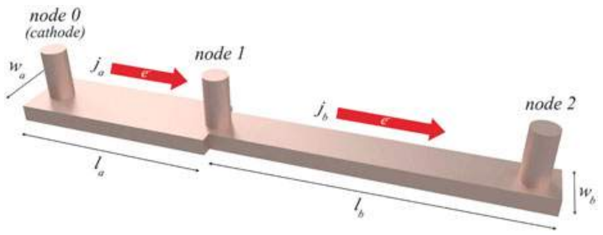


Fig. 5 Interconnect example for EM analysis for straight 3-terminal wire

where V_k is the normal nodal voltage (with respect to cathode node cat) at node k, a_k is the total area of branches connected to node k, and A is the total area of the wire. With voltage of node i (V_i), steady state stress at that node (σ_i) can be calculated as $\sigma_i = \beta(V_E - V_i)$, where $\beta = \frac{eZ}{\Omega}$, e is elementary charge, Z is effective charge number, and Ω is the atomic lattice volume. A *critical EM voltage* $V_{crit,EM}$ is defined by

$$V_{crit,EM} = \frac{1}{\beta}(\sigma_{\text{crit}} - \sigma_{\text{init}}) \tag{3}$$

where σ_{init} is the initial stress. In order to check whether the interconnect wire is immortal, we need to check the following condition

$$V_{crit,EM} > V_E - V_i \tag{4}$$

Note that $V_E - V_i$ is proportional to the stress at cathode node (σ_{cat}).

If this condition is met for all the nodes, EM failure will not happen. Since generally the cathode node has the lowest voltage within an interconnect wire, we may just check the cathode node instead of all the nodes, which means

$$V_{crit,EM} > V_E - V_{\text{cat}} \tag{5}$$

where V_{cat} is the voltage at the cathode. Note that inequality (5) can be applied to both power and ground networks.

The method can be illustrated using the following example. Figure 5 shows a 3-terminal wire. In this wire, node 0 is treated as the ground node. Current densities in two segments are j_a and j_b which may not be the same because they will be determined by the rest of the circuit. The *EM voltage* become

$$V_E = \frac{a_0V_0 + a_1V_1 + a_2V_2}{2A} = \frac{a_1V_1 + a_2V_2}{2A} \tag{6}$$

where

$$\begin{aligned} V_0 &= 0, & a_0 &= l_a w_a, & \sigma_0 &= \beta V_E \\ V_1 &= j_a l_a \rho, & a_1 &= l_a w_a + l_b w_b, & \sigma_1 &= \beta(V_E - V_1) \\ V_2 &= j_b l_b \rho + j_a l_a \rho, & a_2 &= l_b w_b, & \sigma_2 &= \beta(V_E - V_2) \end{aligned} \tag{7}$$

$$A = \frac{a_0 + a_1 + a_2}{2} \tag{8}$$

We can compare V_E and $V_{crit,EM}$ to see if this wire is immortal.

4 Transient EM-Induced Stress Estimation

In general, the failure process of an interconnect is divided into nucleation phase, incubation phase and growth phase. In the nucleation phase, the stress at the cathode keeps increasing. When it reaches critical stress, a void will be nucleated. The time to reach the critical stress is called nucleation time (t_{nuc}). After the nucleation phase, the void starts to grow (t_{inc}) and eventually leads to wire failure after a period of time (t_{growth}). The TTF or lifetime of the wire can be described as

$$TTF = t_{\text{life}} = t_{\text{nuc}} + t_{\text{inc}} + t_{\text{growth}} \tag{9}$$

4.1 Transient EM-Induced Stress Modeling

4.1.1 Nucleation Phase Modeling

It is well-known that the nucleation phase is accurately modeled by Korhonen's equation [20]

$$\frac{\partial \sigma(x,t)}{\partial t} = \frac{\partial}{\partial x}\left[\kappa\left(\frac{\partial \sigma(x,t)}{\partial x} + \Gamma\right)\right] \tag{10}$$

where $\kappa = \frac{D_a B \Omega}{k_B T}$, $D_a = D_0 \exp(-\frac{E_a}{k_B T})$, and $\Gamma = \frac{eZ}{\Omega}\rho_w j$. B is effective bulk elasticity modulus, Ω is atomic lattice volume, k_B is Boltzmann constant, T is temperature, Z is effective charge number, ρ_w is the wire electrical resistivity, x is coordinate along the line, t is time, and j is current density.

Korhonen's equation describes the stress distribution accurately; this PDE-based model is hard to solve directly using numerical methods and has very low efficiency for tree-based EM assessment. Recently a few numerical methods have been proposed such as finite difference methods [5, 11] and analytical expressions based approaches [7, 32]. In this work, an integral transformation method for straight multi-segment wires [32] is employed. Suppose we have a multi-segment wire, after discretizing Korhonen's equation, the stress can be expressed as

$$\sigma(x,t) = \sum_{m=1}^{\infty} \frac{\psi_m(x)}{N(\lambda_m)} \bar{\sigma}(\lambda_m, t) \tag{11}$$

where the norm of eigenfunctions $N(\lambda_m)$ is

$$N(\lambda_m) = \int_{\chi=0}^{L} [\psi_m(\chi)]^2 d\chi \tag{12}$$

and the transformed solution of stress $\bar{\sigma}(\lambda_m, t)$ is

$$\begin{aligned} \bar{\sigma}(\lambda_m, t) = &\left(\int_{\chi=0}^{L} \psi_m(\chi) \cdot \sigma_0(\chi) d\chi\right) e^{-\kappa\lambda_m^2 t} + \frac{1}{\lambda_m^2}\left(1 - e^{\kappa\lambda_m^2 t}\right) \\ &\cdot \sum_{k=1}^{n} \frac{eZ\rho}{\Omega} j_k \cdot \left(\cos\frac{x_{k-1}}{L} m\pi - \cos\frac{x_k}{L} m\pi\right) \end{aligned} \tag{13}$$

Eigenvalues λ_m and eigenfunctions $\psi(x)$ are the solutions of the Sturm–Liouville problem corresponding to the diffusion Eq. (10) and the boundary conditions, which are

$$\lambda_m = \frac{m\pi}{L}, \ \psi_m(x) = \cos\frac{x}{L} m\pi \tag{14}$$

With Eq. (11), given critical stress σ_{crit}, the nucleation time t_{nuc} can be obtained quickly by using nonlinear equation solving methods such as Newton's method or bisection method.

4.1.2 Incubation Phase Modeling

After the void is nucleated, the incubation phase starts. In this phase, resistance of the interconnect remains almost unchanged since the cross section of the via is not covered by the void and the current can still flow through the copper.

In power grid networks, the interconnect trees are generally multi-segment wires. All segments connected with the void can contribute to the void growth since electron wind at each segment can accelerate or slow down the void growth according to their directions. In this phase, void growth rate v_d is estimated to be [25]

$$v_d = \frac{D_a eZ\rho}{kTW_m} \sum_i j_i W_i \tag{15}$$

where j_i and W_i are the current density and width of the ith segment, respectively. W_m is the width of the main segment where the void is formed.

Then the incubation time t_{inc} can be expressed as

$$t_{\text{inc}} = \frac{\Delta L_{\text{crit}}}{v_d} \tag{16}$$

where ΔL_{crit} is the *critical void length.*

4.1.3 Growth Phase Modeling

After the incubation phase, the void fully covers the via, initiating the growth phase. In this phase the resistance starts increasing. It is important to note that, early failure and late failure have different failure mechanisms.

For early failure, the wire fails once the void covers the via, which means the wire fails at the end of incubation phase and there is no growth phase ($t_{\text{growth}} = 0$). Hence the failure time is the sum of t_{nuc} and t_{inc}.

For late failure, after the void size reaches the critical size, there will be no open circuit because the current can still flow through the barrier layer. In this case, the void growth will lead to resistance increase. When the resistance increases to the critical level, the interconnect wire is considered to be failed. The growth time for late failure is

$$t_{\text{growth}} = \frac{\Delta r(t)}{v_d \left[\dfrac{\rho_{Ta}}{h_{Ta}\,(2H + W)} - \dfrac{\rho_{Cu}}{HW} \right]} \tag{17}$$

where ρ_{Ta} and ρ_{Cu} are the resistivity of tantalum (the barrier liner material) and copper, respectively. W is the line width, H is the copper thickness, and h_{Ta} is the liner layer thickness.

However, the void may saturate before reaching the *critical void length.* The saturation length is expressed in [18] as

$$L_{ss} = L_{\text{line}} \times \left[\frac{\sigma_T}{B} + \frac{eZ\rho jL}{2B\Omega} \right] \tag{18}$$

where L_{ss} is the void saturated length, L_{line} is the total length of the wire, and σ_T is thermal stress. Void growth may stop before the calculated t_{growth} because of the saturated void. If it happens, we treat the wire as immortal or its lifetime is larger than the target lifetime.

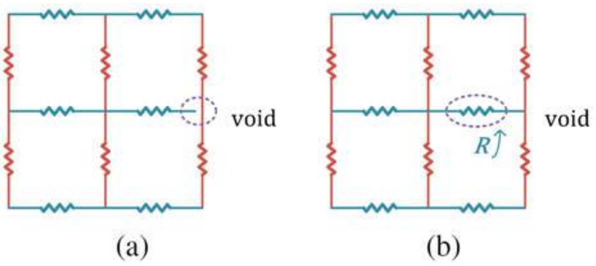


Fig. 6 The electrical impact of different failure mechanisms on the interconnect wires: (**a**) early failure mode; (**b**) late failure mode

4.2 Transient EM Analysis for a Multi-Segment Interconnect Wire

One important aspect of transient EM analysis is calculating the lifetime of a given wire and its electrical conditions. If the increased resistance of the nucleated branch exceeds a threshold, the interconnect tree is marked as failed.

To compute the lifetime t_{life} of a given wire, we need to make sure that the wire is mortal and Eq. (5) is not satisfied. If a target lifetime t_{target} is given, the analysis method will give the resistance change ΔR at the target lifetime.

For those mortal wires, we start with time $t = 1000$ years and use bisection method to find t_{nuc}. The transient hydrostatic stress will be computed by Eq. (11). Once the stress of one segment hits the critical stress, the wire is deemed as nucleated.

Then we need to determine if the wire is void incubation phase immortal. If the saturated void length is less than the critical length, the incubation time (eventually the lifetime) becomes infinite and the resistance remains unchanged.

Otherwise, the failure mode of the wire should be determined by looking at the current direction in the cathode node based on the patterns in Fig. 4.

If the wire is in the early failure mode, then the wire will become an open circuit: the whole interconnect tree will be disconnected from another interconnect wire as shown in Fig. 6a. For the wire in the late failure mode, we have another solution. The wire resistance change will be incurred and the growth time will be computed when the resistance change reaches the threshold as shown in Fig. 6b. If the target lifetime is given, then the wire resistance change ΔR will be computed at the target lifetime.

5 EM Immortality Constrained Optimization for Multi-Segment Interconnects

Study and experimental data show that the current-induced stress developed in the individual segments within an interconnect tree is not independent [14, 29]. In other words, if we just look at the current density for each segment individually, it may appear as if all wire segments are immortal, but the whole interconnect tree could still be mortal. The reason is that the stress in one segment of an interconnect tree depends on other segments [28]. As discussed before, this issue has been resolved by the recently proposed fast EM immortality check method for general multi-segment interconnect wires [23].

In this section, we introduce the EM immortality constrained power grid wire-sizing optimization method considering multi-segment interconnect wires. It can be noticed that the new EM constraint will ensure that all the wires are EM immortal, so we call this method EM immortal power supply optimization.

5.1 Problem Formulation

Let $G = \{N, B\}$ be a P/G network with n nodes $N = \{1, \ldots, n\}$ and b branches $B = \{1, \ldots, b\}$. Each branch i in B connects two nodes i_1 and i_2 with current flowing from i_1 to i_2. l_i and w_i are the length and width of branch i, respectively. ρ is the sheet resistivity. The resistance r_i of branch i is

$$r_i = \frac{V_{i_1} - V_{i_2}}{I_i} = \rho \frac{l_i}{w_i} \tag{19}$$

5.1.1 Objective Function

The total routing area of a power grid network in terms of voltages, currents, and lengths of branches can be expressed as follows

$$f(V, I) = \sum_{i \in B} l_i w_i = \sum_{i \in B} \frac{\rho I_i l_i^2}{V_{i1} - V_{i2}} \tag{20}$$

We notice that the objective function is linear for branch current variables I and nonlinear for node voltage variables V.

5.1.2 Constraints

The constraints that need to be satisfied for a reliable, working P/G network are shown as follows.

Voltage IR Drop Constraints
In order to ensure proper logic operation, the IR drop from the P/G pads to the nodes should be restricted. For each node, we must specify a threshold voltage

$$V_j > V_{\min} \text{ for power network} \tag{21}$$

where V_j is the nodal voltage and $V_{\min}$ is the minimum required voltage for the power nodes.

Minimum Width Constraints
The widths of the P/G segments are technologically limited to the minimum width allowed for the layer where the segment lies in

$$w_i = \rho \frac{l_i I_i}{V_{i1} - V_{i2}} \geq w_{i,\min} \tag{22}$$

New Electromigration Constraints for Multi-Segment Interconnects
As described before, for a multi-segment interconnect m, the EM constraint should be satisfied

$$V_{crit,EM} > V_{E,m} - V_{\text{cat},m} \tag{23}$$

where $V_{E,m}$ is the *EM voltage* for the mth interconnect tree, which is computed using Eq. (2). $V_{\text{cat},m}$ is the cathode nodal voltage of that tree. Unlike previous methods whose branch currents are monitored and used as constants, in our new method, voltages are used as constraints. Thus, only the cathode node voltage for a whole interconnect tree needs to be monitored and no other complex calculations are required.

We remark that $V_{E,m}$, which is defined in (2), is a function of both nodal voltage and total area of wires. As a result, it is a nonlinear function of the nodal voltage (as the area of a wire segment is a function of both nodal voltage and branch current as defined in the cost function (20)). But if we have the equal width constrains as shown below, then constraint (23) actually becomes a linear function of nodal voltage again. For many practical P/G networks, most wire segments in an interconnect tree indeed have the same width.

Equal Width Constraints
For typical chip layout designs, certain tree branches should have the same width. The constraint is $w_i = w_k$, which can be written as

$$\frac{V_{i1} - V_{i2}}{l_i I_i} = \frac{V_{k1} - V_{k2}}{l_k I_k} \tag{24}$$

Kirchhoff's Current Law (KCL)
For each node j, we have

$$\sum_{k \in B(j)} I_k = 0 \tag{25}$$

where $B(j)$ is the set of branches incident on node j.

5.2 New EM Immortality Constrained P/G Optimization

The power grid optimization aims to minimize objective function (20) subjected to constraints (21)–(25). It will be referred as problem P. Problem P is a constrained nonlinear optimization problem.

5.2.1 Relaxed Two-Step Sequence of Linear Programming Solution

In the aforementioned optimization problem, we notice that the newly added EM constraint (23) is still linear in terms of nodal voltage. As a result, we can follow the relaxed two-phase iterative optimization process [8, 27] and apply the sequence of linear programming technique [27] to solve the relaxed problem. Specifically, we have two phases: the voltage solving phase (P-V phase) and the current solving phase (P-I phase).

P-V Optimization Phase
In this phase, we assume that all branch currents are fixed, then the objective function can be rewritten as

$$f(V) = \sum_{i \in B} \frac{\alpha_i}{V_{i1} - V_{i2}} \tag{26}$$

where $\alpha_i = \rho I_i l_i^2$, subject to constraints (21)–(24). We further restrict the changes of nodal voltages such that their current directions do not change during the optimization process

$$\frac{V_{i1} - V_{i2}}{I_i} \geq 0 \tag{27}$$

Problem P-V is nonlinear; however, it can be converted to a sequence of linear programming problem. By taking the first-order Taylor's expansion of Eq. (26) around the initial solution V^0, the linearized objective function can be written as

$$g\left(V\right)=\sum_{i\in B}\frac{2\alpha_i}{V_{i1}^0-V_{i2}^0}-\sum_{i\in B}\frac{\alpha_i}{\left(V_{i1}^0-V_{i2}^0\right)^2}\left(V_{i1}-V_{i2}\right) \tag{28}$$

Besides, an additional constraint will be added [27]

$$\xi\operatorname{sign}(I_i)\left(V_{i1}^0-V_{i2}^0\right)\leq\operatorname{sign}((I_i)\left(V_{i1}-V_{i2}\right) \tag{29}$$

where $\xi \in (0, 1)$ is a restriction factor, which will be selected by some trials and experience and $\operatorname{sign}(x)$ is the sign function.

Now, the procedure for solving problem *P-V* is transformed to the problem of repeatedly choosing ξ and minimizing $g\left(V\right)$ until the optimal solution is found. Theoretically, given $g\left(V_m\right) < g\left(V_{m-1}\right)$, there always exists a ξ such that $f\left(V_m\right) < f\left(V_{m-1}\right)$; however, one-dimensional line search method is a more efficient way to find the solution point. Specifically, given V_m and V_{m-1}, the search direction can be defined as $d_m = V_m - V_{m-1}$. Line search finds an $\alpha \in [0, 1]$ such that

$$f\left(\alpha d_m+V_{m-1}\right)<f\left(V_{m-1}\right) \tag{30}$$

$\alpha d_m + V_{m-1}$ becomes new V_m for the next iteration.

P-I Optimization Phase
In this phase, we assume that all nodal voltages are fixed, so the objective function becomes

$$f(I)=\sum_{i\in B}\beta_i I_i \tag{31}$$

where $\beta_i = \dfrac{\rho l_i^2}{V_{i1}-V_{i2}}$, subject to constraints (22), (24), and (25). Similarly, we restrict the changes of current directions during the optimization process

$$\frac{I_i}{V_{i1}-V_{i2}}\geq 0 \tag{32}$$

As can be seen, problem *P-I* is a linear programming problem.

5.2.2 New EM Immortality Constrained P/G Optimization Algorithm

The new EM immortality constrained P/G optimization starts with an initial feasible solution. We iteratively solve *P-V* and *P-I*. The global minimum of convex problem *P-V* will be achieved by performing several linear programming processes iteratively. The entire EM immortality constrained power grid network optimization procedure is summarized as Algorithm 1.

Algorithm 1 New EM immortality constrained P/G wire-sizing algorithm

```
Input: Spice netlist $G_I$ containing a P/G network.
Output: Optimized P/G network parameters.
 1: /*Problem Setup*/
 2: $k := 0$.
 3: Compute the initial $V^k$, $I^k$ from $G_I$.
 4: repeat
 5:     /*P-V Phase*/
 6:     Construct constraints (22), (23), (24), (27) and (29) with $I^k$.
 7:     $m := 1$.
 8:     Compute $V_m^k := \arg\min g\left(V^k\right)$ subject to constraints (21), (22), (23), (24), (27) and (29).
 9:     while $f\left(V_m^k\right) > f\left(V_{m-1}^k\right)$ do
10:         Determine the search direction $d_m := V_{m-1}^k - V_m^k$.
11:         Choose step size $\alpha$ for line search.
12:         $V_{m+1}^k := V_m^k + \alpha d_m$.
13:         $m := m + 1$.
14:     end while
15:     $V^{k+1} := V_m^k$.
16:     /*P-I Phase*/
17:     Construct constraints (22), (24) and (32) with $V^{k+1}$.
18:     Compute $I^{k+1} := \arg\min f\left(I^k\right)$ subject to (22), (24), (25), and (32) constraints.
19:     $k := k + 1$.
20: until $\left|f\left(V^k, I^k\right) - f\left(V^{k-1}, I^{k-1}\right)\right| < \varepsilon$
21: Return $f(V, I)$.
```

In practice, only a few linear programmings are needed to reach the optimum solution. Thus the time complexity of our method is proportional to the complexity of linear programming.

6 EM Lifetime Constrained Optimization

In the previous sections, we discussed the power grid sizing optimization ensuring none of the interconnect trees fails based on the voltage-based EM immortality check. However, such EM constraint may be too conservative because in reality, some wires can be allowed to have EM failure as long as the power grid network is still functional (its IR drop is still less than the given threshold) at the target lifetime (e.g., 10 years).

6.1 New EM Lifetime Constrained Optimization Flow

In this section, we propose a new EM lifetime constrained P/G wire sizing optimization method in which some segments of multi-segment interconnect wires will be allowed to fail or to age. The impacts of these segments in terms of resistance change or even wire openings will be explicitly considered and modeled. Such

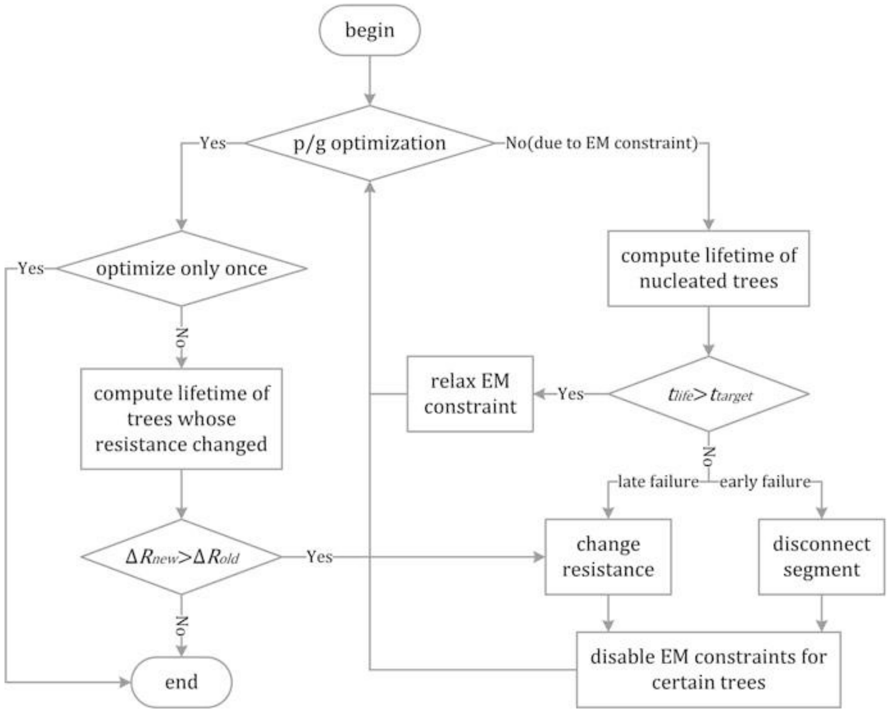


Fig. 7 Flowchart of the EM lifetime constrained P/G optimization process

aging-aware EM optimization essentially takes the EM aging-induced impacts or guard bands into account so that the designed P/G networks can still function nominally during the target lifetime. In this work, we only consider void formation, which is the dominant EM failure effect and will lead to an increase in resistance.

The new optimization flow is shown in Fig. 7. In this new flow, we first check whether a given power supply network can be optimized using Algorithm 1. If the optimization fails due to EM constraint, then the lifetime of all the interconnect trees will be computed based on the EM lifetime estimation method. We have several scenarios to discuss before we perform the optimization again. Let us define $t_{\text{life},m}$ as the lifetime of the mth interconnect tree and t_{target} as the target lifetime.

If $V_{E,m} - V_{cat,m} > V_{crit,EM}$ *and* $t_{life,m} < t_{target}$
The mth interconnect wire will be marked as a *failed wire*. Then we have the following changes for the wire before the next round of optimization.

If it is an early failure case, the cathode node of the wire segment connected by the failed via will be disconnected, which is called *wire disconnection*. The failure cases will depend on the current directions around the cathode node. Also the disconnection will depend on whether the void growth can eventually reach the critical void size or not as discussed in Sect. 4.1.

If it is a late failure case, the wire segment associated with the cathode node will have a *resistance change*. The specific resistance change for each failed segment will be calculated based on the target lifetime using our EM lifetime estimation method.

If an interconnect tree is marked as failed, then its EM constraint will be disabled as we do not need to consider its immortality anymore.

If $V_{E,m} - V_{cat,m} > V_{crit,EM}$ *and* $t_{life,m} > t_{target}$
The lifetime of interconnect wire still meets the target lifetime even though it will have void nucleation and resistance change. This also includes the case in which void growth saturates before its size reaches the critical void size. The wire still works since the current can flow through the barrier layer.

The existing $V_{E,m} - V_{\text{cat},m}$ value is used as the new EM constraint (defined as $V_{E,m,\text{next}} - V_{\text{cat},m,\text{next}}$) for the mth wire only: $V_{E,m} - V_{\text{cat},m} < V_{E,m,\text{next}} - V_{\text{cat},m,\text{next}}$. This is called *constraint relaxation*. The rational behind it is that we expect the EM status of this wire to become worse during the next optimization so its lifetime will not change too much and still meet the given lifetime after the follow-up optimizations.

After *resistance change*, or *wire disconnection*, or *constraint relaxation*, a new round of SLP programming optimization, which is similar to Algorithm 1, is carried out.

7 Summary

In this chapter, a new P/G network sizing technique is presented, which is based on a voltage-based EM immortality check method for general multi-segment interconnect wires and a physics-based EM assessment technique for fast time to failure analysis. The new P/G optimization problem subject to the voltage IR drop and new EM constraints can still be formulated as an efficient sequence of linear programming problem, and will ensure that none of the wires fails if all the constraints are satisfied. To mitigate the overly conservative nature of the optimization formulation, the EM-induced aging effects on power supply networks for a target lifetime are further considered and an EM lifetime constrained optimization method is demonstrated, which allows some short-lifetime wires to fail and optimizes the rest of the wires. The new methods can effectively reduce the area of the power grid networks while ensuring reliability in terms of immortality or target lifetime, which is not the case for the existing current density constrained P/G optimization methods.

Acknowledgments This chapter is supported in part by NSF grant under No. CCF-1527324, and in part by NSF grant under No. CCF-1816361, and in part by NSF grant under No. OISE-1854276 in part by DARPA grant under No. HR0011-16-2-0009. We also thank Prof. Naehyuck Chang of KAIST for some suggestions to improve presentation of this work.

References

1. Alam, S.M., Gan, C.L., Thompson, C.V., Troxel, D.E.: Reliability computer-aided design tool for full-chip electromigration analysis and comparison with different interconnect metallizations. Microelectr. J. **38**(4), 463–473 (2007)
2. Bakoglu, H.B.: Circuits, Interconnections, and Packaging for VLSI. Addison-Wesley, Massachusetts (1990)
3. Black, J.R.: Electromigration-A brief survey and some recent results. IEEE Trans. Electr. Devices **16**(4), 338–347 (1969)
4. Blech, I.A. (1976) Electromigration in thin aluminum films on titanium nitride. J. Appl. Phys. **47**(4), 1203–1208
5. Chatterjee, S., Sukharev, V., Najm, F.N.: Power grid electromigration checking using physics-based models. IEEE Trans. Comput. Aided Design Integr. Circuits Syst. **37**(7), 1317–1330 (2018)
6. Chen, F., Bravo, O., Chanda, K., McLaughlin, P., Sullivan, T., Gill, J., Lloyd, J., Kontra, R., Aitken, T.: A comprehensive study of low-k SiCOH TDDB phenomena and its reliability lifetime model development. In: 2006 IEEE International Reliability Physics Symposium Proceedings, pp. 46–53. IEEE, Piscataway (2006)
7. Chen, H.B., Tan, S.X.D., Huang, X., Kim, T., Sukharev, V.: Analytical modeling and characterization of electromigration effects for multibranch interconnect trees. IEEE Trans. Comput. Aided Design Integr. Circuits Syst. **35**(11), 1811–1824 (2016)
8. Chowdhury, S.: Optimum design of reliable IC power networks having general graph topologies. In: Proceedings of the 1989 26th ACM/IEEE Design Automation Conference (DAC), pp. 787–790. IEEE, Piscataway (1989)
9. Chowdhury, S.U., Breuer, M.A.: Minimal area design of power/ground nets having graph topologies. IEEE Trans. Circuits Syst. **34**(12), 1441–1451 (1987)
10. Chowdhury, S., Breuer, M.A.: Optimum design of IC power/ground nets subject to reliability constraints. IEEE Trans. Comput. Aided Design Integr. Circuits Syst. **7**(7), 787–796 (1988)
11. Cook, C., Sun, Z., Kim, T., Tan, S.X.D.: Finite difference method for electromigration analysis of multi-branch interconnects. In: Proceedings of the 2016 13th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD), pp. 1–4. IEEE, Piscataway (2016)
12. Cook, C., Sun, Z., Demircan, E., Shroff, M.D., Tan, S.X.D.: Fast electromigration stress evolution analysis for interconnect trees using Krylov subspace method. IEEE Trans. Very Large Scale Integr. Syst. **26**(5), 969–980 (2018)
13. Dutta, R., Marek-Sadowska, M.: Automatic sizing of power/ground (P/G) networks in VLSI. In: Proceedings of the 1989 26th ACM/IEEE Design Automation Conference (DAC), pp. 783–786. IEEE, Piscataway (1989)
14. Hau-Riege, S.P., Thompson, C.V.: Experimental characterization and modeling of the reliability of interconnect trees. J. Appl. Phys. **89**(1), 601–609 (2001)
15. Hu, C.K., Small, M.B., Ho, P.S.:Electromigration in Al (Cu) two-level structures: effect of Cu and kinetics of damage formation. J. Appl. Phys. **74**(2), 969–978 (1993)
16. Hu, C.K., Anaperi, D., Chen, S.T., Gignac, L.M., Herbst, B., Kaldor, S., Krishnan, M., Liniger, E., Rath, D.L., Restaino, D., Rosenberg, R., Rubino, J., Seo, S.C., Simon, A., Smith, S., Tseng, W.T.: Effects of overlayers on electromigration reliability improvement for Cu/low k interconnects. In: 2004 IEEE International Reliability Physics Symposium Proceedings, pp. 222–228. IEEE, Piscataway (2004)
17. Huang, X., Yu, T., Sukharev, V., Tan, S.X.D.: Physics-based electromigration assessment for power grid networks. In: Proceedings of the 2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC), pp. 1–6. IEEE, Piscataway (2014)

18. Huang, X., Kteyan, A., Tan, S.X.D., Sukharev, V.: Physics-based electromigration models and full-chip assessment for power grid networks. IEEE Trans. Computer-Aided Design Integr. Circuits Syst. **35**(11), 1848–1861 (2016)
19. ITRS: International Technology Roadmap for Semiconductors (ITRS) Interconnect, 2015 edition (2015). http://public.itrs.net
20. Korhonen, M.A., Bo/rgesen, P., Tu, K.N., Li, C.Y.: Stress evolution due to electromigration in confined metal lines. J. Appl. Phys. **73**(8), 3790–3799 (1993)
21. Lienig, J., Thiele, M.: Fundamentals of Electromigration-Aware Integrated Circuit Design. Springer, Berlin (2018)
22. Nassif, S.R.: Power grid analysis benchmarks. In: Proceedings of the 2008 Asia and South Pacific Design Automation Conference (ASP-DAC), pp. 376–381. IEEE, Piscataway (2008)
23. Sun, Z., Demircan, E., Shroff, M.D., Kim, T., Huang, X., Tan, S.X.D.: Voltage-based electromigration immortality check for general multi-branch interconnects. In: Proceedings of the 2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), pp. 1–7. IEEE, Piscataway (2016)
24. Sun, Z., Demircan, E., Shroff, M.D., Cook, C., Tan, S.X.D.: Fast electromigration immortality analysis for multisegment copper interconnect wires. IEEE Trans. Comput. Aided Design Integr. Circuits Syst. **37**(12), 3137–3150 (2018)
25. Sun, Z., Sadiqbatcha, S., Zhao, H., Tan, S.X.D.: Accelerating electromigration aging for fast failure detection for nanometer ICs. In: Proceedings of the 2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC), pp. 623–630. IEEE, Piscataway (2018)
26. Tan, S.X.D., Shi, C.J.R.: Efficient very large scale integration power/ground network sizing based on equivalent circuit modeling. IEEE Trans. Comput. Aided Design Integr. Circuits Syst. **22**(3), 277–284 (2003)
27. Tan, S.X.D., Shi, C.J.R., Lee, J.C.: Reliability-constrained area optimization of VLSI power/-ground networks via sequence of linear programmings. IEEE Trans. Comput. Aided Design Integr. Circuits Syst. **22**(12), 1678–1684 (2003)
28. Tan, S.X.D., Amrouch, H., Kim, T., Sun, Z., Cook, C., Henkel, J.: Recent advances in EM and BTI induced reliability modeling, analysis and optimization. Integr. VLSI J. **60**, 132–152 (2018)
29. Thompson, C.V., Hau-Riege, S.P., Andleigh, V.K.: Modeling and experimental characterization of electromigration in interconnect trees. In: AIP Conference Proceedings, AIP, vol. 491, pp. 62–73 (1999)
30. Verbruggen, A.H., van den Homberg, M.J.C., Jacobs, L.C., Kalkman, A.J., Kraayeveld, J.R., Radelaar, S.: Resistance changes induced by the formation of a single void/hillock during electromigration. In: AIP Conference Proceedings, AIP, vol. 418, pp. 135–146 (1998)
31. Wang, K., Marek-Sadowska, M.: On-chip power-supply network optimization using multigrid-based technique. IEEE Trans. Comput. Aided Design Integr. Circuits Syst. **24**(3), 407–417 (2005)
32. Wang, X., Wang, H., He, J., Tan, S.X.D., Cai, Y., Yang, S.: Physics-based electromigration modeling and assessment for multi-segment interconnects in power grid networks. In: Proceedings of the 2017 Design, Automation and Test in Europe Conference and Exhibition (DATE), pp. 1727–1732. IEEE, Piscataway (2017)

33. Wang, X., Yan, Y., He, J., Tan, S.X.D., Cook, C., Yang, S.: Fast physics-based electromigration analysis for multi-branch interconnect trees. In: Proceedings of the 2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), pp. 169–176. IEEE, Piscataway (2017)
34. Zhang, L.: Effects of scaling and grain structure on electromigration reliability of cu interconnects. PhD Thesis, University of Texas at Austin (2010)
35. Zhou, H., Sun, Y., Sun, Z., Zhao, H., Tan, S.X.D.: Electromigration-lifetime constrained power grid optimization considering multi-segment interconnect wires. In: Proceedings of the 2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC), pp. 399–404. IEEE, Piscataway (2018)
36. Zhou, H., Sun, Z., Sadiqbatcha, S., Chang, N., Tan, S.X.D.: EM-aware and lifetime-constrained optimization for multisegment power grid networks. IEEE Trans. Very Large Scale Integr. Syst. **27**(4), 940–953 (2019)

Monitor Circuits for Cross-Layer Resiliency

Mahfuzul Islam and Hidetoshi Onodera

1 Introduction

The end of supply voltage scaling has pushed circuit designers to find for new solutions to reduce power consumption. One key reason for the stoppage of supply scaling is variability including aging. The International Technology Roadmap for Semiconductors (ITRS) highlights performance variability and reliability management in the next decade as a red brick (i.e., a problem with no known solutions) for the design of computing hardware [1]. Instead of operating under predefined supply voltage and clock frequency, the circuit must adapt itself according to its process conditions, as well as to the dynamic changes of temperature, aging, and workload to harness the full potential of technology scaling. With the resilient operations, a chip's lifetime can be extended, and energy consumption can be reduced.

Due to significant variations in temperature, workload, and aging, dynamic tuning of not only the supply voltage and clock frequency but also the threshold voltages has become a necessity for energy-efficient operation. However, without knowing the device and environmental parameters, tuning of these parameters is not possible. On-chip monitor circuits which provide the information about device and environment come to play an important role. On-chip monitors realize an interface between hardware and software, which then can be utilized for software-controlled optimization. The future LSI (Large Scale Integration) chip will require lots of monitors to track transistor performances, temperature changes, supply voltage droops, and leakage current variations. This chapter describes some design techniques of monitor circuits based on delay cells and then presents a reconfigurable monitor

M. Islam (✉) · H. Onodera
Kyoto University, Kyoto, Japan
e-mail: islam.akmmahfuzul.3w@kyoto-u.ac.jp; onodera.hidetoshi.4x@kyoto-u.ac.jp

© The Author(s) 2021
J. Henkel, N. Dutt (eds.), *Dependable Embedded Systems*, Embedded Systems,
https://doi.org/10.1007/978-3-030-52017-5_16

architecture to realize different delay characteristics with a small area footprint. An extraction methodology of physical parameters from a set of monitor circuits is presented for model-hardware correlation.

2 Cross-Layer Resiliency

This section describes the benefit of realizing cross-layer resiliency by dynamic tuning of threshold voltage, supply voltage, and clock frequency. Cross-layer resiliency enables energy-efficient operation by eliminating excessive margins. We highlight the importance of run-time sensing of circuit delay, leakage current, switching power, temperature, and threshold voltage to realize minimum energy operation under process, voltage, temperature, activity and temperature variations. Multiple on-chip monitor circuits are required to sense these parameters. Although monitor circuits are not a part of the actual circuit, they are essential components for run-time tuning.

2.1 Parameter Fluctuation and Aging

Variations in physical parameters such as transistor threshold voltage, and temperature have spatial distributions over a chip with both of the random and systematic components. Besides the physical parameter variations, environmental variations also affect circuit performance significantly. Temperature variations of more than 50 °C between different parts within a chip are reported [2]. Increase in temperature degrades circuit performance and increases leakage power. According to ITRS, supply voltage fluctuation is considered to be ±10% of the nominal voltage. Sudden drop of supply voltage may cause critical timing failure causing system malfunctioning. Because of process variation, some chips can be slow and some chips can be fast. Fast chips tend to be leaky causing larger energy consumption. Designers thus face a challenge to meet both of the delay and power constraints, since the circuit needs to operate correctly under all of the variation scenarios.

Device characteristics also degrade over time. Aging causes reliability issues where high temperature accelerates device aging. Device phenomena such as Negative Bias Temperature Instability (NBTI) is reported to cause 10% of delay degradation in digital circuits for a 70-nm process over 10 years [3]. Designing the circuit for the worst possible scenario is energy inefficient as it increases area, power, and cost. A chip may face extreme worst-case scenarios once in several years. The conventional worst-case design methodology, where the operating conditions of a circuit are set such as to meet the worst-case performance, is way too energy inefficient and new design paradigm incorporating on-chip monitor circuits have become indispensable. In the new design paradigm, parameters such as the supply voltage and threshold voltage are tuned in the run-time such that the target delay and power profile are achieved. As a result, instead of worrying for the worst-case performance, the circuit can now be designed to achieve optimal performances.

2.2 *Cross-Layer Resiliency for Energy-Efficient Operation*

Figure 1 shows a typical design hierarchy of a system-on-a-chip. First, transistor models for a target process technology node are given to circuit designers. These transistor models contain statistical models to simulate the effects of variations on circuit performance. To guarantee error-free circuit operation, a circuit is tested for extreme cases by using the assumed models. As a result, the circuits tend to be over-designed which result in excessive energy consumption. From a system perspective, the circuits need to operate at different supply voltages and clock frequencies while ensuring correct operations. The selection of adequate clock frequency and supply voltage is performed pessimistically. Design-time optimization is an open-loop operation; thus the operating conditions are set for the worst-cases. The solution obviously is to create a feedback loop into the system which can only be realized by tuning circuit parameters in the run-time. Run-time tuning relaxes the design constraints on the circuit and as a result the circuit become better optimized compared with the one where no run-time tuning is performed.

Figures 2 and 3 show two profiles of energy consumption for an LSI. Figure 2 shows simulated energy and frequency contour plots on the threshold voltage (V_{th}) and the supply voltage (V_{dd}) plane for a model circuit operating at an activity rate of 1%. The model circuit used here is a delay line of 40 inverter cells. A commercial

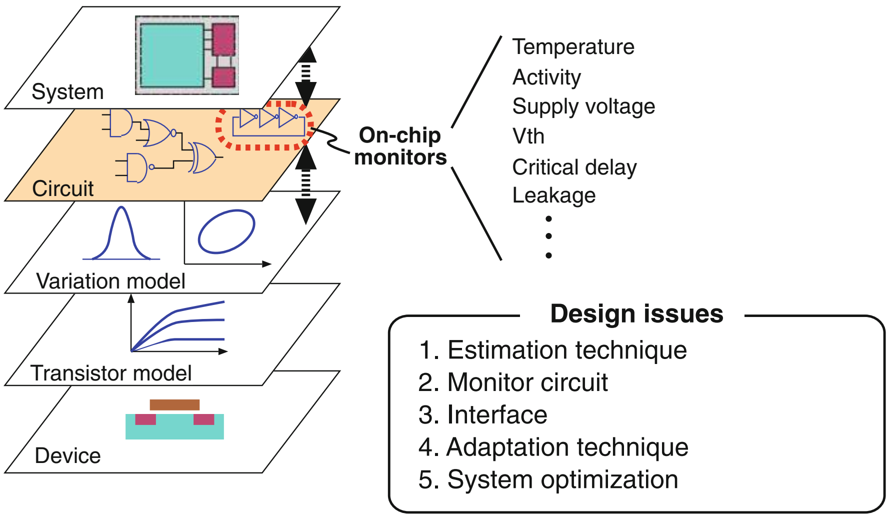


Fig. 1 Cross-layer optimization with the use of monitor circuits

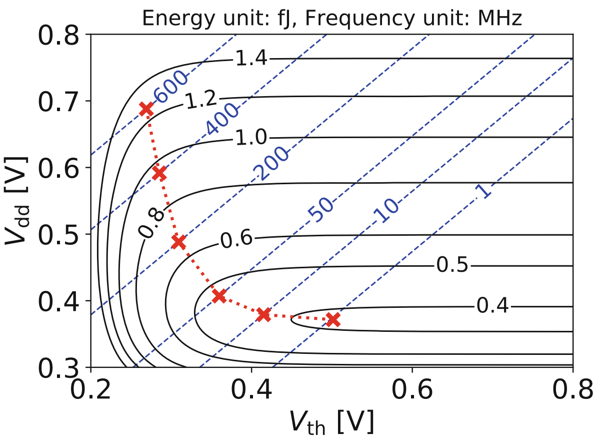


Fig. 2 Energy and frequency contour plot on the V_{th} and V_{dd} plane. Activity rate of 0.01 is assumed

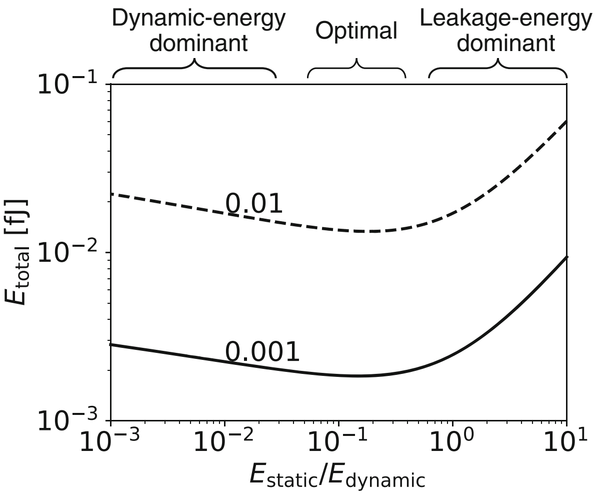


Fig. 3 Total energy per clock cycle against the ratio between static and dynamic energy for a clock frequency of 100 MHz. Having a balanced static and dynamic energy is the key to minimum energy operation

65 nm process is assumed here. Cross points in the plot show the sets of V_{th} and V_{dd} values that give the minimum energy operation for each operating frequency. We observe that the required V_{th} and V_{dd} values, that realize the minimum energy operation, differ significantly with the changes in the clock frequency. Dynamic adaptation of V_{th} and V_{dd} values ensures minimum energy operation for any operating frequency. Figure 3 shows the total energy of the circuit operating at 100 MHz under different combinations of V_{th} and V_{dd} against the ratio of static energy (E_{static}) to dynamic energy ($E_{dynamic}$). We observe that a ratio of 10 to 50% realizes near minimum energy operation. Under the variations of circuit activity, operating frequency and temperature, the energy ratio varies largely. To ensure minimum energy operation, V_{dd} and V_{th} values need to be tuned such that a ratio between 10 and 50% is realized. From the figures, the need for run-time tuning of V_{dd} and V_{th} values are apparent but the problem is how to realize such a mechanism.

Two key mechanisms are required to realize a feedback system. One is the sensing mechanism of the output. The other is to feedback the output to the input of the system. Sensing mechanism is an essential component here. In the case of an LSI, the output parameters are the V_{th} values, circuit delays, temperature, leakage current, and switching current. Sensing these parameters requires multiple on-chip monitor circuits. The monitors provide real-time information of the hardware which can then be used to set the parameters of V_{dd}, V_{th} and clock frequency optimally for reliable operation.

2.3 *Role of Monitor Circuits*

The past trend of using smaller transistors to achieve higher operating frequency has come to an end [4]. Instead of the clock frequency, system throughput and energy per throughput are the modern specifications for a device. The new era of LSI scaling is a system-on-a-chip (SoC) approach that combines a diverse set of components including adaptive circuits, integrated on-chip monitors, sophisticated power-management techniques, and increased parallelism to build products that are many-core, multi-core, and multi-function [5]. The ability to adapt to the changes in environment and performance will give us the full benefit of technology scaling. Tuning mechanisms and on-chip monitors are needed to realize circuits that have the ability to adapt. The future SoC must have capabilities of post-silicon self-healing, self-configuration, and error correction. Effective use of on-chip monitor circuits will play a major role in continuing the advancement of LSI. Use of on-chip monitors provides us the following advantages:

1. Reduce design margin in each layer of design hierarchy by eliminating pessimism.
2. Tune system parameters based on the actual hardware profile.
3. Provide information for silicon debugging and timing analysis.

To harness the above advantages, the following characteristics of on-chip monitor circuits are preferred:

Digital	Digital in nature realizes robust operation under different supply voltages.
Design automation	Monitor circuits for threshold voltage, temperature, supply voltage, interconnect, activity, and leakage current are required. Thus, design automation is a key factor here for low-cost implementation of the monitors. Cell-based design with delay cells are preferred.
Area efficiency	Area efficiency is an important parameter for fine-grain and distributed implementation of monitor circuits on the chip.

As the target parameters such as the temperature and leakage current are analog values, mechanisms to convert the analog values to digital values are required to interface with the other components of the system. Two design methodologies can be adopted for designing monitor circuit. One methodology performs operations in the analog domain to sense and amplify the effect of the parameter variation and then convert the analog value to a digital value. The other methodology converts the analog value to a digital value as early as possible and then make operations in the digital domain. Incorporating the analog value in the delay of a logic gate realizes the later. Furthermore, the well established cell-based design methodology for automation can be adopted readily for the delay-based implementation of monitor circuits. We therefore explore several delay-based implementations of monitor circuits in this chapter.

3 Delay-Based On-Chip Monitor Design

Delay-based monitor circuits use the mechanisms of converting the target analog value to the delay of a logic gate. The topology of the logic gate thus need to be designed such that the target parameter variation is amplified in the delay. To understand the delay-based monitoring, we first give an overview of the general delay characteristics of logic gates. Then we explore several techniques to tune the delay characteristics such that the monitoring of a target parameter can be realized. Finally, we demonstrate a cell-based design of a reconfigurable monitor circuit that can sense the parameters of nMOSFET and pMOSFET threshold voltages.

3.1 Delay Characteristics

Delay-based monitoring is based on the fact that the delay of a logic gate contains information of the transistor drain current I_d. Figure 4 shows four delay paths consisting of different logic gates and interconnects. A delay path of Fig. 4a consists of inverter gates. Delay paths of Fig. 4b and c consist of NAND2 and NOR2 gates. A delay path of Fig. 4d consists of inverter gates with long interconnecting wires. Depending on the topology of the logic gate and the interconnect length, delays of different gates and interconnect show different behavior to process, voltage, and temperature variation. Figure 5 shows the topology of four different logic gates. Figure 5a shows a conventional inverter topology. Figure 5b shows a NAND2 topology where two nMOSFETs are placed in stack. Figure 5c shows a NOR2 topology where two pMOSFETs are placed in stack. Figure 5d shows an inverter topology where two pMOSFETs and two nMOSFETs are placed in stack to mimic the delay behavior of the both of the NAND2 and NOR2 gates.

Under the presence of large within-die random variation, each delay path might behave differently. At a higher supply voltage, a particular path may show the

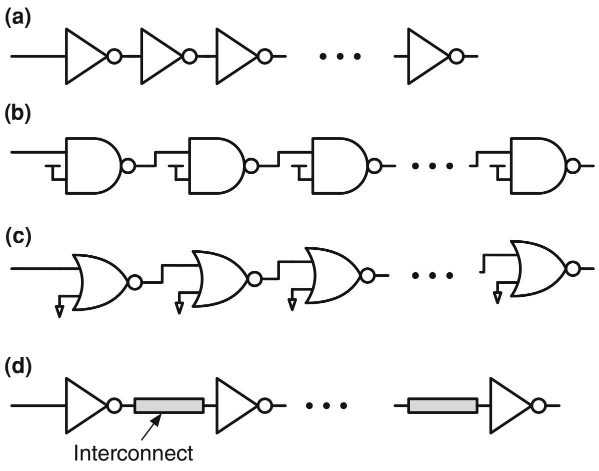


Fig. 4 Delay paths consisting of (**a**) inverter gates, (**b**) NAND2 gates, (**c**) NOR2 gates, and (**d**) inverter gates with long wires

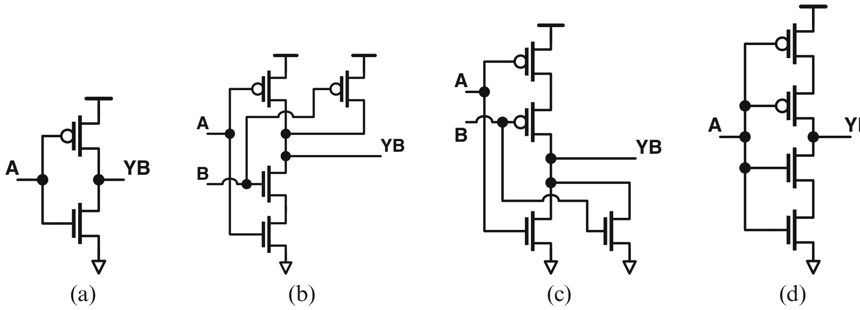


Fig. 5 Topology of different delay cells. (**a**) Inverter gate. (**b**) NAND2 gate. (**c**) NOR2 gate. (**d**) Universal delay cell

worst-case delay, whereas at a lower supply voltage, a different path may show the worst-case delay. Figure 6 shows the delay change against the change of supply voltage. Topology with a stacked transistor shows higher sensitivity to V_{dd} change than that without a stacked transistor. Topology with a reduced V_{gs} value shows much higher sensitivity to V_{dd} change. The important point is that the delays of different topologies show different sensitivities to process, supply voltage, and temperature changes. Under the presence of within-die variation, the gates of the same logic type also show different delay behavior. Thus, accurate delay estimation of a circuit is challenging. Instead, we can monitor the delay of a representative circuit that gives us a reasonable prediction of the actual delay of the circuit.

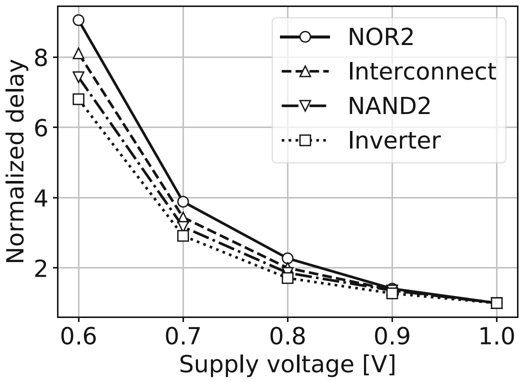


Fig. 6 Delay versus supply voltage for different inverter topologies

3.2 *Delay Model*

A delay model is useful to intuitively understand the different delay characteristics for different topology. The rise and fall delays of an inverter gate can be approximated by the following equations:

$$d_{\text{rise}} = \frac{C_{\text{load}}\, V_{\text{logic}}}{I_{\text{dp}}}, \tag{1}$$

$$d_{\text{fall}} = \frac{C_{\text{load}}\,(V_{\text{dd}} - V_{\text{logic}})}{I_{\text{dn}}}. \tag{2}$$

Here, I_{dp} and I_{dn} are the drain currents of pMOSFET and nMOSFET during the ON state, respectively. C_{load} is the load capacitance that consists of the gate capacitance of MOSFETs of the next gate, drain capacitance of pMOSFET and nMOSFET, and interconnect parasitic capacitance. V_{logic} is the logical threshold voltage at which the next gate switches its output value. To model the transistor drain current, EKV model based equation of Eq. 3 is useful to express the drain current that is continuous from weak-inversion to strong-inversion operation: [6, 7].

$$I_{\text{d}} = k \cdot \frac{W}{L} \cdot \ln^{\alpha}\left[1 + \exp\left(\frac{V_{\text{gs}} - (V_{\text{th}} - \gamma V_{\text{bs}} - \lambda V_{\text{ds}})}{\alpha\, n\, V_T}\right)\right]. \tag{3}$$

Here, k is a technology-related parameter. γ is the body bias coefficient and λ is the short-channel coefficient. Short-channel effect reduces the threshold voltage when large V_{ds} is applied to the transistor. Thus, large V_{ds} value increases ON current which is beneficial to switching delay, but causes exponential increase in the leakage current.

For the pull-down operation of an inverter gate of Fig. 5a, V_{bs} is zero and V_{ds} changes from V_{dd} to V_{logic}. However, in the case of a NAND2 gate, the values of V_{bs} and V_{ds} differ. The source of the nMOSFET that is connected to the output is not tied to ground. As a result, V_{bs} becomes negative that causes the V_{th} to increase. Consequently, the V_{ds} value remains within a small value. Smaller V_{ds} value causes less short-channel effect resulting in a higher V_{th} value than a larger V_{ds} value. As a result of negative V_{bs} value and smaller V_{ds} value, the drain current decreases which causes the delay to increase.

3.3 Delay-Based Monitor Circuits

Design of on-chip monitors requires careful choosing of the right topology. Here, we discuss several delay-based design techniques that realize monitoring of different parameters.

3.3.1 Critical Path Monitor

The first and the most important parameter to monitor is the maximum delay of a circuit to ensure that the circuit operates at a certain clock frequency without any timing error. The maximum delay of a circuit is the maximum of delays of all the paths. As a circuit consists of thousands of delay paths, we can choose the following two methods to monitor the maximum delay.

1. Monitor the delays of actual paths, and
2. Monitor the delay of a representative delay path.

The first method, which is in-situ monitoring, requires additional circuitry in the actual delay paths. In the case of in-situ monitors, the Flip-Flops (FF) in a circuit are replaced with special FFs with error detection sequential (EDS) functions. The EDS can either detect whether a timing error has occurred [8, 9] or warn us before the occurrence of actual errors [10–12]. Supply voltage and clock frequency are adapted accordingly based on the EDS signals. The drawback of EDS-based in-situ monitors is that the additional circuits add extra delays, and increase area and power. To reduce the delay and area overhead, we can replace only those FFs where the delays are critical. During the design phase, we can make a list of the potential critical delay paths. However, as shown in Fig. 6, paths show different sensitivity to process, supply and temperature changes. Thus, the number of candidates tend to increase drastically under process, voltage, and temperature variations. Another fundamental drawback to be overcame is that a critical path is not always sensitized. Thus, it is necessary to properly estimate the actual timing slack of the critical path.

The second method requires an additional delay path that is placed near the actual circuit that can track the delay of the actual circuit. This delay path is often called a critical path monitor (CPM). The requirement of such a CPM is that it tracks the maximum delay of the target circuit for all conditions of process, voltage, and temperature variations. CPM is thus a delay path that is synthesized such that it tracks the worst delay of the circuit. However, there is no universal solution on how

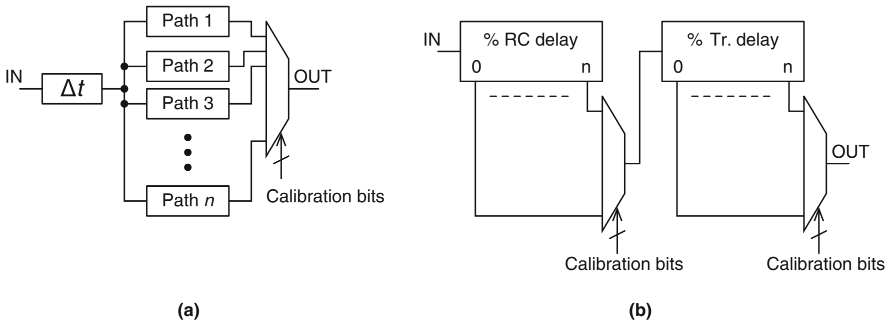


Fig. 7 Synthesis of critical delay path from a combination of series and parallel delay paths. (**a**) Parallel paths. (**b**) Series paths

to design a CPM that meets the above criterion. Two approaches have been proposed on how to synthesize a CPM. One approach is to synthesize a critical path monitor from a list of potential critical paths during the design phase [13–15]. The other approach is to design a reconfigurable delay path consisting of different logic gates and wire lengths, and then configuring the delay path during the test time, such that the delay correlates with the maximum achievable frequency [16–19]. Figure 7 shows a general concept of the synthesis framework of a critical delay path [20]. Several paths such as the paths shown in Fig. 4 are put in parallel. Then the several paths are placed in series. During the calibration process, combinations of parallel and series paths are explored to find a combination that gives the worst delay for all the operating conditions.

Instead of using a reconfigurable delay line, a general purpose delay line consisting of inverter cells with stacked transistors are also proposed so that the path mimics the worst-case delay [21]. Calibration is nonetheless required which can be performed during the design phase and during the test. To encounter the effect of systematic within-die variations, multiple CPMs can be used that are distributed at various places on the chip [15, 21].

3.3.2 Threshold Voltage Monitor

For adaptation of $V_{\rm th}$ values to their optimum values, $V_{\rm th}$ monitors are required. Although there is no universal definition of $V_{\rm th}$, an arbitrary definition can be used as a reference. For example, the $V_{\rm gs}$ value that gives a fixed $I_{\rm d}$ value is often used to define the $V_{\rm th}$ value. Conversely, we can track the $V_{\rm th}$ value by observing the change of $I_{\rm d}$ value if the $V_{\rm gs}$ can be set as a function of $V_{\rm th}$. Then the delay change resulting from the $I_{\rm d}$ change can be measured and converted to digital with the use of a reference clock signal. Figures 8 and 9 show two delay cells consisting of inverter gates where either the nMOSFET $V_{\rm gs}$ or the pMOSFET $V_{\rm gs}$ voltage becomes a function of the corresponding $V_{\rm th}$ values ($V_{\rm thp}$ for pMOSFET and $V_{\rm thn}$ for

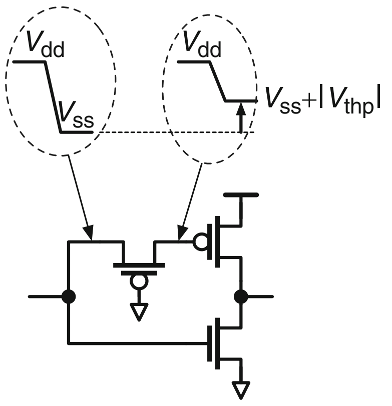


Fig. 8 V_{thp}-dominant delay cell for V_{thp} monitoring

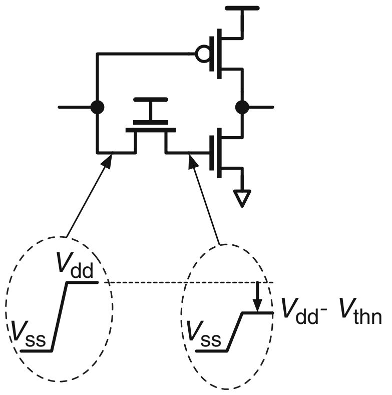


Fig. 9 V_{thn}-dominant delay cell for V_{thn} monitoring

nMOSFET). The V_{th}-sensitive gate-source voltage is realized using pass-transistors as shown in Figs. 8 and 9 [22, 23]. To illustrate the V_{th} monitoring capability of the monitor cells, sensitivity vectors of different inverter topologies are shown in Fig. 10 at nominal supply voltage for a 65 nm bulk process. Here, the sensitivity vector consists of the sensitivity coefficients of the delay to V_{thn} and V_{thp} changes. We observe that the sensitivity coefficients of the pass-transistor inserted cells are multiple times larger than those of conventional inverter, NAND2, and NOR2 cells. An all-digital process variability monitor based on a shared structure of a buffer ring and a ring oscillator is proposed in [24]. The technique utilizes the differences of rise and fall delays of inverter gates because of process variations.

As will be shown next, driving the load with the transistor leakage current also gives us a delay that is exponentially related to V_{th} value change. However, driving the load using leakage current requires careful design because leakage currents through the pull-up and the pull-down paths get involved also. Gate-leakage current is also a factor to degrade the accuracy of such monitors. The topologies of Figs. 8 and 9 give us compact designs that are minimal and fulfill the purpose.

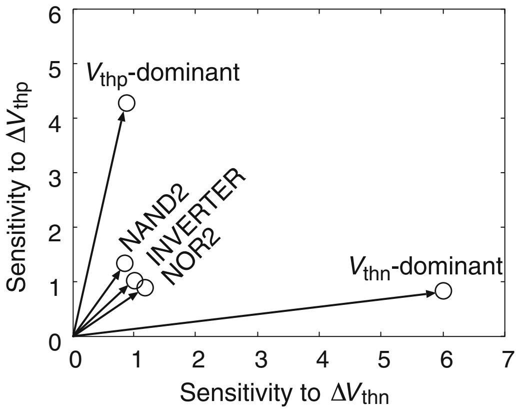


Fig. 10 Sensitivity vectors of different inverter topologies

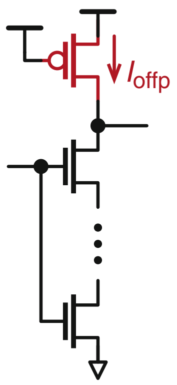


Fig. 11 I_{offp}-dominated delay cell

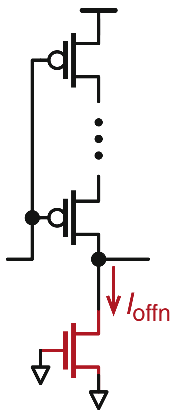
3.3.3 Aging Monitor

A critical path monitor also acts as an aging monitor. However, the differences in activity rate may cause deviations in the aging between an actual critical path and a monitor path. Therefore, delay paths of different activity rates can be implemented to track aging. Multiple delay lines consisting of inverter gates with different activity would give us precise aging information. Decoupling the NBTI and PBTI effects can be useful for debugging and modeling purposes. In that case, different architectures are proposed for independent NBTI and PBTI monitoring [25].

3.3.4 Sub-threshold Leakage Monitor

Sub-threshold leakage monitor helps us to estimate the leakage current of a circuit. The information can then be used to tune V_{th}, V_{dd} or frequency optimally. Figure 11 shows a delay cell whose rise delay is several orders of magnitude larger than the fall delay. The rise delay is driven by the pMOSFET OFF current, while the fall

Fig. 12 I_{offn}-dominated delay cell



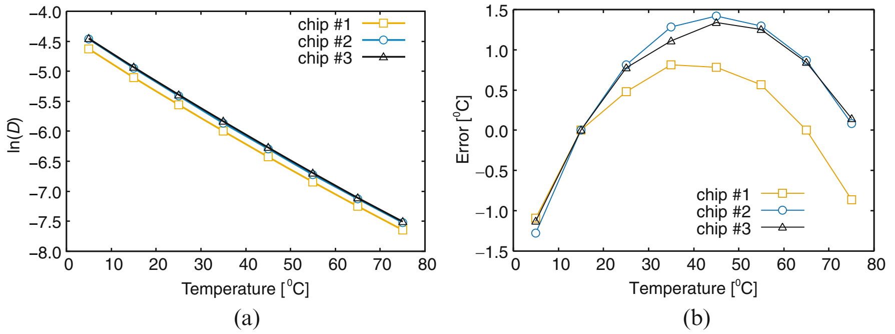


Fig. 13 Temperature monitoring utilizing inverter delay driven by nMOSFET OFF current. (**a**) Logarithm of oscillation period driven by nMOSFET OFF current against temperature. (**b**) Monitoring error against temperature after an one-point calibration

delay is driven by the nMOSFET ON current. As a result, the delay of a path consisting of this cell is proportional to the pMOSFET OFF current. Similarly, delay of a path consisting of delay cells of Fig. 12 is proportional to the nMOSFET OFF current. Figure 13a shows the change of measured oscillation period for a delay path consisting of 125 inverter stages against the temperature change. The inverter topology of Fig. 12 is used here. The target process is a 65 nm bulk process. The oscillation period here corresponds to the average OFF current of 125 nMOSFETs. The logarithm of the delay changes linearly with the temperature showing that the monitor tracks the leakage current change correctly.

3.3.5 Temperature Monitor

As leakage current is sensitive to temperature variation, a leakage current monitor can be used for on-chip temperature monitoring. The logarithm of the oscillation period, D, can be expressed by the following equation:

$$\ln(D) = a_T + b_T \cdot T, \qquad (4)$$

where T is the absolute temperature, a_T and b_T are temperature coefficients. Figure 13b shows the monitoring error after a one-point calibration for a 65 nm bulk process. Calibration is performed at 15 °C. An error range of −1.3 °C to 1.4 °C is observed. The above error range is small enough for real-time thermal and reliability management.

3.3.6 Supply Voltage Monitor

Supply voltage fluctuation has always been a concern which is getting more severe with the reduction of supply voltage. Supply voltage fluctuation has a static component which results from the power delivery network (PDN) and a dynamic component which is the result of transition from idle to active state of a circuit. As critical path monitors are also sensitive to supply voltage fluctuations and have a high bandwidth, they can also detect dynamic supply voltage fluctuations [17]. In the case of CPMs, the output is the timing information obtained by comparing the path delay and clock period. Thus, the error information does not give whether the error is from temperature or supply voltage for example. However, when combined with other monitors such as temperature and threshold voltage, identification of the causes of timing error becomes possible. The identification of the sources of timing error allows correct optimization and lifetime enhancement. On-chip supply voltage droop monitoring mechanisms have been proposed to evaluate the power delivery network (PDN) [26].

3.3.7 Activity Monitor

Run-time estimation of the static and the dynamic energy can be used to achieve the minimum energy operation as suggested by Fig. 3. As the dynamic energy is proportional to circuit activity rate, we can estimate the dynamic energy by calculating the activity rate of a circuit. A digital dynamic power meter (DDPM) has been used that computes a rolling average of signal activity over a fixed number of clock cycles [27]. The accuracy of the power estimation here depends on careful selection of signals, such that they correspond to the activity of structures that have high power consumption. Instead of monitoring key logic signals, a clock activity adder (CAA) for switching power estimation is also proposed [28]. The approach of the CAA takes advantage of the fact that switching power is highly correlated to register clock activity. Similarly, hardware-event monitors such as memory-access counters and instruction-execution counters can be used for dynamic energy estimation [29]. These monitors depend on counting signal transitions rather than the delay itself.

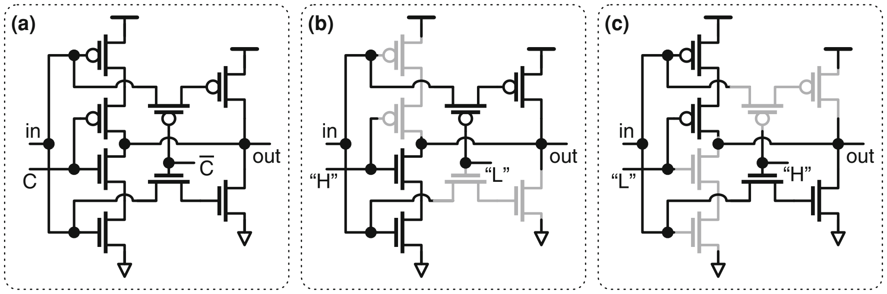


Fig. 14 A reconfigurable inverter cell topology for V_{thp} and V_{thn} monitoring. "C" is a control signal. (**a**) Reconfigurable topology. (**b**) V_{thp}-sensitive configuration, and (**c**) V_{thn}-sensitive configuration

3.4 Reconfigurable Delay Path for Multiple Parameter Monitoring

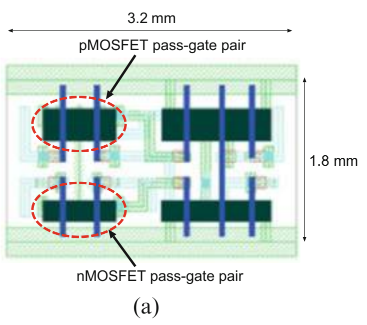
Delay-based sensing enables us to design a reconfigurable architecture to monitor multiple parameters by configuring the delay path accordingly [30, 31]. For example, we can use the topology of Fig. 14a to monitor both of the V_{thp} and V_{thn} variations. Figure 14b and c shows the two configurations to make the delay V_{thp}- and V_{thn}-sensitive, respectively.

3.5 Cell-Based Design

The use of delay cells provides the advantage of the use of cell-based design flow that enables us to place and distribute the monitors into different parts of the chip. For example, temperature monitors need to be placed at hot-spots where power density is high. Power density maps are generated during the design phase. A cell-based design example in a 65 nm bulk triple-well process for a reconfigurable V_{th} monitoring circuit is shown in Fig. 15. The cells with green highlights in Fig. 15b are the monitor cells of Fig. 15a. The placements of the cells are performed carefully utilizing the "do not touch" and "relative adjacent placement" features of the place and route tool.

3.6 On-Chip Measurement and System Interface

The monitoring circuit needs to be interfaced with system for adaptation and self-tuning. The following three mechanisms can be adopted for on-chip measurement of monitor circuits.



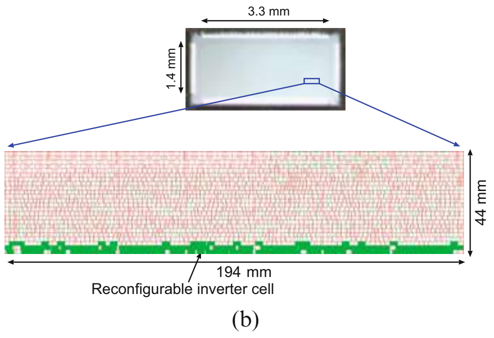


Fig. 15 Delay characteristics for different topology and supply voltage. (**a**) Cell layout of a reconfigurable V_{th} monitor delay cell. (**b**) Chip micrograph and layout of a reconfigurable monitor circuit including the controller

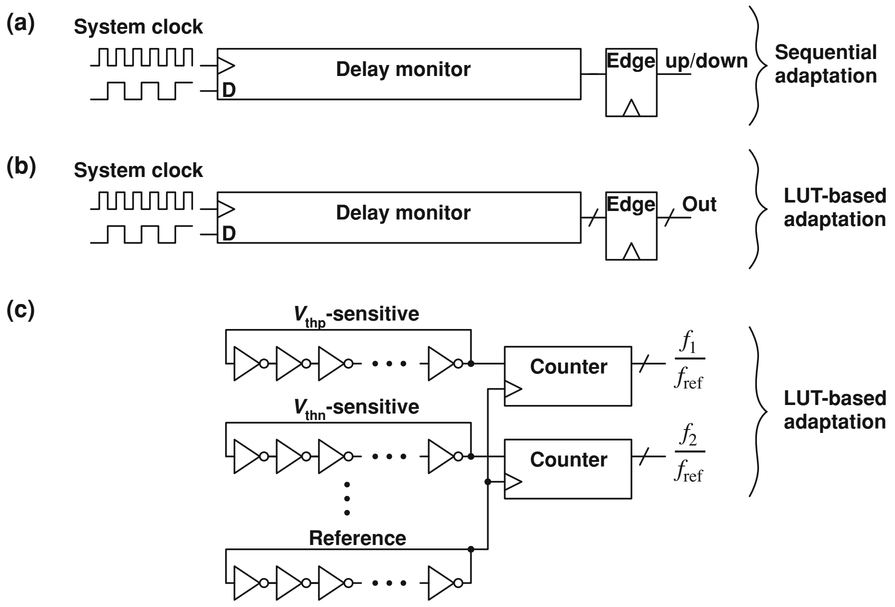


Fig. 16 Three different measurement methods for system interfacing. (**a**) EDS-based method. (**b**) Time-to-digital conversion based method, and (**c**) Frequency ratio based method

1. Edge detection [16, 17, 32].
2. Frequency counting [33].

Edge detection based system can have either a single bit output [16] or multiple bits output [17, 32]. Figure 16 shows three different methods for digitizing the monitored delay. Figure 16a checks whether the delay is smaller or larger than the system clock period [17, 32]. If the delay is smaller, adaptation such as slowing down the system by reducing the supply voltage can be performed. If the delay is larger, the system will speed up by increasing the supply voltage for example. To ensure that the transitions occur without any timing error, margins are added in the delay path. These margins include within-die random delay effects as well as the response time of the adaptation. A resolution window can also be added to ensure that the adaptation occurs without inducing any timing error.

Figure 16b uses multiple edge detectors to convert the time between the path delay and the clock period to digital codes [16]. The digital codes are then sent to the system controller where a look-up table (LUT) based adaptation can be implemented. Figure 16c shows a measurement method that uses frequency counting [33]. This measurement method is particularly useful for monitoring device parameters of V_{th}, temperature, and so on. Using the system clock for the conversion will require calibration of the monitoring circuit for every supply voltage which will increase the test cost. Instead, we can utilize a locally generated clock using a ring oscillator. The output in this case is the ratio of the monitor frequency and the reference frequency. The measured values of the frequency ratio are then compared with predefined values to monitor how much the monitoring parameter varied from the targeted values. For applications where the clock frequency is fixed, process and temperature sensitive monitors can also implemented with edge detection mechanisms. An up/down counter based detection circuit to detect the V_{th} deviation from predefined values has been employed for dynamic adaptation of V_{th} values [34].

4 Parameter Extraction for Model-Hardware Correlation

The circuit techniques described in Sect. 3 realize delay characteristics that are sensitive to particular parameter variations. However, they do not give us the value of the parameter variation itself. In this section, we describe a parameter extraction technique that takes the delay values of multiple delay paths and then estimates the variations in each of the parameters. The parameters can be transistor threshold voltage, temperature, gate-length or any device related parameter. We can then utilize the extracted parameters for test strategies and process optimization.

4.1 Parameter Extraction Methodology

In the case of an inverter gate, the gate–source voltage of each transistor goes through different values during a “High” to “Low” and a “Low” to “High”switching events. Thus, it is not straight forward to relate physical device parameters to the delay information. Parameter estimation gets harder when the supply voltage is lowered, such that the delay becomes non-linear to the parameter changes. So, the question is how to correlate the model to each chip to get a good accuracy.

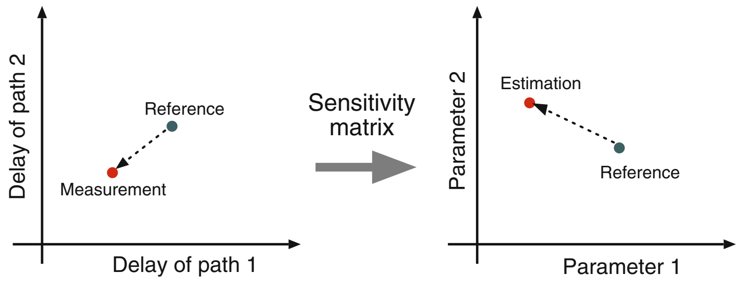


Fig. 17 Estimation of physical parameters from multiple delay paths. Sensitivity coefficient links the physical parameters to delay values

Section 3 demonstrates different types of delay paths to monitor various physical and environmental parameters. The designs are carefully performed to make the delay particularly sensitive to the parameter of interest. The techniques allow us to comparatively track the change of the parameters in the run-time. However, because of the mismatch in the model and hardware, the absolute parameter monitoring contains errors. Calibrations need to be performed to reduce the errors to acceptable ranges. For debugging purposes, we may want to correlate our transistor models to actual transistor characteristics in the chip. To perform model-hardware correlation, key model parameters such as the V_{th} and β may suffice as they are the dominant sources of fluctuations, although other parameters may also be used.

Figure 17 illustrates the concept of parameter extraction from multiple delay values. The left side of the figure plots the delay of a path against the delay of a different path. The round point shows a point which is obtained by circuit simulation. The cross point emulates a measured value from a chip. The difference between the two points here contains process information. Using sensitivity coefficients, we can estimate the amount of deviation in the process parameters and transform the delay space to process space which is shown in the right side of the figure. The key point here is not to use transistor I–V characteristics, rather use the delay characteristics to extract these parameters. For robust extraction of the parameters, we need to design the delay paths, such that the sensitivity matrix has a low condition number [22]. We can then build a system of linear equations using the sensitivity coefficients.

4.2 Measurement Results

To demonstrate the monitoring capability of the V_{thp}-sensitive and V_{thn}-sensitive delay cells of Figs. 8 and 9, measurements of ring oscillators are performed for a 65 nm bulk process. Figure 18 plots the values of V_{thp} and V_{thn} estimated under different body bias conditions for a particular chip. In the figure, the x-axis refers to V_{thp} estimations and the y-axis refers to V_{thn} estimations. Rectangular points are

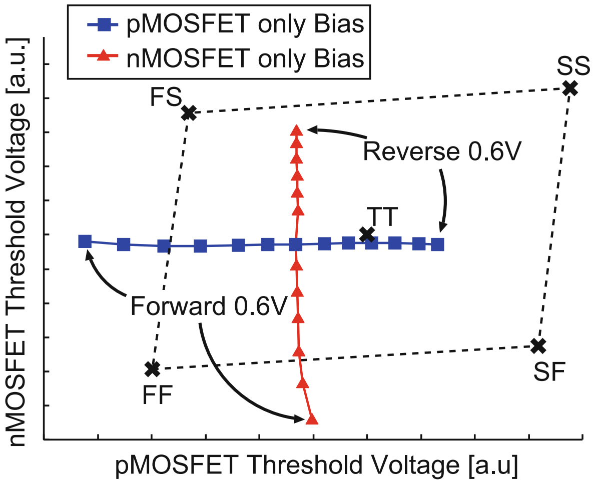


Fig. 18 Estimation results of V_{thn} and V_{thp} of a chip for different body bias values. Either the pMOSFETs or the nMOSFETs are biased simultaneously

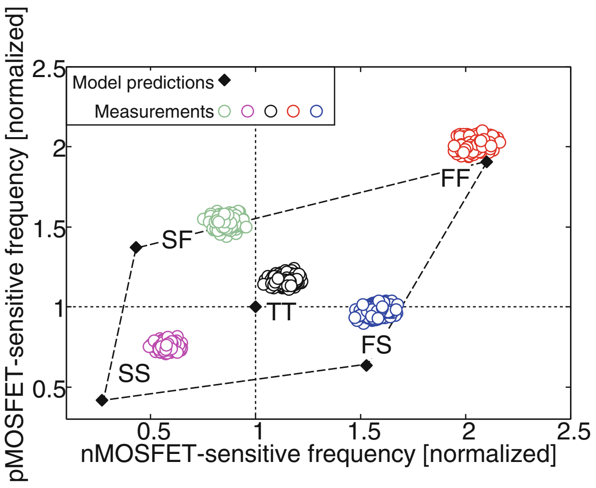


Fig. 19 V_{thp}-sensitive RO (ring oscillator) frequencies against V_{thn}-sensitive RO frequencies

estimated values of V_{thp} and V_{thn} when only pMOSFET is biased. Triangular points refer to estimated values of V_{thp} and V_{thn} when only nMOSFET is biased. When only pMOSFET is biased, the estimated point moves in the horizontal direction referring that only V_{thp} is being changed in the estimation. When only nMOSFET is biased, the estimated point moves in the vertical direction referring that only V_{thn} is being changed in the estimation. Thus, it is demonstrated that any change in the threshold voltage can be detected correctly by the proposed monitor circuits.

Figure 19 shows the measured frequencies of V_{thp}-sensitive and V_{thn}-sensitive ring oscillators from several chips (open circles). The chips have been fabricated targeting either of the five process corners of "TT," "SS," "FF," "FS," and "SF." The values are normalized by the values simulated with the transistor models targeted for the "TT" process corner. Frequency values simulated using the other corner models

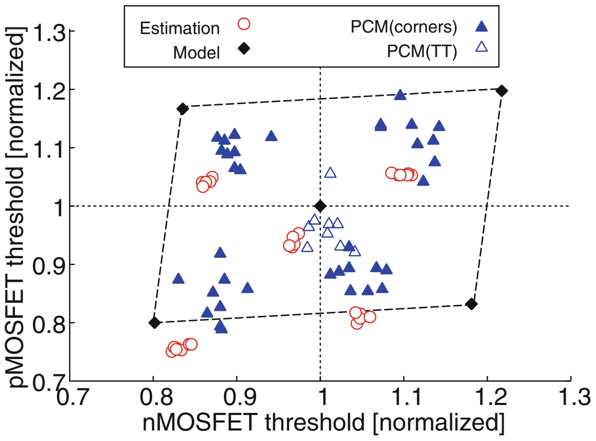


Fig. 20 V_{th} estimation results for different corner chips

are also plotted in the figure (closed squares). In the figure, process shifts from the "TT" model prediction are observed. Clear deviations are observed for "TT," "SS," "SF," and "FS" corners. The silicon values are higher than the model predictions. With comparison with the models, we can have quick understanding of process shift for each chip. This information allow us to take decisions for silicon debug and test pattern generation. We can now extract the device parameters of V_{thp}, V_{thn}, and β using sensitivity analysis, model-hardware correlation can be obtained that allows us accurately predict the delay performance. Figure 20 plots the estimated V_{thp} and V_{thn} values. V_{th} values provided in the corner models are also plotted in the figure. Furthermore, V_{th} values provided by the Process Control Modules (PCM) that are generally placed in the scribe-lines are also plotted. The estimated values correlate with the PCM data and also show die-to-die variations.

5 Conclusion

In this chapter, we have shown the importance of cross-layer resiliency for energy-efficient and robust operation of circuits. Cross-layer resiliency is performed by tuning the threshold voltage and supply voltage in run-time based on information of process, leakage current, circuit activity, and temperature. Run-time monitoring of these parameters are essential in achieving cross-layer resiliency. To incorporate the monitor circuits into a cell-based design flow, we have discussed delay-based monitoring techniques. Cell-based design of monitor circuits enables to place the monitors inside the circuit. Placing the monitors inside the target circuit realizes better correlations between the monitor behavior and the actual circuit behavior.

We have discussed a general design methodology to synthesize a critical path monitor. There are several methods to monitoring the critical delay having a trade-off relationship between accuracy and implementation cost. Implementation cost here can be area overhead, test cost, and/or both. The selection of a suitable critical path monitor thus has to be made based on the critical nature of the application.

Besides the critical path monitoring, run-time monitoring of physical parameters of threshold voltage, temperature, and leakage current are essential for energy-efficient operation under parameter fluctuation and aging. Utilizing the relationship between the delay of a logic gate and the physical parameters, several circuit topologies are discussed that amplify the effect a certain parameter. Threshold-dominant inverter topologies and leakage current driven inverters are suitable for temperature, leakage current, and threshold voltage monitoring.

References

1. ITRS, International Technology Roadmap for Semiconductors. Technical Report [Online]. Available: http://www.itrs.net
2. Borkar, S., Karnik, T., Narendra, S., Tschanz, J., Keshavarzi, A., De, V.: Parameter variations and impact on circuits and microarchitecture. In: Design Automation Conference, pp. 338–342 (2003)
3. Paul, B.C., Kang, K., Kufluoglu, H., Alam, M.A., Roy, K.: Impact of NBTI on the temporal performance degradation of digital circuits. IEEE Electron Device Lett. **26**(8), 560–562 (2005)
4. Horowitz, M.: 1.1 Computing's energy problem (and what we can do about it). In: IEEE International Solid-State Circuits Conference, pp. 10–14 (2014)
5. Bohr, M.: The new era of scaling in an SoC world. In: IEEE International Solid State Circuits Conference, pp. 23–28 (2009)
6. Enz, C.C., Krummenacher, F., Vittoz, E.A.: An analytical MOS transistor model valid in all regions of operation and dedicated to low-voltage and low-current applications. Analog Integr. Circ. Sig. Process **8**(1), 83–114 (1995)
7. Marr, B., Degnan, B., Hasler, P., Anderson, D.: Scaling energy per operation via an asynchronous pipeline. IEEE Trans. Very Large Scale Integr. VLSI Syst. **21**(1), 147–151 (2013)
8. Ernst, D., Kim, N.S., Das, S., Pant, S., Rao, R., Pham, T., Ziesler, C., Blaauw, D., Austin, T., Flautner, K., Mudge, T., Ave, B., Arbor, A.: Razor: a low-power pipeline based on circuit-level timing speculation. In: IEEE/ACM International Symposium on Microarchitecture, pp. 7–18 (2003)
9. Das, S., Tokunaga, C., Pant, S., Ma, W.-H., Kalaiselvan, S., Lai, K., Bull, D.M., Blaauw, D.T.: RazorII: In situ error detection and correction for PVT and SER tolerance. IEEE J. Solid State Circuits **44**(1), 32–48 (2009)
10. T. Nakura, K. Nose, M. Mizuno, Fine-grain redundant logic using defect-prediction flip-flops. In: IEEE International Solid-State Circuits Conference, pp. 21–23 (2007)
11. Bowman, K.A., Tschanz, J.W., Kim, N.S., Lee, J.C., Wilkerson, C.B., Lu, S.L.L., Karnik, T., De, V.K.: Energy-efficient and metastability-immune resilient circuits for dynamic variation tolerance. IEEE J. Solid State Circuits **44**(1), 49–63 (2009)
12. Das, B.P., Onodera, H.: Frequency-independent warning detection sequential for dynamic voltage and frequency scaling in ASICs. IEEE Trans. Very Large Scale Integr. VLSI Syst. **22**(12), 2535–2548 (2014)
13. Firouzi, F., Ye, F., Chakrabarty, K., Tahoori, M.B.: Aging-and variation-aware delay monitoring using representative critical path selection. ACM Trans. Des. Autom. Electron. Syst. **20**(3), 39:1–39:23 (2015)
14. Chan, T.-B., Gupta, P., Kahng, A. B., Lai, L.: Synthesis and analysis of design-dependent ring oscillator (DDRO) performance monitors. IEEE Trans. Very Large Scale Integr. VLSI Syst. **22**(10), 2117–2130 (2014)

15. Park, J., Abraham, J.A.: A fast, accurate and simple critical path monitor for improving Energy-Delay product in DVS systems. In: IEEE/ACM International Symposium on Low Power Electronics and Design, pp. 391–396 (2011)
16. Drake, A., Senger, R., Deogun, H., Carpenter, G., Ghiasi, S., Nguyen, T., James, N., Floyd, M., Pokala, V.: A distributed critical-path timing monitor for a 65nm high-performance microprocessor. In: IEEE International Solid-State Circuits Conferencepp. 398–399 (2007)
17. Tschanz, J., Bowman, K., Walstra, S., Agostinelli, M., Karnik, T., De, V.: Tunable replica circuits and adaptive voltage-frequency techniques for dynamic voltage, temperature, and aging variation tolerance. In: IEEE Symposium on VLSI Circuits, pp. 112–113 (2009)
18. Drake, A.J., Floyd, M.S., Willaman, R.L., Hathaway, D.J., Hernandez, J., Soja, C., Tiner, M.D., Carpenter, G.D., Senger, R.M.: Single-cycle, pulse-shaped critical path monitor in the POWER7+ microprocessor. In: Proceedings of the International Symposium on Low Power Electronics and Designpp. 193–198 (2013)
19. Tokunaga, C., Ryan, J.F., Karnik, T., Tschanz, J.W.: Resilient and adaptive circuits for voltage, temperature, and reliability guardband reduction. In: IEEE International Reliability Physics Symposium, pp. 1–5 (2014)
20. Drake, A.J., Senger, R.M., Singh, H., Carpenter, G.D., James, N.K.: Dynamic measurement of critical-path timing. In: IEEE International Conference on Integrated Circuit Design and Technology, pp. 249–252 (2008)
21. Ikenaga, Y., Nomura, M., Suenaga, S., Sonohara, H., Horikoshi, Y., Saito, T., Ohdaira, Y., Nishio, Y., Iwashita, T., Satou, M., Nishida, K., Nose, K., Noguchi, K., Hayashi, Y., Mizuno, M.: A 27% Active-Power-Reduced 40-nm CMOS multimedia SoC with adaptive voltage scaling using distributed universal delay lines. IEEE J. Solid State Circuits **47**(4), 832–840 (2012)
22. Islam, A.K.M.M., Tsuchiya, A., Kobayashi, K., Onodera, H.: Variation-sensitive monitor circuits for estimation of global process parameter variation. IEEE Trans. Semicond. Manuf. **25**(4), 571–580 (2012)
23. Fujimoto, S., Mahfuzul Islam, A.K., Matsumoto, T., Onodera, H.: Inhomogeneous ring oscillator for within-die variability and RTN characterization. IEEE Trans. Semicond. Manuf. **26**(3), 296–305 (2013)
24. Iizuka, T., Asada, K.: All-digital PMOS and NMOS process variability monitor utilizing shared buffer ring and ring oscillator. IEICE Trans. Electron. **E95-C**(4), 627–634 (2012)
25. Kim, T.T.H., Lu, P.F., Jenkins, K.A., Kim, C.H.: A ring-oscillator-based reliability monitor for isolated measurement of NBTI and PBTI in high-k/metal gate technology. IEEE Trans. Very Large Scale Integr. VLSI Syst. **23**(7), 1360–1364 (2015)
26. Nishizawa, S., Onodera, H.: A ring oscillator with calibration circuit for on-chip measurement of static IR-drop. IEEE Trans. Semicond. Manuf. **26**(3), 306–313 (2013)
27. Krishnaswamy, V., Brooks, J., Konstadinidis, G., Curtis, M., Pham, H., Turullols, S., Shin, J., Yifan, Y., Zhang, H.: Fine-grained adaptive power management of the SPARC m7 processor. In: IEEE International Solid-State Circuits Conference, pp. 1–3 (2015)
28. Mair, H.T., Gammie, G., Wang, A., Lagerquist, R., Chung, C., Gururajarao, S., Kao, P., Rajagopalan, A., Saha, A., Jain, A., et al.: A 20nm 2.5 GHz ultra-low-power tri-cluster CPU subsystem with adaptive power allocation for optimal mobile SoC performance. In: IEEE International Solid-State Circuits Conference, pp. 76–77 (2016)
29. Hokimoto, S., Ishihara, T., Onodera, H.: Minimum energy point tracking using combined dynamic voltage scaling and adaptive body biasing. In: IEEE International System-on-Chip Conference, pp. 1–6 (2016)
30. Islam, A.K.M.M., Onodera, H.: Area-efficient reconfigurable ring oscillator for device and circuit level characterization of static and dynamic variations. Jpn. J. Appl. Phys. **53**(4S), 04EE08 (2014)
31. Islam, A.K.M.M., Shiomi, J., Ishihara, T., Onodera, H.: Wide-supply-range all-digital leakage variation sensor for on-chip process and temperature monitoring. IEEE J. Solid State Circuits **50**(11), 2475–2490 (2015)

32. Bowman, K., Tschanz, J., Lu, S., Aseron, P., Khellah, M., Raychowdhury, A., Geuskens, B., Tokunaga, C., Wilkerson, C., Karnik, T., et al.: A 45 nm resilient microprocessor core for dynamic variation tolerance. IEEE J. Solid State Circuits **46**(1), 194–208 (2011)
33. Kim, Y., Shin, D., Lee, J., Lee, Y., Yoo, H.J.: A 0.55 V 1.1 mW artificial intelligence processor with on-chip PVT compensation for autonomous mobile robots. IEEE Trans. Circuits Syst. Regul. Pap. **65**(2), 567–580 (2018)
34. Mahfuzul, I., Kamae, N., Ishihara, T., Onodera, H.: A built-in self-adjustment scheme with adaptive body bias using P/N-sensitive digital monitor circuits. In: IEEE Asian Solid State Circuits Conferencepp. 101–104 (2012)

Dealing with Aging and Yield in Scaled Technologies

Wei Ye, Mohamed Baker Alawieh, Che-Lun Hsu, Yibo Lin, and David Z. Pan

1 Introduction

The aging and yield issues arise with aggressive scaling of technologies and increasing design complexity [51, 53]. These issues impact the circuit performance and functionality throughout the product life cycles. The sources of aging and yield concerns lie in different aspects, getting more severe with technology scaling.

Modern VLSI designs have to cope with unreliable components and processes. Device aging and interconnect electromigration effects are likely to cause unexpected performance degradation and even malfunctions at the end of circuit life cycles. Meanwhile, process variations may lead to manufacturing defects and inconsistent device characterization, causing yield issues. Ignoring these effects leads short lifetime of designs and low yield, eventually increases the costs in volume production and maintenance.

Thus, for the robustness of VLSI design methodology and cycles, reliability and yield need to be accurately modeled, systematically optimized, and seamlessly integrated into the existing design flow. This chapter will survey critical aging and yield issues, and then review the state-of-the-art techniques to tackle them, including both modeling and optimization strategies which reside across the Physics and Circuit/Gate layers as part of the overall dependability scheme shown in Fig. 1. The strategies often involve synergistic cross-layer optimization due to the complicated VLSI design procedures nowadays. Novel modeling techniques leveraging machine learning are analyzed along with analytical optimization approaches.

W. Ye · M. B. Alawieh · C.-L. Hsu · D. Z. Pan (✉)
University of Texas at Austin, Austin, TX, US
e-mail: weiye@utexas.edu; mohdbaker@utexas.edu; chsu1@utexas.edu; dpan@ece.utexas.edu

Y. Lin
Peking University, Beijing, China
e-mail: yibolin@pku.edu.cn

J. Henkel, N. Dutt (eds.), *Dependable Embedded Systems*, Embedded Systems,
https://doi.org/10.1007/978-3-030-52017-5_17

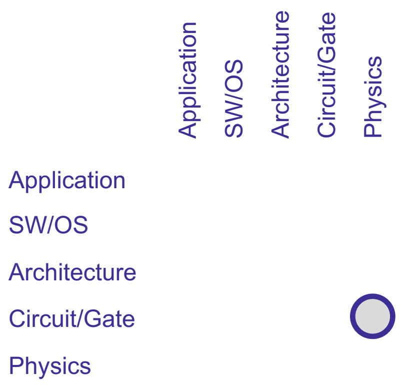


Fig. 1 This chapter covers reliability and yield issues across the borders between Physics and Circuit/Gate levels

The chapter starts by investigating the device and interconnect reliability issues for modern VLSI designs in Sect. 2. It covers different device aging effects, such as bias temperature instability and hot carrier injection, as well as electromigration effects on power/ground and signal interconnections. The section introduces the modeling techniques along with optimization strategies to increase the design robustness under these effects. Section 3 dives into the state-of-the-art practices in yield issues for both analog and digital circuits. This section examines the impacts of process variations on circuit performance and manufacturing defects followed by effective modeling techniques to capture these issues early in the design flow. In the end, the chapter is concluded with Sect. 4.

2 Reliability Modeling and Optimization

With the continued feature size shrinking, reliability issue becomes increasingly severe. This section covers recent researches on aging modeling and analysis and divide the aging concerns into two sub-categories: aging at the device level and aging at the interconnect level.

2.1 Device Aging

As CMOS technologies continue to shrink, device reliability becomes a major challenge for high performance computing (HPC) and automotive applications which require robust circuit design. This section presents the device reliability modeling and optimization techniques along with mitigation strategies in advanced CMOS technologies.

Device reliability can be divided into time-independent and time-dependent categories. Time-independent reliability issues are caused by manufacturing variations

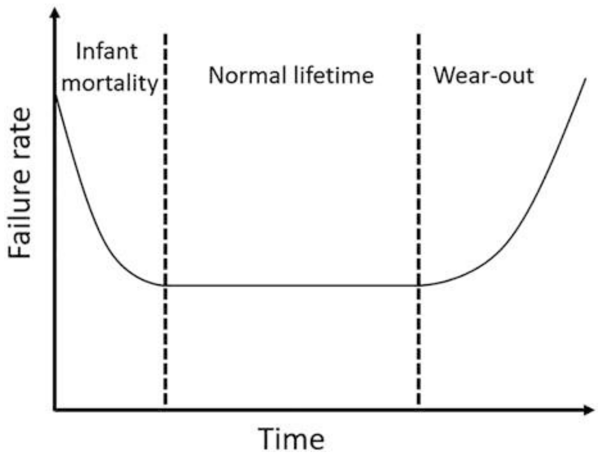


Fig. 2 Bathtub curve that illustrates the device life cycle

or noise such as random telegraph noise (RTN) or soft errors. Time-dependent reliability issues, also known as aging effects, can be illustrated using the bathtub curve in Fig. 2 which has high but decreasing failure rate in early life, low and constant failure rate in normal operation, and increasing high failure rate at the end of life wear-out period. This section focuses on modeling time-dependent reliability issues including *bias temperature instability* (BTI) and *hot carrier injection* (HCI).

BTI is an aging mechanism characterized by an increase in the device threshold voltage and a decrease in its mobility which eventually lead to an increase in the gate delay, and thus performance degradation [9, 58]. The two major factors contributing to the BTI phenomenon are the voltage bias and temperature. The term bias refers to the gate-to-source voltage bias applied to the transistor gate which is mostly a negative bias for PMOS, and a positive bias for NMOS. The theory behind BTI can be jointly explained by the reaction–diffusion (R-D) model and the charge trapping (CT) model [58]. The R-D model describes the degradation process when hole accumulation dissolves Si-H bond (reaction) and hydrogen diffuses away (diffusion), whereas the recovery stage takes place when voltage bias and duty factor stress is not present [25, 26]. The CT model explains the threshold voltage degradation by the trapped charge in the defected gate dielectrics. Early studies focused on BTI mitigation partially due to the fact that BTI dominates aging in early stages; however, HCI is more important at later stages where HCI contributes 40%–80% of device aging after 10 years of deployment [21, 47].

HCI is an aging phenomenon that degrades device drain current and is caused by the accumulation of carriers (electrons or holes) under the lateral electric fields, which can gain enough energy to damage and degrade the device mobility [16]. The traditional theory behind HCI was called lucky electron model, which is a field-based model [12, 57]. However, with the scaling of the supply voltage, the reduced electric field made HCI prediction based on field-based models a challenging task.

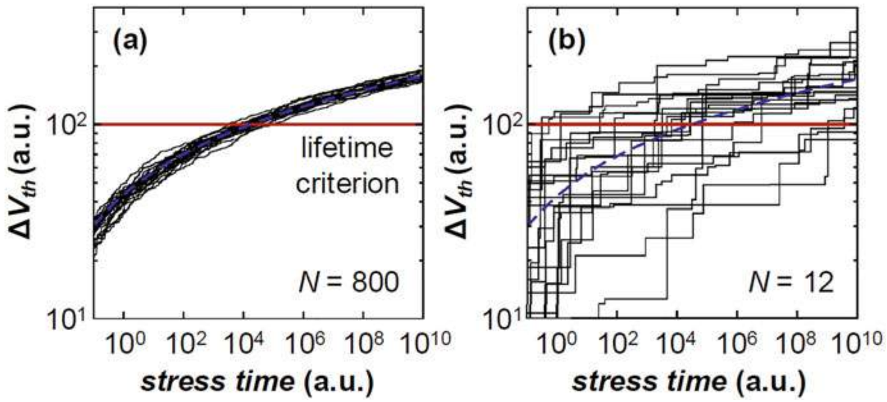


Fig. 3 BTI threshold voltage shift vs time for sub-45nm CMOS. BTI effect has stochastic nature in deep-sub-micro devices. Averaging each sample across large sample size of 800 (**a**) can achieve a well-defined voltage vs stress time curve while the voltage vs stress time trend in a much small sample size (**b**) contains larger variation [34]

Recent researches have proposed energy-driven theories to generalize HCI effects when devices are in low supply voltage [27, 52].

Characterizing aging degradation on circuit performance using aging model is a crucial step prior to optimization. Researchers can build deterministic models for BTI and HCI-related aging in old technologies such as 180 nm node. However, Kaczer et al. studied the threshold voltage shift vs time under the BTI effect and found its stochastic nature in deep-sub-micron nodes as shown in Fig. 3 [34]. Lorenz et al. proposed the first gate-level timing analysis considering NBTI and HCI [43]. Huard et al. [32] characterized a digital library gates under NBTI and HCI aging effects. Ren et al. discovered that BTI and HCI-related aging effects have layout dependencies [54]. In [21, 22], Fang et al. proposed frameworks to analyze BTI and HCI impacts on large digital circuits and [59] used ring oscillator-based sensors to estimate HCI/BTI induced circuit aging. Moreover, flip-flop based sensor was introduced in [2] to predict BTI aging circuit failure.

Recent researches not only model the aforementioned aging issues, but also propose design methods and optimizations for more reliable designs. Reliability optimization can be done at architecture level, logic synthesis level, and physical design level. At the architecture level, [48] demonstrated an aging analysis framework that examines NBTI and HCI to predict performance, power, and aging in the early design phase. Firouzi et al. [23] alleviated NBTI effects by using NOP (No operation) assignment and insertion in the MIPS processor. At synthesis level, Kumar et al. introduced standard cell mapping that considers signal probabilities to reduce BTI stress [35]. In [20], both HCI and BTI were considered during logic synthesis stage and put tighter timing constraint on paths with higher aging rate. Chakraborty et al. [13] optimized NBTI-induced clock skew in gated clock tree. Gate sizing [55, 64] and pin-reordering/logic restructuring [68] are also implemented to minimize BTI effects. At the physical design level, Hsu et al. [29] proposed a layout-dependent aging mitigation framework for critical path timing during standard cell placement stage and [81] introduced aging-aware FPGA placement. Gate replacement techniques were used in [65] to co-optimize circuit aging and leakage.

2.2 *Interconnect Electromigration*

As IC technologies continue to scale, complex chip functionalities have been made possible by virtue of increasing transistor densities and aggressive scaling of interconnects. Besides, interconnects are getting thinner and running longer. These factors bring along higher current densities in metal wires, a phenomenon that further exacerbates *electromigration* (EM). The failure time from EM is worsened even further by the local temperature increase caused by self-heating of underlying FinFETs.

EM is the gradual displacement of atoms in metal under the influence of an applied electric field and is considered the primary failure mechanism for metal interconnects. After the migration of atoms with electrons in a metal line for a certain period, a void grows on one side, which increases the resistance of the metal line and may eventually lead to open circuits. Hillock is formed on the other side and may cause short circuits. Figure 4 shows the scanning electron microscopy (SEM) images of void and hillock.

2.2.1 Power EM Modeling

An empirical model for the mean time to failure (MTTF) of a metal line subjected to EM is given by Black's equation [11]

$$\mathrm{MTTF} = \frac{A}{J^n} \exp\left(\frac{E_a}{kT}\right), \tag{1}$$

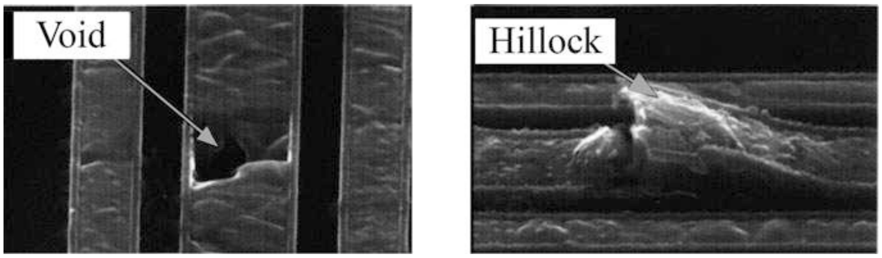


Fig. 4 A void and a hillock generated by electromigration [10]

where A is a constant which comprises the material properties and the geometry of the interconnect, J is the current density, E_a is the activation energy, k is the Boltzmann constant, and T is the temperature. n is the constant exponent of the current density and is usually set to 2. With Black's equation in Eq. (1), the relation between interconnect lifetime and both current and temperature can be readily estimated.

Power grid is one of the interconnect structures most vulnerable to EM due to its high unidirectional currents. Lower-level metal layers of power grids are more susceptible to EM failures due to smaller wire width. Besides, EM violations are most likely to occur around weak power grid connections, which deliver current to high power-consuming regions.

Hsu et al. [30] proposed an average power-based model to evaluate power grid static EM at placement stage. Ye et al. [78] further modified the model by considering the sum of the dynamic and leakage currents for a standard cell at this stage, which is given by:

$$I = \alpha \cdot C \cdot V_{\text{DD}} \cdot f + I_{\text{leak}},$$

where α is the cell activity factor, V_{DD} is the supply voltage, and f is the system clock frequency. C is the sum of the load capacitance and the output pin capacitance. Load capacitance further includes downstream gate capacitance and interconnect capacitance. Since nets have not been routed at this stage, half-perimeter wirelength (HPWL) [8] is widely adopted to estimate interconnect capacitance in placement. Power tile is defined as the region between two adjacent VDD (or VSS) power stripes and the adjacent power rails. Figure 5 demonstrates how to calculate the maximum current in the local power rails within a power tile. P_l and P_r are the left and right endpoints of the VDD power rail. d_i^l and d_i^r are the distances from the midpoint of the i-th cell to P_l and P_r, respectively. R_i^l and R_i^r are the wire resistances of the corresponding metal segments, which are proportional to d_i^l and d_i^r. The following equations hold

$$I_i^l = d_i^r / \left(d_i^l + d_i^r\right) I_i, \quad I_i^r = d_i^l / \left(d_i^l + d_i^r\right) I_i.$$

The currents drawn by all the cells in the power tile from P_l and P_r are computed as:

$$I^l = \sum_i I_i^l, \quad I^r = \sum_i I_i^r.$$

Therefore, there exists an EM violation in a particular power tile if $\max\{I^l, I^r\} > I_{\text{limit}}$. In this way, the EM failures in the local power rails can be estimated at the placement stage; thus, enabling an EM-aware placement that can effectively reduce the EM violations.

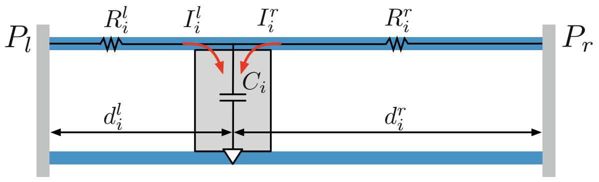


Fig. 5 The power grid model for current calculation in a power tile [78]

2.2.2 Signal EM Modeling

Previously, electromigration on signal interconnects does not draw great attention. Alternating current (AC) flows inside signal interconnects; when the direction of the current in an interconnect is reversed, the direction of EM diffusion is also reversed. The damage caused by EM can be partially cleared due to this compensation by material backflow. This effect is known as a self-healing, which can significantly extend the lifetime of a wire. Black's equation for AC is given by [40, 63]:

$$\mathrm{MTTF} = \frac{A}{(J_+ - \gamma J_-)^n} \exp\left(\frac{E_a}{kT}\right), \tag{2}$$

where J_+ and J_- are the current densities during positive and negative pulses. γ is the self-healing coefficient which is determined by the duty factor of the current and other factors influencing the scale of self-healing, such as the frequency [39]. Previously, signal electromigration has attracted little attention due to the benefits of healing effect. However, EM failures in signal interconnects are no longer negligible due to higher clock frequencies, large transistor density, and the negative impact of FinFET self-heating at advanced nodes.

In [76], a set of features from the placement is extracted to train a machine learning model for EM detection before routing. Despite the fact that the current profile for the design is not available at the placement stage, multiple features that are highly correlated with the current can be crafted. These features can be divided into net-specific features and neighborhood related features. The net-specific features—including HPWL, the number of net pins, etc.—capture the net attributes. On the other hand, neighborhood related features are used to capture information about possible congestion around net pins.

The pre-routing signal EM hotspot prediction can be reduced to a classification problem [76]. A two-stage detection approach based on logistic regression shown in Fig. 6 is introduced to reduce the number of false alarms. In the first stage, a classification model M1 is trained to predict EM hotspots using all the nets in the training dataset. After the first stage, all nets with NH (Non-hotspot) prediction will be labeled as NH without further processing. For nets labeled H (Hotspot) by M1, a new model, M2, is trained to prune out false alarms. With an accurate classification model to detect signal EM hotspots based on the information available

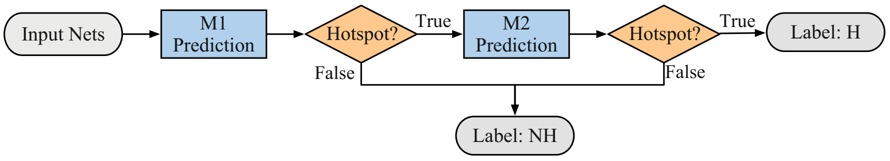


Fig. 6 The flow of the two-stage signal EM hotspot detection approach [76]

at the placement stage, early stage EM handling is enabled, which reduces iterative EM fixing cost.

2.2.3 EM Optimization Flow

Through the preceding EM modeling in Eqs. (1) and (2), EM failures can be detected after the physical design stage, and then be fixed through layout modification. Xie et al. [69] proposed control logics to balance currents in both directions of power rails to mitigate the EM effects. Lienig [38] suggested the exploitation of several EM inhibiting measures, such as bamboo structure, short-length, and reservoir effects. Other studies [14, 33] considered global routing for EM optimization. In [49] de Paris et al. adopted a design strategy using non-default routing (NDR) rules to re-route the wire segments of EM-unsafe signal nets that present high current densities.

Conventionally, EM checking is invoked after the routing stage [36]. Current densities in metal wires are computed and compared with foundry-specified limits to detect EM failures. Next, the failures are fixed with engineering change order (ECO) efforts. EM checking leverages post-routing information to detect violations, which consequently limits the efficiency of addressing techniques. In the routing phase, the locations of standard cells and the corresponding current distribution are already fixed and the traditional fixing approaches such as wire widening and cell resizing are not effective enough to handle the ever-growing number of EM violations [1]. It is of vital importance to incorporate EM detection and fixing techniques into earlier stages of physical design (PD).

Two clear benefits are associated with such early stage EM handling. First, the number of EM violations can be decreased further by using various techniques at different design stages. Second, introducing early stage mitigation techniques can help reduce the resulting overhead when compared to post-routing fixing techniques. Thus, moving the EM detection and resolving steps to earlier stages of the physical design can help in reducing runtime or the number of iterations needed for design closure. In [78], a series of detailed placement techniques was proposed to mitigate power grid EM. Ye et al. [76] proposed a multistage EM mitigation approach at placement and routing phases to address the problematic nets detected by the classification model.

3 Yield Modeling and Optimization

3.1 Performance Modeling

With technologies descending deep into the sub-micron spectrum, process variation manifests itself among the most prominent factors limiting the product yield of analog and mixed-signal (AMS) circuits. Thus, it is indispensable to consider this variation in the design flow of modern ICs [42]. Conventionally, performance modeling has been adopted to capture this variability through analytical models that can be used in various applications such as yield estimation and design optimization [4].

Given a set of samples, the performance model coefficients are conventionally obtained through least-squares regression (LSR). However, LSR can build accurate models only when the number of samples is much greater than the number of unknown coefficients. Thus, given the high dimensionality of the performance models in complex AMS circuit designs, the simulation cost for building accurate models can be exorbitant. Hence, most recent performance modeling techniques incorporate additional information about the model to reduce the number of simulations needed [3, 5, 7].

3.1.1 Sparse Modeling

Although the number of basis functions representing the process variability is large, a few of these basis functions are required to accurately model a specific performance of interest (PoI). Hence, the vector of coefficients contains a small number of non-zero values corresponding to important basis functions [37]. This information can be incorporated in the modeling by constraining the number of non-zero coefficients in the final model.

While constraining the number of non-zero coefficients accurately reflects the sparse regression concept, the optimization problem is NP-hard. Besides heuristic approaches that select important basis functions in a greedy manner, Bayesian approaches have been widely applied to address this challenge [37]. In practice, a shrinking prior on the model coefficients is used to push their values close to zero. Examples of this include applying a Gaussian or Laplacian prior which results in Ridge and Lasso regression formulations, respectively. This allows incorporating sparse prior knowledge; however, such approaches do not perform explicit variable selection and they penalize high coefficients values by pushing all coefficients close to zero instead of selectively setting unimportant ones to zero.

On the other hand, a Bayesian spike and slab feature selection technique can be employed to efficiently build accurate performance models [7]. Spike and slab models explicitly partition variables into important and non-important, and then solve for the values of the important variables independently of the feature selection mechanism. A hierarchical Bayesian framework is utilized to determine both the importance and value of the coefficients simultaneously. At its highest level, the hierarchy dictates that a particular coefficient is sampled from one of the two zero-mean prior Gaussian distributions: a low variance distribution centered around zero, referred to as the **spike**, and a large variance distribution, referred to as the **slab**.

This mixture of priors approach has demonstrated superior results compared to traditional sparse modeling schemes while also providing a feature selection framework that can easily select important features in the model [7].

3.1.2 Semi-Supervised Modeling

Traditionally, performance modeling has been approached from a purely supervised perspective. In other words, performance models were built by using labeled samples obtained through expensive simulations. However, as the complexity of designs increased, obtaining enough samples to build accurate models has become exorbitant. Recently, a new direction, derived from semi-supervised learning, has been explored to take advantage of unlabeled data to further improve the accuracy of performance modeling for AMS designs [3, 5].

In practice, the hierarchical structure of many AMS circuits can be leveraged to incorporate unlabeled data via Bayesian co-learning [5]. In particular, such an approach is composed of three major components. First, the entire circuit of interest is partitioned into multiple blocks based on the netlist hierarchy. Second, circuit-level performance models are built to map the block-level performance metrics to the PoI at the circuit level. Such a mapping is often low-dimensional; thus it can be accurately approximated by using a small number of simulation samples. Third, by combining the aforementioned low-dimensional models and an unlabeled data set, a complex, high-dimensional performance model for the PoI can be built based on semi-supervised learning.

To implement this modeling technique, a Bayesian inference is formulated to integrate the aforementioned three components, along with the prior knowledge on model coefficients, in a unified framework. Experimental results shown in [5] demonstrate that the proposed semi-supervised leaning approach can achieve up to $3.6\times$ speedup when compared to sparse regression-based approaches.

While many AMS circuits exhibit a hierarchical structure, this feature is not always present. Hence, a more general semi-supervised framework which makes no assumption about the AMS circuit structure is desirable [3]. This can be achieved by incorporating a co-learning technique that leverages multiple views of the process variability to efficiently build a performance model. The first is the device level variations such as ΔV_{TH} or Δw_{eff}, while the second view is the underlying set of independent random variables, referred to as process variables. Traditionally, performance modeling targets expressing the PoI as an analytical function of process variables; however, capitalizing on information provided by the device level variability as an alternative view can help efficiently build the performance model for the PoI [3].

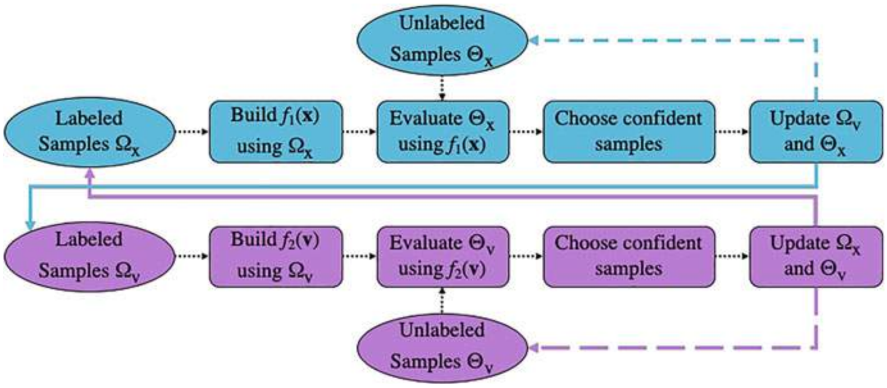

Unlabeled Samples Θ_x
Labeled Samples Ω_x
Build $f_1(\mathbf{x})$ using Ω_x
Evaluate Θ_x using $f_1(\mathbf{x})$
Choose confident samples
Update Ω_v and Θ_x
Labeled Samples Ω_v
Build $f_2(\mathbf{v})$ using Ω_v
Evaluate Θ_v using $f_2(\mathbf{v})$
Choose confident samples
Update Ω_x and Θ_v
Unlabeled Samples Θ_v

Fig. 7 An iteration of the semi-supervised co-learning modeling framework is illustrated [3]

As shown in Fig. 7, the key idea is to use a small number of labeled samples to build an initial model for each of the views of the data (**x** and **v**), then attempt to iteratively bootstrap from the initial models using unlabeled data. In other words, initial models can be used to give pseudo-labels for unlabeled data, then the most confident predictions from a particular model are used as pseudo-samples for the other model. In each iteration step, *highly confident* pseudo-samples are fused with the small number of available labeled samples to build a new model. Experimental results demonstrated up to 30% speedup compared to sparse regression-based approaches [3].

3.1.3 Performance Optimization

Besides capturing the major sources of variability in AMS designs, one of the main applications of performance modeling is yield estimation and optimization. In practice, performance optimization can make use of trained models towards optimizing the performance of the design. This is established by first capturing correlations between the performance variability and the device sizes or reconfiguration knobs, then adjusting these parameters to improve the parametric yield [4, 6].

Moreover, with the increase in AMS circuits complexity, increasing nonlinearity stands out as major factor limiting the capabilities of performance modeling and optimization. Hence, performance optimization techniques relying on non-parametric surrogate models and Bayesian optimization frameworks have been recently proposed [31, 83]. These surrogate models are typically Gaussian Processes, and Bayesian optimization is used to find optimal values given a black-box function.

Bayesian Optimization is a sequential sampling based optimization technique for optimizing block-box objective functions. At each step, a set of optimal sampling locations are selected based on a chosen acquisition function. Then, queries of the objective function to be optimized, e.g. performance of an AMS circuit, which can be costly, are only made at these optimized locations, e.g. via circuit simulations for AMS verification. The new data collected at each step augments the training dataset to retrain a probabilistic surrogate model that approximates the black-box function. Such iterative sampling scheme contributes directly to the accuracy of the surrogate model and guides the iterative global optimization process [31, 83].

3.2 Hotspot Detection

As the feature size of semiconductor transistors continues shrinking, the gap between exploding design demands and semiconductor manufacturability using current mainstream 193 nm lithography is becoming wider. Various designs for manufacturability (DFM) techniques have been proposed; however, due to the complexity of lithography systems and process variation, failures to print specific patterns still happen, which are referred to as lithography hotspots. Examples of two hotspot patterns are shown in Fig. 8.

The hotspot detection problem is to locate the lithography hotspots on a given layout in physical design and verification stages. Conventional simulation-based hotspot detection often relies on accurate yet complicated lithography models and therefore is extremely time-consuming. Efficient and accurate lithography hotspot detection is more desired for layout finishing and design closure in advanced technology nodes.

Pattern matching and machine learning based techniques have been proposed for quick and accurate detection of hotspots. Pattern matching forms a predefined library of hotspot layout patterns, and then compares any new pattern with the patterns in the library [70, 79]. There are some extensions that use fuzzy pattern matching to increase the coverage of the library [41, 66]. However, pattern matching,

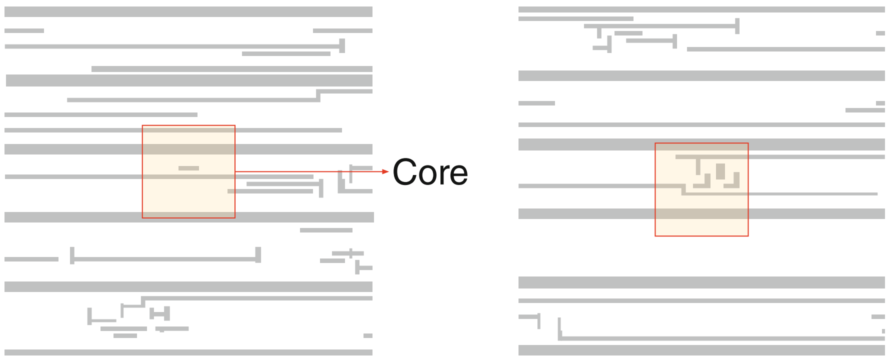


Fig. 8 Example of two hotspot patterns. Core corresponds to the central location where a hotspot appears

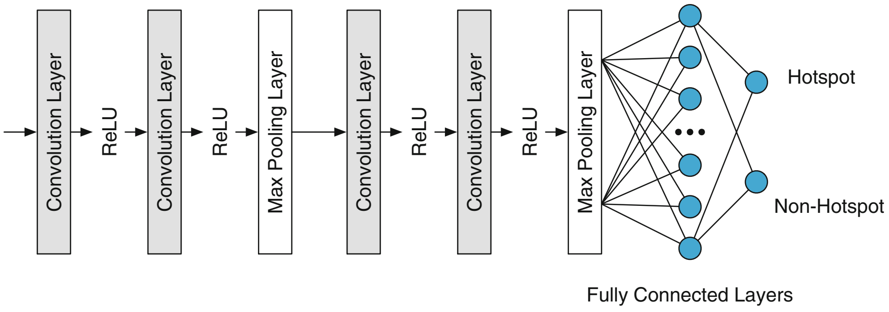


Fig. 9 An example of a neural network for hotspot detection [74]

including fuzzy pattern matching, is insufficient to handle never-before-seen hotspot patterns. Recently, machine learning based approaches have demonstrated good generalization capability to recognize unseen hotspot patterns [17, 18, 45, 50, 80, 82].

3.2.1 Lithography Hotspot Detection with Machine Learning Models

Various machine learning models have been used as hotspot detection kernels with the goal of achieving high accuracy and low false alarms, including support vector machine (SVM) [18, 80], artificial neural network (ANN) [18], and boosting methods [45, 82]. Zhang et al. [82] have also proposed an online learning scheme to verify newly detected hotspots and incrementally update the model. Recently, deep neural networks (DNNs) have been adopted for hotspot detection [46, 60]. DNNs are able to perform automatic feature extraction on the high-dimensional layout during training, which spares the efforts spent on manual feature extraction. Promising empirical results have been observed with DNNs in several papers [46, 60, 73, 74]. Figure 9 shows a typical configuration of the structure of a DNN.

The performance of DNNs usually relies heavily on manual efforts to tune the networks, e.g., the number and types of layers. Matsunawa et al. [46] proposed a DNN structure for hotspot detection that can achieve low false alarms. Yang et al. [74] proposed Discrete Cosine Transform (DCT) based feature representation to reduce the image size for DNNs with a biased learning to improve accuracy and decrease false alarms.

3.2.2 Evaluation of Hotspot Detection Models

One special characteristic of lithography hotspot detection tasks is the imbalance in the layout datasets. Those lithography defects are critical, but their relative number is significantly small across the whole chip. Among various machine learning

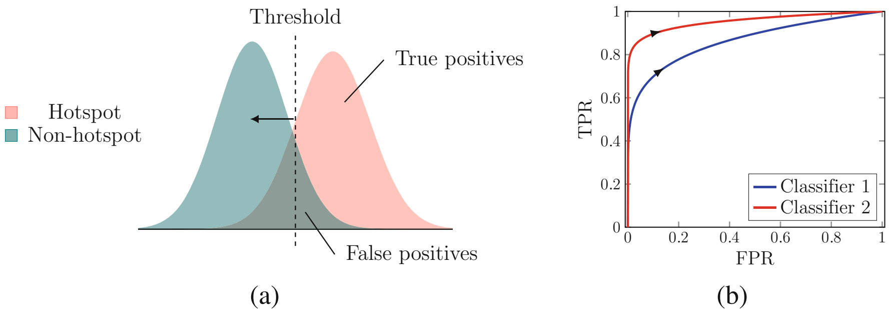


Fig. 10 (**a**) An overlapping distribution of predicted scores for positive and negative samples and (**b**) the ROC curves of two example classifiers. As the threshold in (**a**) moves to the left, both FPR and TPR in (**b**) go up accordingly [77]

models at hand, the one with a highest true positive rate (TPR) and a lowest false positive rate (FPR) is preferred, but in real-world scenarios, there is always a trade-off between the two metrics. As Fig. 10a demonstrates, if the predicted score implies the belief of the classifier that a sample belongs to the positive class, decreasing the decision threshold (i.e., moving the threshold to the left) will increase both TPRs and FPRs.

The receiver operating characteristic (ROC) curve is considered a robust performance evaluation and model selection metric for imbalanced learning problems. For each setting of the decision threshold of a binary classification model (Fig. 10a), a pair of TPR and FPR values is obtained. By varying the decision threshold over the range [0, 1], the ROC curve plots the relationship between TPR and the FPR (Fig. 10b).

The area under the ROC curve (AUC) is a threshold-independent metric which measures the fraction of times a positive instance is ranked higher than a negative one [62]. The closer the curve is pulled towards the upper left corner, the better is the ability of the classifier to discriminate between the two classes. For example, in Fig. 10b, classifier 2 has a better performance compared to classifier 1. Given that AUC is a robust measure of classification performances especially for imbalanced problems, it is useful to devise algorithms that directly optimize this metric during the training phase.

It has been proven that AUC is equivalent to the Wilcoxon–Mann–Whitney (WMW) statistic test of ranks [28, 44, 67]. However, AUC defined by the WMW metric is a sum of indicator functions which is not differentiable, to which gradient-based optimization methods cannot be applied. In order to make the problem tractable, it is necessary to apply convex relaxation to the AUC by replacing the indicator function with pairwise convex surrogate loss function. There are different forms of surrogate functions: pairwise squared loss [19, 24], pairwise hinge loss [61, 84], pairwise logistic loss [56], and piecewise function given in [71]. Ye et al. [77] compare these surrogate functions and show that those new surrogate loss functions are promising to outperform the cross-entropy loss when applied to the state-of-the-art neural network model for hotspot detection.

3.2.3 Data Efficient Hotspot Detection

Despite the effective machine learning models for hotspot detection, most of them rely on a large amount of data for training, resulting in huge data preparation overhead. Thus, it is necessary to improve the data efficiency during model training, i.e., to achieve high accuracy with as small amount of data as possible.

Chen et al. [15] proposed to leverage the information in unlabeled data during model training, when the amount of labeled data is small. They develop a semi-supervised learning framework, using a multi-task network with two branches to train the classification task for hotspot detection and the other unsupervised clustering task at the same time. The network will label those unlabeled data samples with pseudo-labels in each iteration. The pseudo-labeled data will be selected and added to training with different weights in the next iteration, where the weights here are determined by the clustering branch. The experimental results demonstrate over 3–4% accuracy improvement with 10%–50% amount of labeled training data.

Sometimes, there is additional flexibility to the learning problem where labels for unlabeled data be can queried. This extra capability enables the use of active learning which can actively select the data samples for training a better model. Yang et al [72] propose to iteratively query the actual labels for unlabeled data samples with low classification confidence in each training step and add these samples for training in the next step. The experiments on ICCAD 2016 contest benchmarks show similar accuracy with only 17% of training data samples.

One should note that semi-supervised learning and active learning are two orthogonal approaches to tackle the insufficient of labeled training data. Semi-supervised learning assumes the availability of unlabeled data, while active learning assumes the capability of querying the labels for unlabeled data. They can even be combined to achieve better data efficiency [85].

3.2.4 Trustworthiness of Hotspot Detection Models

Conventionally, hotspot detection approaches have been evaluated by judging upon the detection accuracy and the false alarm rate. While these metrics are indeed important, model trustworthiness is yet another metric that is critical for adopting machine learning based approaches. Addressing this concern requires machine learning models to provide confidence guarantees alongside the label predictions.

In practice, methods for obtaining confidence guarantees when using deep neural network are costly and not yet mature. However, Bayesian-based methods are the typical option when confidence estimation is needed. This can be achieved by adopting a Gaussian Process (GP) based classification that can provide a confidence metric for each predicted instance. With this approach, a label from a trained model is only valid when its confidence level matches a user-defined metric, otherwise, the prediction is marked as untrusted and lithography simulation can be used to further verify the results [75].

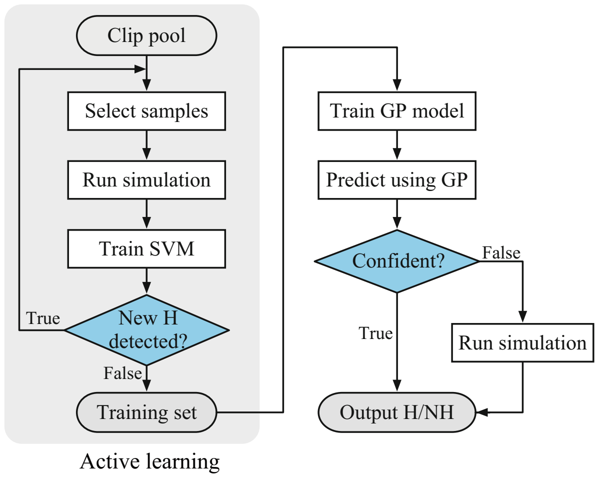


Fig. 11 Overall flow of Litho-GPA including data preparation with active sampling and hotspot detection with Gaussian process [75]

The flow of Litho-GPA, a framework for hotspot detection with Gaussian Process assurance, is illustrated in Fig. 11. In addition to addressing the issue of trust, Litho-GPA adopts active learning to reduce the amount of training data while favoring balance between classes in this dataset.

As a first step, an iterative weak classifier-based sampling scheme is leveraged to prepare a training set containing enough hotspots. Next, a Gaussian Process Regression (GPR) model is trained for the classification task with the selected data samples. This learned model is then used to make predictions with confidence estimation on the testing set. If GPR demonstrated high confidence in the predicted label, the result is trusted; otherwise, the unsure testing samples are verified with lithography simulations.

Experimental results shown in [75] demonstrate Litho-GPA can achieve comparable accuracy to the state-of-the-art deep learning approaches while obtaining on average 28% reduction in false alarms.

4 Conclusion

In this chapter, different important aging and yield issues in modern VLSI design and manufacturing have been discussed. These issues include device aging, interconnect electromigration, process variation, and manufacturing defects are likely to cause severe performance degradation or functionality failure, and thus need to be addressed early in the physical design flow. The chapter has surveyed recent techniques to not only build models for capturing these effects, but also to develop strategies for optimizing them with the proposed models. These practices demonstrate that synergistic optimization and cross-layer feedback are encouraged to resolve the aforementioned aging and yield issues for robust VLSI design cycles.

References

1. Addressing signal electromigration (EM) in today's complex digital designs. https://www.eetimes.com/document.asp?docid=1280370
2. Agarwal, M., Paul, B.C., Zhang, M., Mitra, S.: Circuit failure prediction and its application to transistor aging. In: 25th IEEE VLSI Test Symposium (VTS'07), pp. 277–286 (2007)
3. Alawieh, M.B., Tang, X., Pan, D.: S^2PM: semi-supervised learning for efficient performance modeling of analog and mixed signal circuit. In: IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC), pp. 268–273 (2019)
4. Alawieh, M.B., Wang, F., Kang, R., Li, X., Joshi, R.: Efficient analog circuit optimization using sparse regression and error margining. In: IEEE International Symposium on Quality Electronic Design (ISQED), pp. 410–415 (2016)
5. Alawieh, M.B., Wang, F., Li, X.: Efficient hierarchical performance modeling for analog and mixed-signal circuits via bayesian co-learning. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **37**(12), 1–13 (2018)
6. Alawieh, M.B., Wang, F., Tao, J., Yin, S., Jun, M., Li, X., Mukherjee Tamal Negi, R.N.: Efficient programming of reconfigurable radio frequency (RF) systems. In: IEEE/ACM International Conference on Computer-Aided Design (ICCAD), pp. 772–779 (2017)
7. Alawieh, M.B., Williamson, S.A., Pan, D.Z.: Rethinking sparsity in performance modeling for analog and mixed circuits using spike and slab models In: ACM/IEEE Design Automation Conference (DAC) (2019)
8. Alpert, C.J., Mehta, D.P., Sapatnekar, S.S.: Handbook of Algorithms for Physical Design Automation. CRC press, New York (2008)
9. Amrouch, H., Khaleghi, B., Gerstlauer, A., Henkel, J.: Reliability-aware design to suppress aging. In: ACM/IEEE Design Automation Conference (DAC), pp. 1–6 (2016)
10. Arnaud, L., Tartavel, G., Berger, T., Mariolle, D., Gobil, Y., Touet, I.: Microstructure and electromigration in copper damascene lines. Microelectron. Reliab. **40**(1), 77–86 (2000)
11. Black, J.R.: Electromigration—a brief survey and some recent results. IEEE Trans. Electron Devices **16**(4), 338–347 (1969)
12. C. Hu S. Tam, F.H.P.K.T.C., Terrill, K.W.: Hot-electron-induced MOSFET degradation—model, monitor, and improvement. IEEE J. Solid-State Circuits **20**(1), 295–305 (1985)
13. Chakraborty, A., Ganesan, G., Rajaram, A., Pan, D.Z.: Analysis and optimization of NBTI induced clock skew in gated clock trees. In: IEEE/ACM Proceedings Design, Automation and Test in Europe (DATE), pp. 296–299 (2009)
14. Chen, X., Liao, C., Wei, T., Hu, S.: An interconnect reliability-driven routing technique for electromigration failure avoidance. IEEE Trans. Dependable Secure Comput. **9**(5), 770–776 (2012)
15. Chen, Y., Lin, Y., Gai, T., Su, Y., Wei, Y., Pan, D.Z.: Semi-supervised hotspot detection with self-paced multi-task learning. In: IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC), pp. 420–425 (2019)
16. Chen Ih-Chin, C.J.Y., Chenming, H.: The effect of channel hot-carrier stressing on gate-oxide integrity in MOSFETs. IEEE Trans. Electron Devices **35**(12), 2253–2258 (1988)
17. Ding, D., Torres, J.A., Pan, D.Z.: High performance lithography hotspot detection with successively refined pattern identifications and machine learning. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **30**(11), 1621–1634 (2011)

18. Ding, D., Yu, B., Ghosh, J., Pan, D.Z.: EPIC: efficient prediction of IC manufacturing hotspots with a unified meta-classification formulation. In: IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC), pp. 263–270 (2012)
19. Ding, Y., Zhao, P., Hoi, S.C.H., Ong, Y.S.: An adaptive gradient method for online AUC maximization. In: AAAI Conference on Artificial Intelligence, pp. 2568–2574 (2015)
20. Ebrahimi, M., Oboril, F., Kiamehr, S., Tahoori, M.B.: Aging-aware logic synthesis. In: IEEE/ACM International Conference on Computer-Aided Design (ICCAD), pp. 61–68 (2013)
21. Fang, J., Sapatnekar, S.S.: The impact of hot carriers on timing in large circuits. In: 17th Asia and South Pacific Design Automation Conference, pp. 591–596 (2012)
22. Fang, J., Sapatnekar, S.S.: The impact of BTI variations on timing in digital logic circuits. IEEE Trans. Device Mater. Reliab. **13**(1), 277–286 (2013)
23. Firouzi, F., Kiamehr, S., Tahoori, M.B.: NBTI mitigation by optimized NOP assignment and insertion. In: 2012 Design, Automation Test in Europe Conference Exhibition (DATE), pp. 218–223 (2012)
24. Gao, W., Jin, R., Zhu, S., Zhou, Z.H.: One-pass AUC optimization. In: International Conference on Machine Learning (ICML), pp. III906–III914 (2013)
25. Grasser, T., Kaczer, B., Goes, W., Reisinger, H., Aichinger, T., Hehenberger, P., Wagner, P., Schanovsky, F., Franco, J., Roussel, P., Nelhiebel, M.: Recent advances in understanding the bias temperature instability. In: 2010 International Electron Devices Meeting, pp. 4.4.1–4.4.4 (2010). doi:10.1109/IEDM.2010.5703295
26. Grasser, T., Kaczer, B., Goes, W., Reisinger, H., Aichinger, T., Hehenberger, P., Wagner, P.J., Schanovsky, F., Franco, J., Luque, M.T., et al.: The paradigm shift in understanding the bias temperature instability: from reaction–diffusion to switching oxide traps. IEEE Trans. Electron Devices **58**(11), 3652–3666 (2011)
27. Guerin, C., Huard, V., Bravaix, A.: The energy-driven hot-carrier degradation modes of nMOSFETs. IEEE Trans. Device Mater. Reliab. **7**(2), 225–235 (2007)
28. Hanley, J.A., McNeil, B.J.: The meaning and use of the area under a receiver operating characteristic (roc) curve. Radiology **143**(1), 29–36 (1982)
29. Hsu, C., Guo, S., Lin, Y., Xu, X., Li, M., Wang, R., Huang, R., Pan, D.Z.: Layout-dependent aging mitigation for critical path timing. In: IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC), pp. 153–158 (2018)
30. Hsu, M.K., Katta, N., Lin, H.Y.H., Lin, K.T.H., Tam, K.H., Wang, K.C.H.: Design and manufacturing process co-optimization in nano-technology. In: IEEE/ACM International Conference on Computer-Aided Design (ICCAD), pp. 574–581 (2014)
31. Hu, H., Li, P., Huang, J.Z.: Enabling high-dimensional Bayesian optimization for efficient failure detection of analog and mixed-signal circuits. In: ACM/IEEE Design Automation Conference (DAC) (2019)
32. Huard, V., Parthasarathy, C., Bravaix, A., Guerin, C., Pion, E.: CMOS device design-in reliability approach in advanced nodes. In: 2009 IEEE International Reliability Physics Symposium, pp. 624–633 (2009)
33. Jiang, I.H.R., Chang, H.Y., Chang, C.L.: WiT: optimal wiring topology for electromigration avoidance. IEEE Trans. Very Large Scale Integr. Syst. **20**(4), 581–592 (2012)
34. Kaczer, B., Mahato, S., Valduga de Almeida Camargo, V., Toledano-Luque, M., Roussel, P.J., Grasser, T., Catthoor, F., Dobrovolny, P., Zuber, P., Wirth, G., Groeseneken, G.: Atomistic approach to variability of bias-temperature instability in circuit simulations. In: 2011 International Reliability Physics Symposium, pp. XT.3.1–XT.3.5 (2011)
35. Kumar, S.V., Kim, C.H., Sapatnekar, S.S.: NBTI aware synthesis of digital circuits. In: ACM/IEEE Design Automation Conference (DAC), pp. 370–375 (2007)
36. Li, B., Muller, P., Warnock, J., Sigal, L., Badami, D.: A case study of electromigration reliability: from design point to system operations. In: IEEE International Reliability Physics Symposium (IRPS), pp. 2D.1.1–2D.1.6 (2015)
37. Li, X.: Finding deterministic solution from underdetermined equation: large-scale performance modeling by least angle regression. In: ACM/IEEE Design Automation Conference (DAC), pp. 364–369 (2009)

38. Lienig, J.: Electromigration and its impact on physical design in future technologies. In: ACM International Symposium on Physical Design (ISPD), pp. 33–40 (2013)
39. Lienig, J., Thiele, M.: Fundamentals of Electromigration-Aware Integrated Circuit Design. Springer, Berlin (2018)
40. Liew, B.K., Cheung, N.W., Hu, C.: Projecting interconnect electromigration lifetime for arbitrary current waveforms. IEEE Trans. Electron Devices **37**(5), 1343–1351 (1990)
41. Lin, S.Y., Chen, J.Y., Li, J.C., Wen, W.Y., Chang, S.C.: A novel fuzzy matching model for lithography hotspot detection. In: ACM/IEEE Design Automation Conference (DAC), pp. 68:1–68:6 (2013)
42. Lin, Y., Alawieh, M.B., Ye, W., Pan, D.: Machine learning for yield learning and optimization. In: IEEE International Test Conference (ITC), pp. 1–10 (2018)
43. Lorenz, D., Georgakos, G., Schlichtmann, U.: Aging analysis of circuit timing considering NBTI and HCI. In: 2009 15th IEEE International On-Line Testing Symposium, pp. 3–8 (2009)
44. Mann, H.B., Whitney, D.R.: On a test of whether one of two random variables is stochastically larger than the other. Ann. Math. Statist. **18**(1), 50–60 (1947)
45. Matsunawa, T., Gao, J.R., Yu, B., Pan, D.Z.: A new lithography hotspot detection framework based on AdaBoost classifier and simplified feature extraction. In: Proceedings of SPIE, vol. 9427 (2015)
46. Matsunawa, T., Nojima, S., Kotani, T.: Automatic layout feature extraction for lithography hotspot detection based on deep neural network. In: SPIE Advanced Lithography, vol. 9781 (2016)
47. Nigam, T., Parameshwaran, B., Krause, G.: Accurate product lifetime predictions based on device-level measurements. Proceedings of the 2009 IEEE International Reliability Physics Symposium, pp. 634–639 (2009)
48. Oboril, F., Tahoori, M.B.: ExtraTime: modeling and analysis of wearout due to transistor aging at microarchitecture-level. In: IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), pp. 1–12 (2012)
49. de Paris, L., Posser, G., Reis, R.: Electromigration aware circuits by using special signal non-default routing rules. In: IEEE International Symposium on Circuits and Systems (ISCAS), pp. 2795–2798 (2016)
50. Park, J.W., Torres, A., Song, X.: Litho-aware machine learning for hotspot detection. IEEE Trans. Comput. Aided Design Integr. Circuits Syst. **37**(7), 1510–1514 (2018)
51. Pecht, M., Radojcic, R., Rao, G.: Guidebook for managing silicon chip reliability. CRC Press, New York (2017)
52. Rauch, S.E., La Rosa, G.: The energy-driven paradigm of nMOSFET hot-carrier effects. IEEE Trans. Device Mater. Reliab. **5**(4), 701–705 (2005)
53. Reis, R., Cao, Y., Wirth, G.: Circuit design for reliability. Springer, Berlin (2015)
54. Ren, P., Xu, X., Hao, P., Wang, J., Wang, R., Li, M., Wang, J., Bu, W., Wu, J., Wong, W., Yu, S., et al.: Adding the missing time-dependent layout dependency into device-circuit-layout co-optimization: new findings on the layout dependent aging effects. In: IEEE International Electron Devices Meeting (IEDM) (2015)
55. Roy, S., Pan, D.Z.: Reliability aware gate sizing combating NBTI and oxide breakdown. In: International Conference on VLSI Design, pp. 38–43 (2014)
56. Rudin, C., Schapire, R.E.: Margin-based ranking and an equivalence between AdaBoost and RankBoost. J. Mach. Learn. Res. **10**(Oct), 2193–2232 (2009)
57. Tam, S., Ko, P.K., Hu, C.: Lucky-electron model of channel hot-electron injection in MOSFET's. IEEE Trans. Electron Devices **31**(9), 1116–1125 (1984)
58. Sapatnekar, S.S.: What happens when circuits grow old: aging issues in CMOS design. In: 2013 International Symposium on VLSI Technology, Systems and Application (VLSI-TSA), pp. 1–2 (2013)
59. Sengupta, D., Sapatnekar, S.S.: Estimating circuit aging due to BTI and HCI using ring-oscillator-based sensors. IEEE Trans. Comput. Aided Design Integr. Circuits Syst. **36**(10), 1688–1701 (2017)

60. Shin, M., Lee, J.H.: Accurate lithography hotspot detection using deep convolutional neural networks. J. Micro/Nanolithogr. MEMS MOEMS **15**(4), 043507 (2016)
61. Steck, H.: Hinge rank loss and the area under the ROC curve. In: European Conference on Machine Learning, pp. 347–358. Springer, Berlin (2007)
62. Swets, J.A., Pickett, R.M.: Evaluation of diagnostic systems: methods from signal detection theory. Academic Press, New York (1982)
63. Ting, L., May, J., Hunter, W., McPherson, J.: AC electromigration characterization and modeling of multilayered interconnects. In: IEEE International Reliability Physics Symposium (IRPS), pp. 311–316 (1993)
64. Vattikonda, R., Wang, W., Cao, Y.: Modeling and minimization of PMOS NBTI effect for robust nanometer design. In: ACM/IEEE Design Automation Conference (DAC), pp. 1047–1052 (2006)
65. Wang, Y., Chen, X., Wang, W., Cao, Y., Xie, Y., Yang, H.: Leakage power and circuit aging cooptimization by gate replacement techniques. IEEE Trans. Very Large Scale Integr.(VLSI) Syst. **19**(4), 615–628 (2011)
66. Wen, W.Y., Li, J.C., Lin, S.Y., Chen, J.Y., Chang, S.C.: A fuzzy-matching model with grid reduction for lithography hotspot detection. IEEE Trans. Comput. Aided Design Integr. Circuits Syst. **33**(11), 1671–1680 (2014)
67. Wilcoxon, F.: Individual comparisons by ranking methods. Biom. Bullet. **1**(6), 80–83 (1945)
68. Wu, K.C., Marculescu, D.: Joint logic restructuring and pin reordering against NBTI-induced performance degradation. In: IEEE/ACM Proceedings Design, Automation and Test in Europe (DATE), pp. 75–80 (2009)
69. Xie, J., Narayanan, V., Xie, Y.: Mitigating electromigration of power supply networks using bidirectional current stress. In: ACM Great Lakes Symposium on VLSI (GLSVLSI), pp. 299–302 (2012)
70. Xu, J., Sinha, S., Chiang, C.C.: Accurate detection for process-hotspots with vias and incomplete specification. In: IEEE/ACM International Conference on Computer-Aided Design (ICCAD), pp. 839–846 (2007)
71. Yan, L., Dodier, R.H., Mozer, M., Wolniewicz, R.H.: Optimizing classifier performance via an approximation to the Wilcoxon-Mann-Whitney statistic. In: International Conference on Machine Learning (ICML), pp. 848–855 (2003)
72. Yang, H., Li, S., Tabery, C., Lin, B., Yu, B.: Bridging the gap between layout pattern sampling and hotspot detection via batch active sampling (2018). arXiv preprint:1807.06446
73. Yang, H., Luo, L., Su, J., Lin, C., Yu, B.: Imbalance aware lithography hotspot detection: a deep learning approach. In: SPIE Advanced Lithography, vol. 10148 (2017)
74. Yang, H., Su, J., Zou, Y., Yu, B., Young, E.F.Y.: Layout hotspot detection with feature tensor generation and deep biased learning. In: ACM/IEEE Design Automation Conference (DAC), pp. 62:1–62:6 (2017)
75. Ye, W., Alawieh, M.B., Li, M., Lin, Y., Pan, D.Z.: Litho-GPA: Gaussian process assurance for lithography hotspot detection. In: IEEE/ACM Proceedings Design, Automation and Test in Europe (DATE) (2019)
76. Ye, W., Alawieh, M.B., Lin, Y., Pan, D.Z.: Tackling signal electromigration with learning-based detection and multistage mitigation. In: IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC), pp. 167–172 (2019)
77. Ye, W., Lin, Y., Li, M., Liu, Q., Pan, D.Z.: LithoROC: lithography hotspot detection with explicit ROC optimization. In: IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC), pp. 292–298 (2019)
78. Ye, W., Lin, Y., Xu, X., Li, W., Fu, Y., Sun, Y., Zhan, C., Pan, D.Z.: Placement mitigation techniques for power grid electromigration. In: IEEE International Symposium on Low Power Electronics and Design (ISLPED) (2017)
79. Yu, Y.T., Chan, Y.C., Sinha, S., Jiang, I.H.R., Chiang, C.: Accurate process-hotspot detection using critical design rule extraction. In: ACM/IEEE Design Automation Conference (DAC), pp. 1167–1172 (2012)

80. Yu, Y.T., Lin, G.H., Jiang, I.H.R., Chiang, C.: Machine-learning-based hotspot detection using topological classification and critical feature extraction. IEEE Trans. Comput. Aided Design Integr. Circuits Syst. **34**(3), 460–470 (2015)
81. Zhang, H., Kochte, M.A., Schneider, E., Bauer, L., Wunderlich, H., Henkel, J.: Strap: Stress-aware placement for aging mitigation in runtime reconfigurable architectures. In: 2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), pp. 38–45 (2015)
82. Zhang, H., Zhu, F., Li, H., Young, E.F.Y., Yu, B.: Bilinear lithography hotspot detection. In: ACM International Symposium on Physical Design (ISPD), pp. 7–14 (2017)
83. Zhang, S., Lyu, W., Wang, F., Yan, C., Hu, X., Zeng, X.: An efficient multi-fidelity Bayesian optimization approach for analog circuit synthesis. In: ACM/IEEE Design Automation Conference (DAC) (2019)
84. Zhao, P., Hoi, S.C.H., Jin, R., Yang, T.: Online AUC maximization. In: International Conference on Machine Learning (ICML), pp. 233–240 (2011)
85. Zhu, X., Lafferty, J., Ghahramani, Z.: Combining active learning and semi-supervised learning using Gaussian fields and harmonic functions. In: ICML 2003 workshop on the continuum from labeled to unlabeled data in machine learning and data mining, vol. 3 (2003)

Part V
Cross-Layer from Architecture to Application

Michael Glaß

Possibly since the proposal of the *von Neumann architecture*, there exists an invisible barrier between hardware and software in the narrower or architecture and application in the broader sense. Given that powerful compilers have become omnipresent for almost all architectures and languages, application developers are able to treat the architecture layer and respective system components as black boxes—as long as the architecture *overprovides* in the sense that requirements of the application can be *trivially* fulfilled. Two well-known requirements that have questioned this view in the past are performance and energy considerations. The former gave rise to DSPs and GPUs to name two exemplary architectures that address the special needs of complete application domains while at the same time, the applications have to be adapted to efficiently make use of the architectures and avoid performance bottlenecks. The latter—together with real-time requirements—fueled hardware/software co-design in the embedded system domain where individual applications and their requirements are tailored and deployed to typically multiple components featuring different architectures.

In the area of dependable—emphasis on reliable, available, and safe—systems, respective and often stringent requirements have traditionally been addressed by either (a) architectures/components that give guarantees on dependability properties and achieve these by significant conservative margins and guard banding or by (b) system-wide mitigation strategies like dual or triple modular redundancy schemes. In these domains, one can assume dependability to be a prime, if not the prime, design objective while accepting to sacrifice other objectives such as energy, area consumption, or monetary costs.

M. Glaß
Ulm University, Institute of Embedded Systems/Real-Time Systems, Ulm, Germany;
michael.glass@uni-ulm.de

Continuous technology scaling with all its tremendous advantages has at the same time threatened the dependability of CMOS devices: Increasing susceptibility to aging and radiation effects combined with significant variation has reached a point where hiding such effects from the application completely may require excessive and—for many domains and especially embedded systems—uneconomical overheads. At the same time, mentioned system-wide coarse-grained mitigation techniques may be prohibitive due to other important design objectives for embedded systems such as energy budgets, space constraints, and the like. Assuming that errors at architecture layer may have to be exposed to the application, an interesting and key observation is that not all parts of an application are equally prone to errors on component/architecture layer and are equally costly to harden.

In this area of the book at hand, seven chapters present their endeavors to analyze and exploit architectural as well as application properties concurrently to form *cross-layer* approaches that combine architecture and application layer. At the center of this area is the interplay of specific application (domain) properties on the one and the computational and memory elements on the other hand. The area and its chapters are organized as follows:

The following two chapters put focus on the interplay of applications and memory: chapter “Exploiting Memory Resilience for Emerging Technologies: An Energy-Aware Resilience Exemplar for STT-RAM Memories” investigates Spin Transfer Torque Magnetic RAM (STT-RAM) as a technology which may replace SRAM in near-future devices. However, the susceptibility of STT-RAMs to errors especially during write operations depends on the state transition as well as the applied current level. The chapter presents an architectural scheme for STT-RAM cache memories that dynamically profiles the use of each individual bit by the respective application to select cache way and current level; guaranteeing a maximum error rate while minimizing energy consumption. chapter “Design of Efficient, Dependable SoCs Based on a Cross-Layer-Reliability Approach with Emphasis on Wireless Communication as Application and DRAM Memories” investigates inherently error-resilient applications—particularly wireless baseband processing—and proposes to treat hardware errors in a similar fashion as transmission errors over a noisy channel by means of the so-called dynamic resilience actuators. Since wireless baseband applications are not only compute, but can as well be memory intensive, their error resiliency is further exploited by a proposed approximate DRAM technique which trades-off refresh rate and, thus, energy consumption, for reliability according to the robustness of the application.

The next three chapters focus on variation when deploying applications to multi-/many-core systems, covering dependable dynamic resource management, uncertainty-aware reliability analysis, as well as approximate computing: chapter “Power-Aware Fault-Tolerance for Embedded Systems” puts focus on applications to be deployed to multi-/many-core chips under thermal design power constraints. The addressed challenge is to provide an optimal mix of hardware and software hardening techniques for an application such that reliability goals are met without violating power and/or performance constraints. As a remedy, the chapter introduces a power-reliability management technique that combines design-time analysis and optimization as well as run-time adaptation to account for variation in performance, occurring faults, and power. Chapter "Uncertainty-Aware Compositional System-Level Reliability Analysis" addresses the problem that uncertainties arising from manufacturing tolerances, environmental influences, or application inputs may not be known at design time such that respective worst-case assumptions result in extremely pessimistic reliability analysis results. As a remedy, the chapter presents a compositional and uncertainty-aware reliability analysis approach that explicitly models uncertainties and, thus, exposes not only best-case or worst-case values but probability distributions. These can enhance cost-efficient error mitigation by quantifying the probability of different best-/average-/worst-case situations. Chapter "Hardware/Software Codesign for Energy Efficiency and Robustness: From Error-Tolerant Computing to Approximate Computing" puts focus on the margins that are applied to compensate for the ever-increasing variability. The chapter first presents an approach to drastically reduce margins for GPUs by means of an adaptive compiler that aims at equalizing the lifetime of each processing element. Then, the reduction of margins is pushed so far that errors have to accepted and computations become approximations for which the chapter proposes an automatic FPGA design flow for accelerators that employ such approximate computations.

The last two chapters in this area put their attention to two important application domains in current and future embedded systems: Cyber-physical systems and machine learning. Chapter "Reliable CPS Design for Unreliable Hardware Platforms" investigates cyber-physical systems and especially battery-operated systems where control loops are the key part of their applications. While such control loops traditionally focus on certain metrics such as stability or overshoot, they may affect system components such as the batteries themselves as well as the processing components the applications are deployed to by accelerating their aging. As a remedy, the chapter proposes a design flow that combines an optimization of (a) quality-of-control and battery behavior at design time as well as (b) quality-of-control and processor aging at run-time to satisfy safety requirements. Chapter "Robust Computing for Machine Learning-Based Systems" investigates machine learning approaches as key enablers for various upcoming safety-critical applications from the domain of autonomous systems. Especially in this context, the chapter investigates the susceptibility of these both intrinsically compute and memory-intensive applications to reliability as well as security threats. The chapter discusses techniques to enhance the robustness/resilience of machine learning applications and outlines open research challenges in this domain.

Design of Efficient, Dependable SoCs Based on a Cross-Layer-Reliability Approach with Emphasis on Wireless Communication as Application and DRAM Memories

Christian Weis, Christina Gimmler-Dumont, Matthias Jung, and Norbert Wehn

1 Introduction

Technology scaling has reached a point at which process and environmental variabilities are no longer negligible, and can no longer be hidden from system designers, as the exact behavior of CMOS devices becomes increasingly less predictable. This will show in the form of static and dynamic variations, time-dependent device degradation and early life failures, sporadic timing errors, radiation-induced soft errors, and lower resilience to varying operating conditions [35]. Already today, conservative margining, guardbanding, and conservative voltage scaling, come at a large cost. Only turning away from conservative worst-case design methodologies for a 100% reliable physical hardware layer will make further downscaling of CMOS technologies a profitable endeavor [7, 32]. This calls for radically new cross-layer-design concepts [14–16] (Fig. 1).

Until today, these problems have mostly been addressed at the lower design levels. At the higher levels, systems are typically designed under the premise of fault-free underlying hardware. Only in extremely critical applications, such as avionics, where the system cost is less important than its dependability, triple modular redundancy (TMR) and similar techniques are employed on a system level. Thus, to no big surprise, the large body of related work focuses on low-level

C. Weis (✉) · N. Wehn
TU Kaiserslautern, Kaiserslautern, Germany
e-mail: weis@eit.uni-kl.de; wehn@eit.uni-kl.de

C. Gimmler-Dumont
European Patent Office, Bruxelles, Belguim
e-mail: cgimmlerdumont@epo.org

M. Jung
Fraunhofer Institute for Experimental Software Engineering, Kaiserslautern, Germany
e-mail: Matthias.Jung@iese.fraunhofer.de

J. Henkel, N. Dutt (eds.), *Dependable Embedded Systems*, Embedded Systems,
https://doi.org/10.1007/978-3-030-52017-5_18

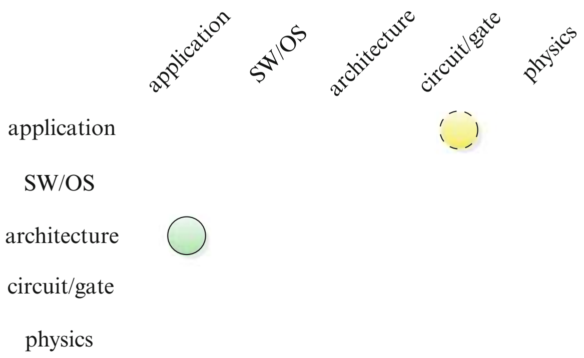


Fig. 1 Main abstraction layers of embedded systems as used in the SPP1500 Dependable Embedded Systems Priority Program of the German Research Foundation (DFG) and this chapter's major (green, solid) and minor (yellow, dashed) cross-layer contributions

techniques to present higher abstraction and design levels with a practically error-free platform built from potentially faulty elements.

To make a platform resilient against transient or permanent faults built-in redundancy or built-in self-recovery techniques have to be employed. They all come at the cost of chip area, power consumption, reduced system throughput, or other implementation related metrics. Lower implementation cost, especially with regard to energy consumption can be obtained when a degradation of hardware reliability to a certain degree is tolerated. In fact, *energy consumption and dependability of integrated circuits can be seen as strongly interrelated problems*: by decreasing the operating voltage, the energy efficiency increases but at the same time the dependability decreases. Thus, energy efficiency and dependability have to be carefully traded off against each other.

An error-resilient architecture that can be seen as practically error-free can be composed of protected components. Applications for these platforms can be implemented in a traditional way, still assuming fault-free operation of the underlying hardware. In addition to this horizontal integration, recent research also evaluates the additional potential of a vertical integration of error resilience on the application level with platforms having a reduced reliability. True *cross-layer optimization* approaches do not only exploit the fact that some important classes of algorithms are inherently error-tolerant, but also adapt applications and hardware architectures jointly to achieve the best possible trade-offs. This chapter focuses on cross-layer optimization for wireless communication systems with emphasis on errors on data path and SRAMs. Furthermore, we consider undependable DRAM subsystems, named *approximate DRAM*.

2 Wireless Baseband Processing

In this section, we present a wireless baseband processing system and a novel cross-layer methodology using resilience actuators (see Sect. 2.1.2) to improve the reliability of this system. Wireless communication systems have an inherent error

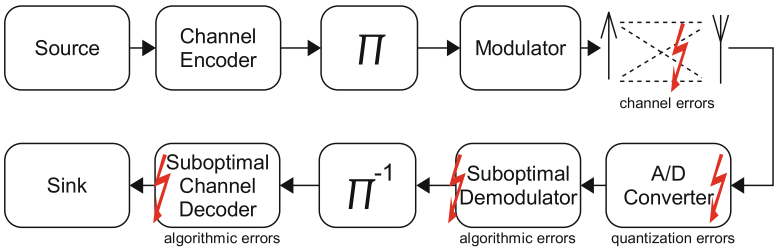


Fig. 2 Wireless communication systems suffer from different sources of errors (indicated by red arrows): channel errors, quantization errors, errors from suboptimal algorithms, and hardware errors. Here, we focus on the receiver side only

resilience. They are designed to recover the originally transmitted data sequence in spite of errors that occur during transmission over a noisy channel. Figure 2 shows a simplified structure of such a system. To achieve a reliable transmission, today's communication systems use advanced forward error correction (FEC) techniques, i.e. the sender adds redundancy to the actual information prior to transmission in the channel encoder. This encoder connects to the modulator via an interleaver (Π). The interleaver is required to break dependencies between neighboring bits while the modulator performs the mapping on symbols (e.g. QAM—quadrature amplitude modulation) that are transmitted over the physical channel. On the receiver side, the noisy signal is converted to the digital domain and fed into the demodulator, which recovers the originally transmitted symbols by exploiting the channel characteristics. After the deinterleaver (Π^{-1}) the channel decoder uses the redundancy to correct transmission errors.

The primary goal is to correct errors from the noisy channel. But implementation efficiency of communication systems in hardware mandates e.g. quantization of data values and the use of suboptimal algorithms, i.e., algorithms that generate results which deviate from the theoretically correct values. Both can be seen as further sources of errors in addition to the noise on channel. In the same way, errors induced by hardware faults can be considered as yet another error source in a communication system. The question is if the hardware errors can be processed in a similar way than the channel and what are the costs.

2.1 *Methodology: Error Mitigation Using Dynamic Resilience Actuators*

Modeling of hardware errors is crucial for the design of dependable systems. Radiation, thermal effects, aging, or process or parameter variations cause distortions on a physical level which can be modeled by probabilistic bit flips according to the resilience articulation point (RAP) model [16] (see also the chapter "RAP Model—Enabling Cross-Layer Analysis and Optimization for System-on-Chip Resilience" in this book). Depending on its location, a bit flip can have very different effects. An error in the controller, for example, usually leads to a system malfunction, whereas individual errors in the memories or the data flow are often inherently corrected by a wireless receiver ([12, 31]). Efficiency in terms of area and energy will be achieved by recovering only from those errors which have a significant impact on the system output and by choosing the layer on which the treatment of these error results in the least overhead.

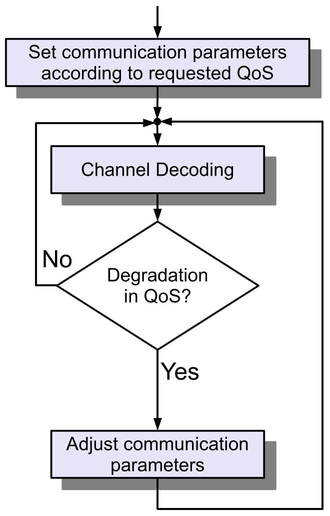
Dynamic approaches for error resilience also have to monitor the current hardware status. This monitoring can be done on different abstraction layers. Examples are error detection sequential (EDS) circuits on microarchitectural layer. EDS circuits are very popular [6]; however, they require pre- and post-silicon calibration. Monitors on higher abstraction layers are application-specific and normally more efficient. For example, [3] proposed to detect timing errors with a small additional hardware block which mimics the critical path under relaxed timing constraints. The result of the mimic hardware is compared to the normally operating unit. Deviations indicate timing errors. For a turbo and convolutional code decoder, the mimic hardware only required 0.7% of the decoder area. In this article we focus on resilience techniques which are employed after hardware errors have been detected, not on the detection methods themselves.

Many state-of-the art publications utilize low-level static resilience techniques to combat the effects of unreliable hardware, e.g., ECC protection of memories, Razor flip flops, or stochastic logic [36]. Static methods have the disadvantage of permanently decreasing the system performance in at least one of the terms of throughput, area, or power, even when no errors occur. In [31] for example, the static protection of a complete LDPC (Low-Density Parity Check) decoder for WiMax/WiFi resulted in an area overhead of 21%.

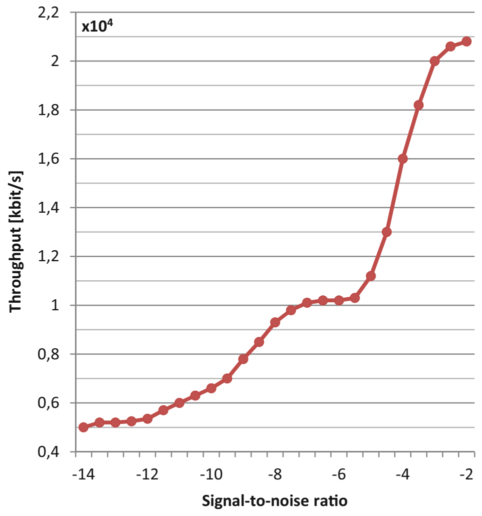
Dynamic techniques often use available hardware resources or have very low additional costs as we will show in Sect. 2.2.1. However, error detection circuits result in additional costs. When comparing static and dynamic methods, this additional cost has to be taken into account. In general, the choice of the protection method will also depend on the expected hardware error statistics as we will demonstrate in the next paragraph. Eventually, a combination of static and dynamic protection will likely result in the least overhead.

2.1.1 The Dynamic Behavior of Wireless Systems

Modern wireless communication standards, such as LTE (Long Term Evolution) or HSDPA (High Speed Downlink Packet Access) provide mechanisms to monitor and dynamically adapt to changes in the Quality-of-Service (QoS). The QoS in a wireless transmission system is typically defined as the bit or frame error rate with respect to a given signal-to-noise ratio. If the desired QoS cannot be achieved for the current transmission channel, communication parameters like code type, code rate, etc. are adjusted to improve the communications performance (see Fig. 3a). A good example for this dynamic behavior is the hybrid automatic repeat request (H-ARQ), which is used in LTE, HSDPA. These systems typically transmit blocks of data at


Set communication parameters according to requested QoS
Channel Decoding
Degradation in QoS?
No
Yes
Adjust communication parameters

(a)



(b)

Fig. 3 The standard communication flow dynamically adjusts communication parameters to achieve the required QoS, e.g., the code rate in Hybrid-ARQ systems. (**a**) Dynamic QoS flow of a modern wireless communication system. (**b**) In Hybrid-ARQ systems the code rate is dynamically adjusted for each block to ensure error-free transmission

a high data rate and with little error protection, i.e., with a very high code rate. If the decoder fails, the transmission of additional data is requested until the block is correctly decoded. Note that such a retransmission does not contain the same data as before. Instead, different information will be sent every time, which had been punctured on the transmitter side before. The additional information decreases the data rate but at the same time increases the probability that the block can be correctly decoded at the receiver. Figure 3b shows the throughput of a H-ARQ system over different SNR values. For high SNR values, decoding succeeds after the first transmission, i.e., the channel decoder can correct all errors, and a high throughput is obtained. With a decreasing SNR, more and more blocks require additional transmissions and the throughput is lowered. *The system dynamically adapts the code rate and the throughput for each block.*

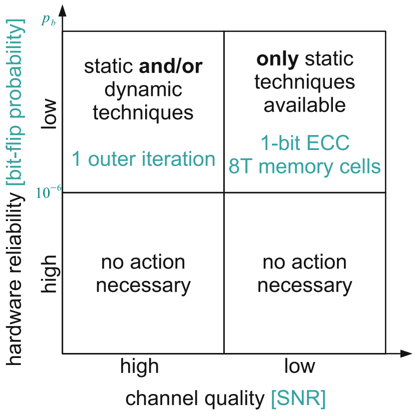
This example shows how wireless receivers adapt dynamically to changes in the transmission channel, i.e., varying SNR, and correct transmission errors. The question is how this idea can be applied to the case of hardware errors. It has been shown that low rates of hardware errors in a wireless receiver are not visible on the system level. This is due to the fact that for low SNR the channel errors dominate. For high SNR, when the channel error rate is very low, the channel decoder is able to correct the hardware errors. For moderate hardware error rates, some dynamic high-level techniques exist, e.g., increasing the number of decoder iterations to counterbalance the impact of hardware errors. However, for very high error rates on the hardware level, a purely software-based mitigation is not possible. An increase of reliability can generally be achieved by either static low-level techniques, like e.g., Razor flip flops, triple modular redundancy, or by dynamic high-level techniques, which exploit the flexibility of the receiver, e.g., increase of decoder iterations, or a combination of both. To their advantage, dynamic techniques are mainly algorithmic changes, which can be controlled by software and do not require a more costly change of the underlying hardware.

Consequently, it is possible to use high-level techniques to mitigate hardware errors in wireless communication systems. However, the channel quality changes over the time and channel noise and hardware noise may change independently from each other. In good channel conditions, we can use a part of the error correction capability of the receiver to combat hardware errors if needed. When the channel quality is very poor, all high-level techniques are needed to obtain the required QoS, and hardware errors have to be counterbalanced by static low-complexity methods. This is shown in Fig. 4a. When the hardware reliability is very high, no action has to be taken. High amounts of hardware errors cannot be overcome using dynamic techniques exclusively. A combination of dynamic and static techniques is mandatory. When the channel quality is very poor, only static techniques are available. For medium noise levels, there are potential trade-offs between dynamic and static techniques.

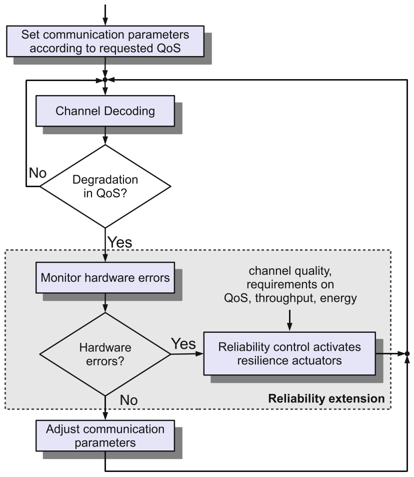
2.1.2 Concept of Dynamic Resilience Actuators

As mentioned before current standards, like HSDPA or LTE, adjust dynamically the QoS at runtime, e.g., higher data throughput rates are specified for higher SNR. This is due to the fact that the computational requirements on the different algorithms decrease with higher SNR in order to enable higher throughput. In future technologies the negotiated QoS may also depend on the reliability of the receiver hardware under given operating conditions. This leads to an entirely new paradigm—adaptive QoS with respect to communication reliability *and* hardware reliability. An illustration of this is the possibility to relax reliability requirements on the underlying hardware instead of providing a higher throughput at high SNR. For example, voltage overscaling can be applied, where the voltage is reduced beyond the point at which fault-free operation of the circuit is guaranteed in order to lower the power consumption of the receiver. In this way, QoS, hardware reliability, and implementation efficiency can be traded off against one another at runtime.

In [3], we presented how this new paradigm can be integrated into the existing QoS flow of wireless communication systems. Figure 4b shows the extended version of the original QoS flow from Fig. 3a. Low rates of hardware errors are implicitly corrected by a wireless receiver. In that case no further action is required. A higher rate of hardware errors results in a degradation of the QoS and, thus, can be detected by the standard QoS flow. The standard QoS flow is already error-resilient by itself, as it dynamically adjusts the communication parameters to obtain a certain QoS. In most cases, however, it will be cheaper in terms of energy to correct a



(a) Depending on the current hardware reliability and channel quality, different static and dynamic techniques (resilience actuators) are available to mitigate the impact of hardware errors. Entries in cyan color quantify the example in Section 2.2.



Set communication parameters according to requested QoS
Channel Decoding
Degradation in QoS?
No
Yes
Monitor hardware errors
channel quality, requirements on QoS, throughput, energy
Hardware errors?
Yes
Reliability control activates resilience actuators
No
Reliability extension
Adjust communication parameters

(b) Extended dynamic QoS flow: The reliability control unit chooses the resilience actuators which result in the least overhead and, thus, in an energy efficient design.

Fig. 4 Our new methodology integrates seamlessly into the existing QoS flow of today's communication systems. The available resilience techniques depend on the current channel quality and hardware reliability. (**a**) Depending on the current hardware reliability and channel quality, different static and dynamic techniques (resilience actuators) are available to mitigate the impact of hardware errors. Entries in cyan color quantify the example in Sect. 2.2. (**b**) Extended dynamic QoS flow: The reliability control unit chooses the resilience actuators which result in the least overhead and, thus, in an energy-efficient design

temporary hardware error by the activation of a dynamic protection mechanism than by changing the communication parameters as, e.g., a H-ARQ based correction is very costly with respect to energy consumption.

As already mentioned a degradation of the QoS can be caused by either channel errors or hardware errors. A differentiation of these two error sources is not possible with the existing QoS monitoring system only. Therefore, it is necessary to monitor the reliability status of each hardware component. Single bit flips in the data path for example are often mitigated by the algorithmic error resilience of the receiver. Application-specific detection circuits like the reduced-size ACS (add-compare-select)-unit for turbo decoding proposed in [3] can indicate the status of one component with only a small overhead.

We introduced a reliability control unit which activates one or several *resilience actuators* according to the current monitoring status. A resilience actuator is a dynamic protection mechanism, which can increase the error resilience either on component or on system level. Resilience actuators can be found on hardware level and on software level. So far, we identified four classes of actuators. On the lowest level, we can change the *hardware operating point*, e.g., the supply voltage or the clock frequency. The trade-off between supply voltage, clock frequency, and power consumption is well studied in the literature. Another possibility is the use of *low-level hardware techniques*, such as the selective protection of critical parts, or setting erroneous likelihood values to zero [31]. Many algorithms have parameters which can be changed at runtime. Advanced channel decoders operate iteratively. The number of iterations is a parameter which can easily be changed for each individual block by the software. For many components, we have a choice of different algorithms, starting from optimal algorithms with a high complexity down to suboptimal algorithms with a very low complexity, which offers a trade-off between QoS and implementation efficiency. The *choice of parameters and algorithms* is another class of actuators [3]. There also exist resilience actuators on *system level*. Adjusting the communication parameters, e.g., by choosing a channel code with a better error correction capability, improves the error resilience, but the effects are not immediate. A faster solution is to shift complexity between different components, when one of the components has a low hardware reliability. It is important to note that resilience actuators are only activated when hardware errors cause a degradation of the QoS.

In general, different actuators or combinations of actuators are suited to deal with different types of hardware errors. Normally, it is preferable to use actuators which do not require changes inside the components or which can be implemented with low complexity. Each actuator offers a different trade-off between hardware reliability, QoS, and implementation performance (throughput, energy). Based on the channel quality and the respective requirements on QoS, throughput, and energy, the reliability control chooses those actuators, which will best fulfill the requirements. Therefore, it is mandatory to characterize each actuator with regard to its influence on communications performance, throughput, area, and energy overhead. Sometimes, the reliability requirements necessitate the use of resilience actuators which have a severe effect, e.g., on the system throughput. In these cases, the reliability control also needs actuators which trade-off throughput and communications performance. The big advantage of this reliability extension is the dynamic protection of the wireless receiver, which is only activated when necessary.

2.2 A Case Study

In the last section, our new methodology was generally introduced. The trade-off between channel quality and hardware resilience and the choice of the resilience actuators are application-specific and cannot be quantified in a general fashion. In this paragraph, we demonstrate our methodology on a concrete example in order to make it more seizable.

Multiple-antenna or MIMO systems have the potential to increase the data rate of wireless communication systems. They belong to the most advanced systems in 4G and 5G communication standards, and their very high complexity is a challenge for

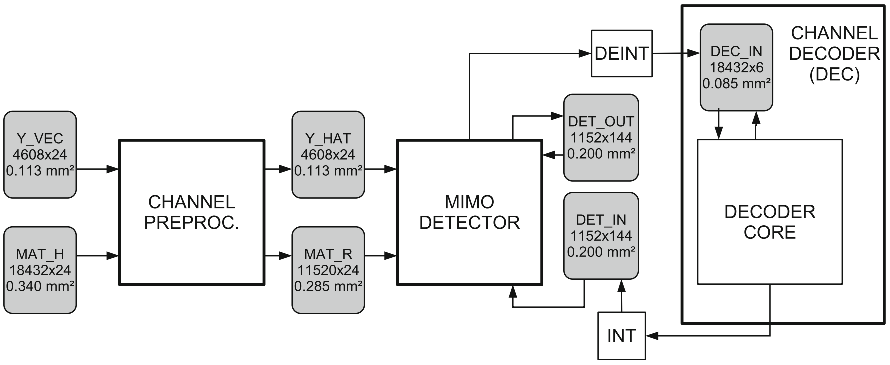


Fig. 5 Generic architecture of an iterative MIMO-BICM receiver including main building blocks and system memories

any hardware implementation. To demonstrate our novel methodology, we chose to apply it to a double-iterative MIMO-BICM (bit-interleaved coded modulation) transmission system.

Aforementioned, a channel code provides redundancy, which allows the correction of transmission errors in the receiver. An interleaver between channel encoder and modulator reduces dependencies between neighboring bits. The modulated symbols are multiplexed to an array of antennas and then transmitted in parallel to increase the data rate. Such a system setup is called MIMO-BICM system. On the receiver side, a MIMO detector decouples the multiple transmission streams, and the channel decoder corrects errors, which have been induced by noise on the communication channel. The most advanced receiver techniques combine the MIMO detector and the channel decoder in an iterative feedback loop to further improve the communications performance of the receiver [17]. These two blocks exchange likelihood values, which reflect their confidence in the results of their computations. The channel decoder can be iterative itself (and often is), which results in a double-iterative receiver structure. The number of iterations is dynamic and depends strongly on the respective system state and QoS requirements.

Multiple-antenna systems are combined with different types of channel codes in the existing standards. WiFi features LDPC codes and convolutional codes, whereas LTE supports only the trellis based convolutional and turbo codes. WiMax supports all three kinds of channel codes. Therefore, we mapped the iterative receiver structure from [11] onto a general architecture framework, which allows us to plug in different MIMO detectors and channel decoders [13]. The generic architecture shown in Fig. 5 connects the main building blocks via several system memories.

We presented in [11] the details of the implementation results for all components of the iterative receiver [13, 34]. All designs were synthesized in a 65 nm low-power bulk CMOS standard cell library. Target frequency after place and route is 300 MHz, which is typical of industrial designs (exception WiMax/WiFi LDPC decoder). The size of the system memories is determined by the largest block length in each communication standard. For example, LTE turbo codes include up to 18,432 bits. In this case, the system memories require approximately 40% of the total system area. For WiMax/WiFi, the maximum block length is only 2304 bits which results in a much smaller area for the system memories. The power consumption of the memories is not neglectable when compared to the other components [13]. The total power consumption depends heavily on the number of inner and outer iterations.

The system memories add substantially to the die area of such an iterative MIMO-BICM receiver. Memories are very susceptible to hardware errors due to their dense and highly optimized layouts. In [12], we analyzed the impact of hardware errors in the different system memories on the system performance of a MIMO-BICM system. We found out that especially the memories containing complex-valued data, i.e. the channel information and the received vectors, are very sensitive. Figure 6 shows the degradation of the communications performance when errors are injected in the channel information memory. Up to a bit error probability of $p_b = 10^{-6}$ the degradation is negligible for the typical frame error rates (FERs) of a wireless system. Afterwards, the performance decreases gradually with an increasing p_b.

We assume that the memory errors result from supply voltage drops which occur regularly during power state switching. In this context, several resilience actuators exist, which can be applied to different degrees of hardware unreliability in order to mitigate the impact of the hardware errors on the system performance [26]. Table 1 lists them with their influence on area, power consumption, and throughput

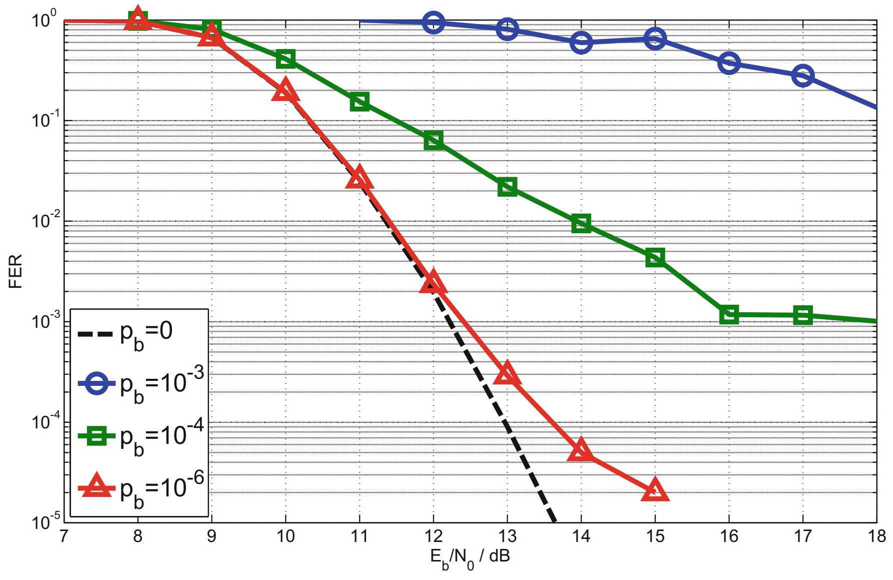


Fig. 6 The system communication performance is gradually decreasing for random bit flips in the channel information memory. p_b depicts the bit error probability

Table 1 Quantitative comparison of resilience actuators for the system memories of an iterative MIMO-BICM receiver

Resilience actuator	Impacts	Tolerated hardware (un-) reliability	Tolerated hardware error probability
Algorithmic error resilience	Area +0%	−200 mV supply voltage	10^{-6}
	Power +0%		
	Throughput −0%		
1 outer iteration	Area +0%	−300 mV supply voltage	$4 \cdot 10^{-5}$
	Power +0%		
	Throughput −75%		
1-bit error correction code	Area +30%	−400 mV supply voltage	$8 \cdot 10^{-4}$
	Power +30%		
	Throughput −0%		
8T memory cells	Area +25%	−500 mV supply voltage	$8 \cdot 10^{-3}$ for the equivalent 6T memory cells
	Power +0%		
	Throughput −0%		

and their error resilience. In Fig. 4a, these actuators are arranged according to our methodology (cyan text). No action has to be taken as long as there is a high hardware reliability, i.e. voltage drops of no more than 200 mV. Within this region, the receiver shows an inherent algorithmic error resilience. For a decreased reliability in which voltage drops up to 300 mV occur, we can react on the highest level by increasing the number of iterations in order to regain communications performance. For transient errors, this leads only to a temporary throughput degradation without loss of communications performance. When errors occur with a high probability $p_b > 5 \cdot 10^{-5}$, high-level resilience actuators cannot provide the necessary resilience. On a lower level, the contents of the memory can be protected by a simple 1-bit error correction code. The resilience can be even further increased on technology level by employing 8-transistor (8T) memory cells instead of 6-transistor (6T) cells resulting in a smaller implementation overhead. 8T memory cells can even tolerate voltage drops of 500 mV. However, the increase in area and power is in both cases permanent.

2.2.1 Resilience Actuators

Error resilience techniques for channel decoding have already been thoroughly investigated (cf. [11]). However, there are potential trade-offs on system level between MIMO detection and channel decoding, which we will discuss in the following. Except for the hardware operating point, we will restrict ourselves to application-specific resilience actuators. Universal, already established, low-level hardware techniques can be applied to any application and result in a constant overhead. Here, we focus on dynamic resilience techniques, which can be switched on and off as necessary and which do not have a large impact on implementation complexity and energy consumption.

As MIMO detectors have no inherent error correction capability, algorithmic changes inside the detector component cannot improve the error resilience. This has to be done either on system level or by changing the hardware operating point. These actions usually have a negative influence on system throughput and/or communications performance. Therefore, we also introduce algorithmic resilience actuators, enabling a trade-off of throughput and communications performance in order to counterbalance these effects.

- *Hardware operating point:* When timing errors occur, the clock frequency can be reduced or the supply voltage can be increased to make the circuit faster. However, both approaches require additional control circuits and energy. The trade-off between supply voltage and energy is well-understood. The number of bit flips in a memory, for example, strongly depends on the voltage: Increasing the supply voltage decreases the soft error rate. According to [8], the soft error rate drops by about 30% when the operating voltage is increased by 100 mV compared to the nominal voltage. Changing the hardware operating point offers a trade-off between reliability and energy consumption, which is often used for voltage overscaling.
- *Adjustment of detection quality:* Changing the detection quality offers a trade-off between communications performance and throughput but has no direct influence on the error resilience. However, a higher throughput augments the available time budget and, thus, offers a higher potential for error resilience. This resilience actuator uses the available algorithmic flexibility and thus has only a negligible influence on power and area consumption.
- *External LLR (log-likelihood ratio) manipulations:* Instead of accessing the MIMO detector directly, we propose low-complexity techniques, which work only on the LLR-input and -output values of the detector. LLR values have a high robustness against hardware errors. If an LLR value is equal to zero, it contains no information. Thus, the most important information is stored in the sign bit. As long as this sign bit is not compromised, the core information is still correct and the channel decoder can correct the hardware errors. For more details see additionally [11].

Instead of increasing the reliability of components individually, the problem can also be tackled on system level. The double-iterative structure of a MIMO-BICM receiver offers several high-level possibilities to combat the unreliability of its components. We present the most promising techniques in the remainder of this section.

- *Iteration control mechanisms:* An iteration control typically monitors exchanged values in an iterative system and checks stopping conditions to detect the convergence of the processed block. In [12] we analyzed the impact of memory errors on the system behavior of an iterative MIMO system. We observed that errors in any of the memories before the MIMO detector have an increased

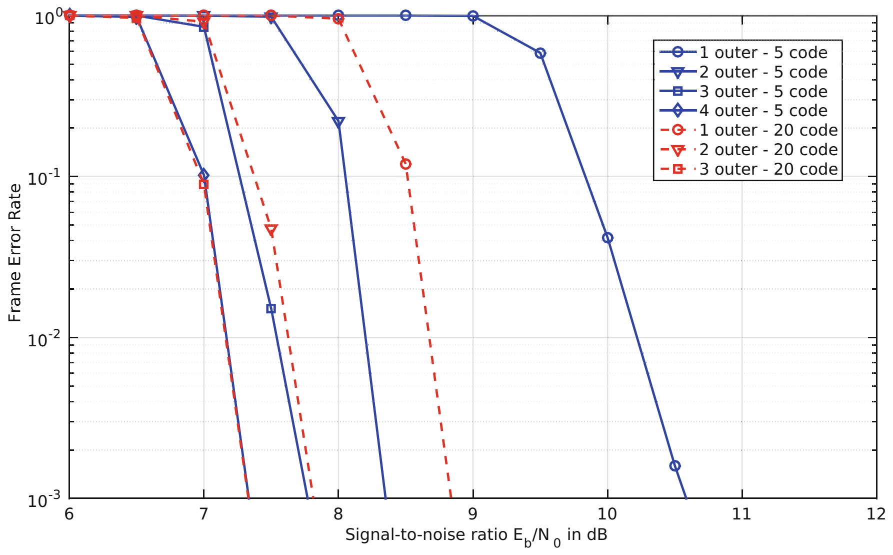


Fig. 7 Example for complexity shifting in a double-iterative MIMO-BICM receiver by varying the number of outer loop and channel decoder iterations

impact on the communications performance if the incorrect values are processed repeatedly during the outer iterations. In [10], it was possible to reduce the number of outer iterations to an average below 2 (from a maximum of 10) without sacrificing communications performance. A further throughput increase is possible by allowing a degradation of communications performance. The additional effort for an iteration control is very low compared to channel decoding [11].

- *Complexity shifting between components:* An example for a global algorithmic adaption is to shift the complexity between system components: When a building block cannot compensate an error locally, the system convergence can still be achieved by increasing the computational effort of other building blocks. Such a shift can be achieved, for instance, between the channel decoder and the MIMO detector, leveraging the outer feedback loop. When the MIMO detector is not able to counterbalance the impact of hardware errors, the number of channel decoder iterations and/or the number of outer loop iterations can be increased in order to maintain the communications performance. Figure 7 shows the frame error rate for a 4×4 antennas, 16-QAM system employing a WiMax-like LDPC code where complexity shifting can be used. We compare the frame error rate for different numbers of decoder iterations and outer iterations. Let us consider the case when the receiver is performing 3 outer iterations and 5 LDPC iterations. When the MIMO detector suffers from hardware errors, e.g. due to a temperature increase, we can temporarily shift more processing to the LDPC decoder by performing only 2 outer iterations and 20 LDPC iterations. The new configuration provides

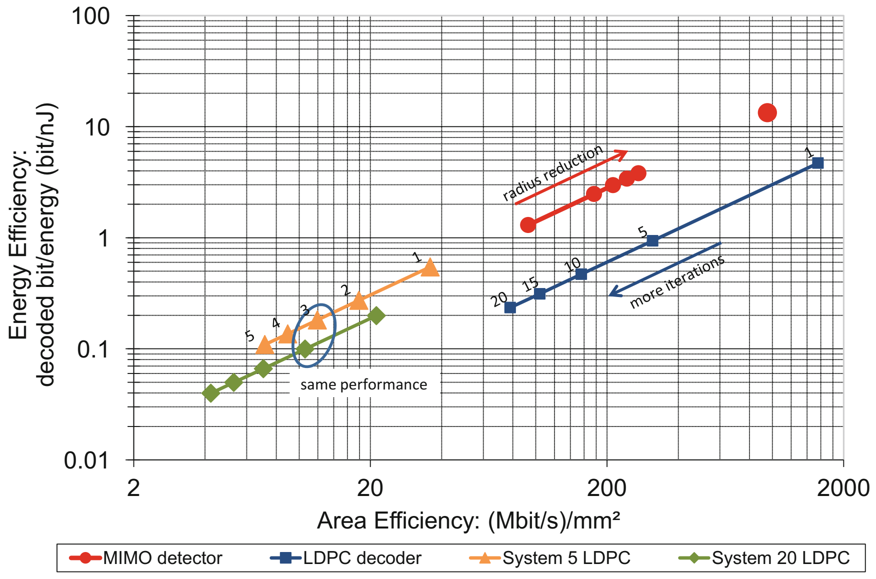


Fig. 8 Implementation efficiency of MIMO detector, WiMax/WiFi LDPC decoder, and two system configurations using the efficiency metrics from [25]

the same communications performance. The question is how such a shift changes the energy efficiency of the MIMO receiver. Figure 8 shows the implementation efficiency of MIMO-BICM receiver and its components. The red curve shows the efficiency of the MIMO detector for different search radii and in a MMSE-SIC configuration (single red point). The blue curve shows the efficiency of an LDPC decoder running with different number of iterations. The LDPC decoder is a flexible decoder which supports all code rates and code lengths from WiMax and WiFi standard. The yellow and the green curve show the system efficiency for different numbers of outer iterations with 5 LDPC iterations (yellow) and 20 LDPC iterations (green), respectively. With the help of this graph, we can quantify the influence of a complexity shift: when changing from 3 outer and 5 LDPC iterations to 2 outer and 20 LDPC iterations, the energy efficiency of the system is reduced by approximately 50% (blue circle). However, the same communications performance is achieved and when the temperature in the MIMO detector decreases, the reliability control can return to the original configuration.

- *Shifting of error correction capability between components:* Typically, MIMO detector and channel decoder are designed and implemented independently of each other. The MIMO transmission scheme provides a large data rate but has no error correction abilities. The error correction capability is solely provided by the channel code to improve the error rate performance of the transmission system. From a system point of view, the MIMO detector does not work on completely independent data as there are dependencies from the overlaying channel code. However, this diversity cannot be exploited by the detector as the channel interleaver hides the code structure from the detector and because most channel code constraints span over many MIMO detection vectors. Kienle [24] introduced a small block code in each MIMO detection vector in order to generate a small diversity gain in the MIMO detector while simplifying the outer channel code to keep the overall coding rate constant. While this approach targeted the decoding complexity, it can also be used to increase the error resilience of the MIMO detector. Each parity check in one MIMO vector improves the error correction capabilities of the detector. On a system level, the diversity gain can be split between detector and decoder dynamically, thus, allowing the system to react dynamically to changing hardware error rates. The only drawback of this approach is that the diversity separation has to be done on the transmitter side, which causes a higher latency.

3 Approximate DRAM

Some communication systems require large data block sizes that cannot be stored in on-chip memories (SRAMs) anymore. In this case data has to be stored externally in DRAMs. Thus, in the following we shift our focus on DRAMs.

Approximate DRAM is a new concept that adapts the idea of approximate computing to DRAMs [23, 30]. Approximate DRAM exploits this fact by lowering the refresh frequency (reducing vendor guardbands) or even disable the refresh completely and accepting the risk of data errors. The underlying motivation for an Approximate DRAM is the increasing power consumption and performance penalty caused by unavoidable DRAM refresh commands. The authors of [29] and [1] predicted that 40–50% of the power consumption of future DRAM devices will be caused by refresh commands. Moreover, 3D integrated DRAMs like Wide I/O or HMC worsen the scenario with respect to increased cell leakage, due to the much higher temperature. Therefore, the refresh frequency needs to be increased accordingly to avoid retention errors [37].

The characteristic refresh parameters of DRAMs, listed in datasheets, are very pessimistic due to the high process margins added by the vendors to ensure correct functionality under worst-case conditions and most important a high yield [28]. Thus, the DRAM refresh rate recommended by the vendors and JEDEC ($t_{\mathrm{REF}} = 64\,\mathrm{ms}$) adds a large guardband, as shown in Fig. 9.

As mentioned before many applications like wireless systems have an inherent error resilience that tolerates these errors and therefore, refresh power often can be reduced with a minimal loss of the output quality.

Figure 9 qualitatively shows the retention error behavior over time and the design space for Approximate DRAM. The sphere around the curve represents the process variation, Variable Retention Times (VRT), and Data Pattern Dependencies (DPD). In general, we have two key parameters for Approximate DRAM: The *data lifetime*

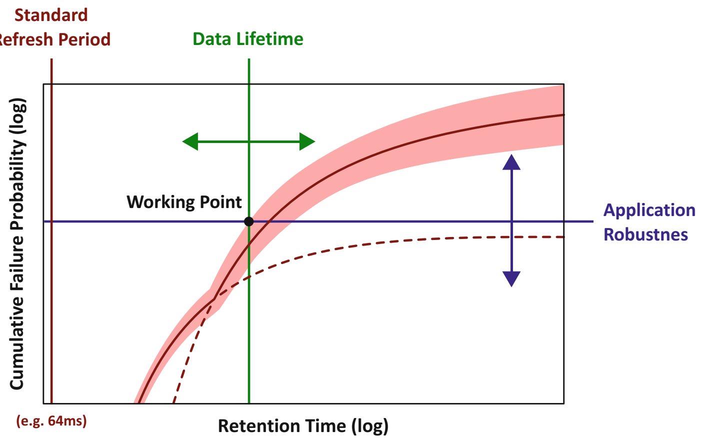


Fig. 9 Qualitative retention error behavior of DRAMs

and the *application robustness*. Both parameters lead to three possibilities in this design space:

- Refresh can be switched off if the data lifetime is smaller than the actual required refresh period.
- Refresh can be turned off if the data lifetime is larger than the required refresh period and the application provides resilience to the resulting number of errors at this working point.
- If the application only provides a maximal robustness the refresh rate is configured according to the resulting working point.

The reliability-energy trade-off for Approximate DRAMs can be explored only by using fast and accurate retention error-aware DRAM models.

In [39] we developed such a model that is usable in full system level simulations. The model was calibrated to the measurement results of DDR3 DRAM devices. A measurement statistic is shown in Fig. 10. Here we measured 40 identical 4 Gbit DDR3 chips from the same vendor. Each single device has been measured ten times at four different temperatures and five retention times, resulting in a total of 8000 measurement points. We plot the retention times versus the normalized and averaged number of errors obtained during each measurement step. The bars mark the minimum and the maximum measured number of errors. We find here a quite prominent variation in the order of 20% (max. number of errors), which shows a large temperature dependency. This needs to be considered as realistic guardband in approximate computing platforms utilizing the Approximate DRAM approach (cf. the sphere in Fig. 9). Additionally, the figure shows a histogram of the absolute number of bit errors (between $1 \cdot 10^6$ and $4 \cdot 10^6$) measured at the data point with 100s retention time and a temperature of 25 °C.

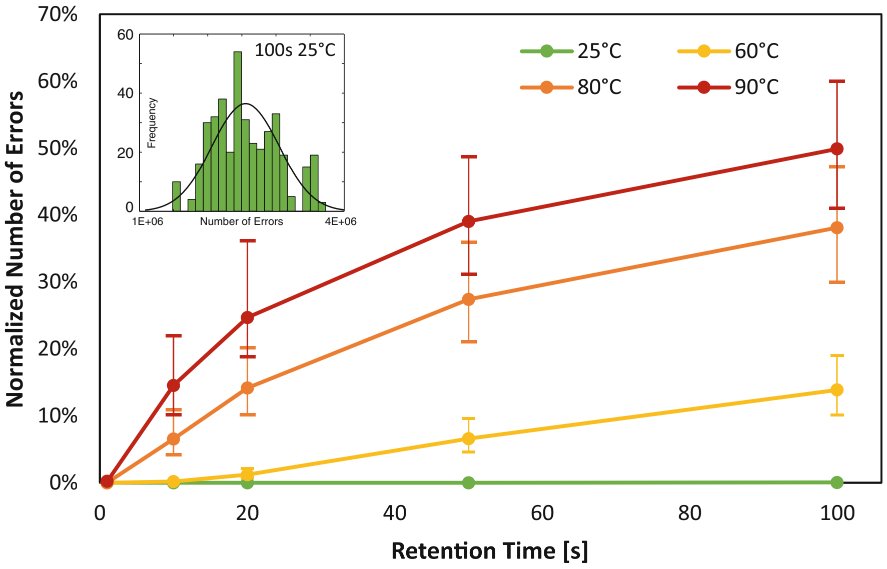


Fig. 10 Retention error measurements of 40 4 Gbit DRAM Devices with different temperatures

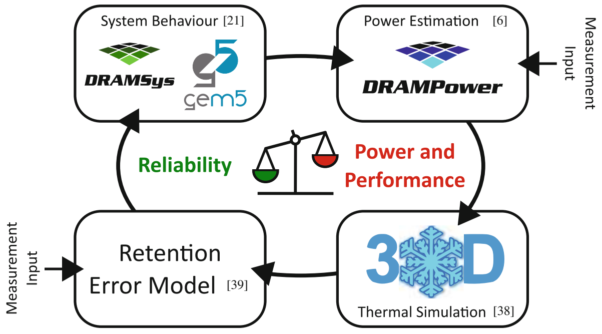


Fig. 11 Simulation Framework for Approximate DRAM Explorations and Reliability Management in SoCs

Figure 11 shows our closed-loop simulation flow for investigations on Approximate DRAM. It is based on SystemC Transaction Level Models (TLM) for fast and accurate simulation. This simulation loop uses the modular DRAMSys framework [21] and consists of four key components: DRAM and core models [21], a DRAM power model [5], thermal models [38], and the aforementioned DRAM retention error model. The remaining models are shortly introduced in the following:

- **DRAM and Core Models:** The DRAM model of the framework is based on a DRAM specific TLM protocol called *DRAM-AT* [20]. Due to TLM's modular fashion several types of DRAM and controller configurations can be modeled. For modeling the cores the *gem5* simulator is used [2]. We developed a coupling

between gem5 and SystemC to be able to integrate this powerful research platform in our simulation loop [19].
- **DRAM Power Model:** Since DRAMs contribute significantly to the power consumption of today's systems [9, 27], there is a need for accurate power modeling. For our framework we use DRAMPower [4, 5], which uses either parameters from datasheets, estimated via DRAMSpec [33] or measurements to model DRAM power.
- **Thermal Model:** 3D packaging of systems like Wide I/O DRAM starts to break down the memory and bandwidth walls. However, this comes at the price of increased power density and less horizontal heat removal capability of the thinned dies. Therefore, we integrated the thermal simulator 3D-ICE [38] in a SystemC wrapper [18] that is included in our closed-loop simulation for Approximate DRAM analysis.

In a detailed case study [22] we used the presented simulation framework (Fig. 11) to investigate the influence of Approximate DRAM on three different applications. We achieved in average a more than 10% decrease of the total energy consumption.

4 Conclusions

Technology scaling is leading to a point where traditional worst-case design is no longer feasible. In this chapter, we presented a new methodology for the design of dependable wireless systems. We combined cross-layer reliability techniques to treat hardware errors with the least possible overhead leading to a high energy efficiency. This methodology enables efficient trade-offs between communications performance, throughput, and energy efficiency. However, the exact trade-off depends on the real application requirements, which was not in the focus of this work. Application-specific resilience actuators together with low-level techniques offer the ability to respond to the changing requirements on reliability and quality-of-service. We illustrated our new methodology on a state-of-the-art generic double-iterative MIMO-BICM receiver which belongs to the most complex systems in modern communication standards.

We identified dynamic resilience actuators on all layers of abstraction. Each actuator offers a trade-off between communications performance, implementation performance (throughput, power), and error resilience. Any actuator which trades off communications performance for throughput, e.g., the sphere radius, can be reused to increase the error resilience, when combined with a reduction of the clock frequency. Throughput and error resilience are, thus, closely related. As we have shown, algorithmic resilience actuators offer a great potential for dynamic trade-offs between communications performance, implementation performance, and error resilience. This work emphasizes the strong mutual dependencies between these three design metrics in a wireless receiver.

When the requirement in communication systems on data block sizes exceeds the capacities of the on-chip memories (SRAMs), external memories, such as DRAMs, have to be used. To reduce their impact on energy and performance we exploited the concept of Approximate DRAM. However, this comes at the cost of reduced reliability. For the exploration of approximate DRAMs we introduced a holistic simulation framework that includes an advanced DRAM retention error model. This model is calibrated to real measurements of recent DRAM devices. Finally, we demonstrated using the holistic simulation platform that the impact of Approximate DRAM on the quality (QoS or QoR) is negligible while saving refresh energy for three selected applications.

References

1. Bhati, I., Chishti, Z., Lu, S.L., Jacob, B.: Flexible auto-refresh: Enabling scalable and energy-efficient DRAM refresh reductions. In: Proceedings of the 42nd Annual International Symposium on Computer Architecture, pp. 235–246. ACM, New York (2015)
2. Binkert, N., Beckmann, B., Black, G., Reinhardt, S.K., Saidi, A., Basu, A., Hestness, J., Hower, D.R., Krishna, T., Sardashti, S., Sen, R., Sewell, K., Shoaib, M., Vaish, N., Hill, M.D., Wood, D.A.: The gem5 simulator. SIGARCH Comput. Archit. News **39**(2), 1–7 (2011). http://doi.acm.org/10.1145/2024716.2024718
3. Brehm, C., May, M., Gimmler, C., Wehn, N.: A case study on error resilient architectures for wireless communication. In: Proceedings of the Architecture of Computing Ssystems, pp. 13–24 (2012)
4. Chandrasekar, K., Akesson, B., Goossens, K.: Improved power modeling of DDR SDRAMs. In: 2011 14th Euromicro Conference on Digital System Design (2011). http://dx.doi.org/10.1109/DSD.2011.17
5. Chandrasekar, K., Weis, C., Li, Y., Akesson, B., Naji, O., Jung, M., Wehn, N., Goossens, K.: DRAMPower: Open-source DRAM power & energy estimation tool (2012). http://www.drampower.info
6. Das, S., Tokunaga, C., Pant, S., Ma, W.H., Kalaiselvan, S., Lai, K., Bull, D.M., Blaauw, D.T.: RazorII: in situ error detection and correction for PVT and SER tolerance. IEEE J. Solid State Circuits **44**(1), 32–48 (2009). https://doi.org/10.1109/JSSC.2008.2007145
7. Designing Chips without Guarantees. IEEE Design & Test of Computers **27**(5), 60–67 (2010). https://doi.org/10.1109/MDT.2010.105
8. Dixit, A., Wood, A.: The impact of new technology on soft error rates. In: Proceedings of the IEEE International Reliability Physics Symposium (IRPS) (2011). https://doi.org/10.1109/IRPS.2011.5784522
9. Farahini, N., Hemani, A., Lansner, A., Clermidy, F., Svensson, C.: A scalable custom simulation machine for the Bayesian confidence propagation neural network model of the brain. In: 2014 19th Asia and South Pacific Design Automation Conference (ASP-DAC), pp. 578–585 (2014). https://doi.org/10.1109/IRPS.2011.578452210.1109/ASPDAC.2014.6742953
10. Gimmler, C., Lehnigk-Emden, T., Wehn, N.: Low-complexity iteration control for MIMO-BICM systems. In: Proceedings of the IEEE 21th International Symposium on Personal, Indoor and Mobile Radio Communications PIMRC 2010. Istanbul (2010)
11. Gimmler-Dumont, C., Wehn, N.: A cross-layer reliability design methodology for efficient, dependable wireless receivers. ACM Trans. Embed. Comput. Syst. **13**, 1–29 (2014)
12. Gimmler-Dumont, C., Brehm, C., Wehn, N.: Reliability study on system memories of an iterative MIMO-BICM system. In: Proceedings of the IFIP/IEEE International Conference on Very Large Scale Integration 2012 (2012)

13. Gimmler-Dumont, C., Kienle, F., Wu, B., Masera, G.: A system view on iterative MIMO detection: dynamic sphere detection versus fixed effort list detection. VLSI Design J. **2012**, Article ID 826350 (2012). https://doi.org/10.1155/2012/826350
14. Gupta, P., Agarwal, Y., Dolecek, L., Dutt, N., Gupta, R.K., Kumar, R., Mitra, S., Nicolau, A., Rosing, T.S., Srivastava, M.B., Swanson, S., Sylvester, D.: Underdesigned and opportunistic computing in presence of hardware variability. IEEE Trans. Comput. Aided Design Integr. Circuits Syst. **32**(1), 8–23 (2013). https://doi.org/10.1109/TCAD.2012.2223467
15. Henkel, J., Bauer, L., Becker, J., Bringmann, O., Brinkschulte, U., Chakraborty, S., Engel, M., Ernst, R., Hartig, H., Hedrich, L., et al.: Design and architectures for dependable embedded systems. In: 2011 Proceedings of the 9th International Conference on Hardware/Software Codesign and System Synthesis (CODES+ ISSS), pp. 69–78. IEEE, Piscataway (2011)
16. Herkersdorf, A., Aliee, H., Engel, M., Glaß, M., Gimmler-Dumont, C., Henkel, J., Kleeberger, V.B., Kochte, M.A., Kühn, J.M., Mueller-Gritschneder, D., Wehn, N., et al.: Resilience articulation point (RAP): cross-layer dependability modeling for nanometer system-on-chip resilience. Microelectr. Reliab. **54**(6), 1066–1074 (2014)
17. Hochwald, B., ten Brink, S.: Achieving near-capacity on a multiple-antenna channel. IEEE Trans. Commun. **51**(3), 389–399 (2003). https://doi.org/10.1109/TCOMM.2003.809789
18. Jung, M.: Icewrapper - a systemC wrapper for 3D-ICE (2015). http://www.uni-kl.de/3d-dram/tools/icewrapper/
19. Jung, M., Wehn, N.: Coupling gem5 with systemC TLM 2.0 virtual platforms. In: gem5 User Workshop, International Symposium on Computer Architecture (ISCA). Portland (2015)
20. Jung, M., Weis, C., Wehn, N., Chandrasekar, K.: TLM modelling of 3D stacked wide I/O DRAM subsystems: a virtual platform for memory controller design space exploration. In: Proceedings of the 2013 Workshop on Rapid Simulation and Performance Evaluation: Methods and Tools, RAPIDO '13, pp. 5:1–5:6. ACM, New York (2013). http://doi.acm.org/10.1145/2432516.2432521
21. Jung, M., Weis, C., Wehn, N.: DRAMSys: A flexible DRAM subsystem design space exploration framework. IPSJ Trans. Syst. LSI Design Methodol. **8**, 63–74 (2015)
22. Jung, M., Zulian, E., Mathew, D., Herrmann, M., Brugger, C., Weis, C., Wehn, N.: Omitting refresh - A case study for commodity and wide I/O DRAMs. In: 1st International Symposium on Memory Systems (MEMSYS 2015). Washington (2015)
23. Jung, M., Mathew, D.M., Weis, C., Wehn, N.: Efficient reliability management in SoCs - An approximate DRAM perspective. In: 21st Asia and South Pacific Design Automation Conference (ASP-DAC) (2016)
24. Kienle, F.: Low-Density MIMO Codes. In: Proceedings of the 5th International Symposium on Turbo Codes and Related Topics, pp. 107–112. Lausanne (2008)
25. Kienle, F., Wehn, N., Meyr, H.: On complexity, energy- and implementation-efficiency of channel decoders. IEEE Trans. Commun. **59**(12), 3301–3310 (2011). https://doi.org/10.1109/TCOMM.2011.092011.100157
26. Kleeberger, V., Gimmler-Dumont, C., Weis, C., Herkersdorf, A., Mueller-Gritschneder, D., Nassif, S., Schlichtmann, U., Wehn, N.: A cross-layer technology-based study of the impact of memory errors on system resilience. IEEE Micro **33**(4), 46–55 (2013)
27. Krueger, J., Donofrio, D., Shalf, J., Mohiyuddin, M., Williams, S., Oliker, L., Pfreundt, F.J.: Hardware/software co-design for energy-efficient seismic modeling. In: Proceedings of the 2011 International Conference for High Performance Computing, Networking, Storage and Analysis (SC), pp. 1–12 (2011)
28. Lee, D., Kim, Y., Pekhimenko, G., Khan, S., Seshadri, V., Chang, K., Mutlu, O.: Adaptive-latency DRAM: Optimizing DRAM timing for the common-case. In: 2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA), pp. 489–501 (2015). https://doi.org/10.1109/HPCA.2015.7056057
29. Liu, J., Jaiyen, B., Veras, R., Mutlu, O.: RAIDR: Retention-aware intelligent DRAM refresh. In: Proceedings of the 39th Annual International Symposium on Computer Architecture, ISCA '12, pp. 1–12. IEEE Computer Society, Washington (2012). http://dl.acm.org/citation.cfm?id=2337159.2337161

30. Lucas, J., Alvarez-Mesa, M., Andersch, M., Juurlink, B.: Sparkk: Quality-scalable approximate storage in DRAM. In: The Memory Forum (2014). http://www.redaktion.tu-berlin.de/fileadmin/fg196/publication/sparkk2014.pdf
31. May, M., Alles, M., Wehn, N.: A case study in reliability-aware design: A resilient LDPC code decoder. In: Proceedings of the Design, Automation and Test in Europe DATE '08, pp. 456–461. Munich (2008)
32. Mitra, S., Brelsford, K., Kim, Y.M., Lee, H.H.K., Li, Y.: Robust system design to overcome CMOS reliability challenges. IEEE J. Emer. Sel. Topics Circuits Syst. **1**(1), 30–41 (2011). https://doi.org/10.1109/JETCAS.2011.2135630
33. Naji, O., Weis, C., Jung, M., Wehn, N., Hansson, A.: A high-level DRAM timing, power and area exploration tool. In: Embedded Computer Systems Architectures Modeling and Simulation (SAMOS) (2015)
34. Nazar, G.L., Gimmler, C., Wehn, N.: Implementation comparisons of the QR decomposition for MIMO detection. In: Proceedings of the 23rd Symposium on Integrated Circuits and System Design (SBCCI '10), pp. 210–214. ACM, New York (2010). https://doi.org/10.1145/1854153.1854204
35. Nowka, K., Nassif, S., Agarwal, K.: Characterization and design for variability and reliability. In: Proceedings of the IEEE Custom Integrated Circuits Conference CICC 2008, pp. 341–346 (2008). https://doi.org/10.1109/CICC.2008.4672092
36. Qian, W., Li, X., Riedel, M., Bazargan, K., Lilja, D.: An architecture for fault-tolerant computation with stochastic logic. IEEE Trans. Comput. **60**(1), 93–105 (2011). https://doi.org/10.1109/TC.2010.202
37. Sadri, M., Jung, M., Weis, C., Wehn, N., Benini, L.: Energy optimization in 3D MPSoCs with Wide-I/O DRAM using temperature variation aware bank-wise refresh. In: Design, Automation and Test in Europe Conference and Exhibition (DATE), 2014, pp. 1–4 (2014). https://doi.org/10.7873/DATE2014.294
38. Sridhar, A., Vincenzi, A., Ruggiero, M., Brunschwiler, T., Atienza, D.: 3D-ICE: Fast compact transient thermal modeling for 3D ICs with inter-tier liquid cooling. In: Proceedings of the IEEE International Conference on Computer-Aided Design ICCAD 2010 (2010)
39. Weis, C., Jung, M., Ehses, P., Santos, C., Vivet, P., Goossens, S., Koedam, M., Wehn, N.: Retention time measurements and modelling of bit error rates of WIDE I/O DRAM in MPSoCs. In: Proceedings of the IEEE Conference on Design, Automation & Test in Europe (DATE). European Design and Automation Association (2015)

Uncertainty-Aware Compositional System-Level Reliability Analysis

Hananeh Aliee, Michael Glaß, Faramarz Khosravi, and Jürgen Teich

Acronyms

BDD	binary decision diagram
CRA	compositional reliability analysis
CRN	compositional reliability node
CDF	cumulative distribution function
DSE	design space exploration
EA	evolutionary algorithm
ESL	electronic system level
MOEA	multi-objective evolutionary algorithm
MPSoC	multiprocessor system-on-chip
MTTF	mean-time-to-failure
RAL	reliability abstraction level
RTC	Real-Time Calculus
SER	soft error rate

H. Aliee
Helmholtz Zentrum München, Munich, Germany
e-mail: hananeh.aliee@helmholtz-muenchen.de

M. Glaß (✉)
Ulm University, Ulm, Germany
e-mail: michael.glass@uni-ulm.de

F. Khosravi · J. Teich
Friedrich-Alexander-Universität Erlangen-Nürnberg, Erlangen, Germany
e-mail: faramarz.khosravi@fau.de; juergen.teich@fau.de

J. Henkel, N. Dutt (eds.), *Dependable Embedded Systems*, Embedded Systems,
https://doi.org/10.1007/978-3-030-52017-5_19

1 Introduction

Continuous technology scaling necessitates to design today's embedded systems from electronic components with growing inherent unreliability. This unreliability arises from susceptibility to neutron-induced soft errors, negative-bias temperature instability, short-channel effect, gate leakage, etc. Therefore, it is vital to analyze system reliability at design time and employ appropriate reliability-improving techniques if necessary. A variety of reliability analysis techniques have been proposed for both the relatively low levels of abstraction that focus on technology as well as the system level that considers the interplay of hardware and software. But, there exists a gap between the levels where the faults originate, e. g., transistor level, and the system level for which the analysis is required. To close this gap and tame the ever increasing system complexity, *cross-level* analysis methodologies are required. These collect knowledge at lower levels by combining different analysis techniques and provide proper data for the analysis at higher levels of abstraction [24].

Evaluating the reliability of a system at design time, proper reliability-improving techniques can be explored and integrated into the system. However, these techniques typically come with higher monetary costs, latency, energy consumption, etc. This necessitates a multi-objective **DSE!** (**DSE!**) which maximizes reliability without deteriorating other design objectives. Usually, **DSE!** explores and evaluates millions of possible design alternatives (also called implementations) to find the Pareto-optimal ones. Herein, the efficiency of the reliability evaluation and exploration algorithm are the main challenging issues [2]. In [3], we propose an efficient and scalable reliability analysis technique based on Success Trees (STs) which is integrated into a **DSE!** framework to automatically evaluate an implementation's reliability. Most existing analysis techniques quantify a system's reliability without giving any hint on what to change to improve it, such that exploration algorithms basically perform random changes, e. g., through genetic operators in case of **EA!**s (**EA!**s). In [4, 6, 7, 11], we propose to employ the notion of *component importance* to rank components based on their contribution to the system reliability. Later, to improve the reliability of a system with limited budgets, we only need to improve the reliability of highly important components. In [5, 28], we show this guides the **DSE!** towards highly reliable, yet affordable implementations. So far, most existing analysis approaches assume that the reliabilities of components—or their lower bound—are more or less known precisely. Due to shrinking cell geometries, semiconductor devices encounter higher susceptibility to environmental changes and manufacturing tolerances such that a component's reliability has to be considered *uncertain*. An overview of the most important types of uncertainties for system design is given in Fig. 1.

Effects of unreliability and the associated uncertainty of components can propagate to the system level and become a challenge for system-level design methodologies. Even worse, destructive effects such as extreme temperature can affect several components simultaneously, resulting in correlated uncertainties. Neglecting such correlations can impose an intolerable inaccuracy to reliability analysis.

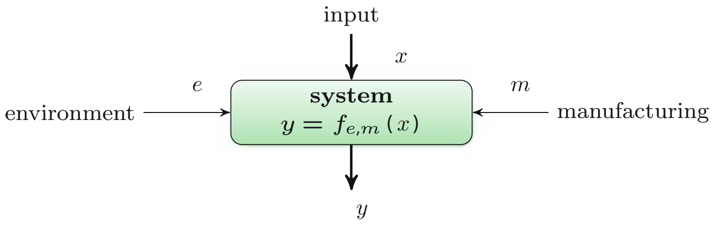


Fig. 1 Uncertainties that influence a system's response and its characteristics: Uncertain environmental influences $\triangle e$ like cosmic rays may cause soft errors. Manufacturing tolerances $\triangle m$ may lead to changing system behavior and permanent defects. Finally, uncertainty may also be present in case the inputs to a system may vary ($\triangle x$) or might not be known at design time

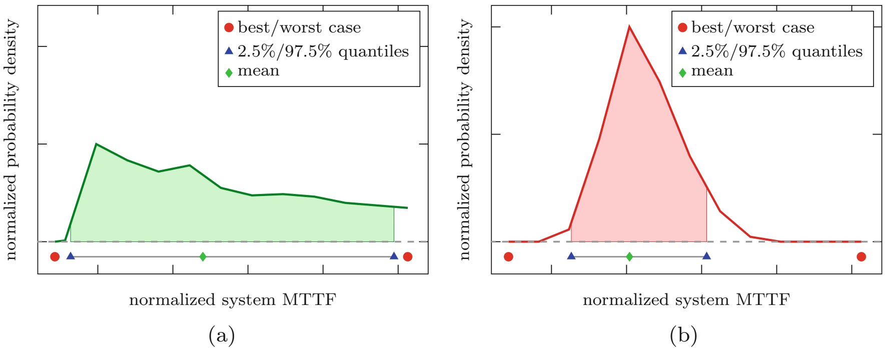


Fig. 2 Impact of uncertainty correlation among the reliability functions of different components on the uncertainty of the system **MTTF!** for an implementation candidate of an H.264 encoder/decoder. (**a**) Correlated component uncertainty. (**b**) Non-correlated component uncertainty

As an example, Fig. 2 depicts the distribution of system **MTTF!** (**MTTF!**) for an H.264 encoder/decoder implementation with and without the consideration of uncertainty correlations. While considering these correlations shows a good match between the simulated cases and the bounds, neglecting them may result in huge deviations from those bounds. This motivates the consideration of uncertainties and especially their correlations in cross-level reliability analysis. In this realm, this chapter introduces a methodology for **CRA!** (**CRA!**) that combines various reliability analysis techniques across different levels of abstraction while being aware of existing uncertainties and their correlations.

Considering uncertainty, system reliability is no longer a single value, but instead represented by a set of samples, upper and lower bound curves, or distribution functions which requires that a **DSE!** can consider implementations with uncertain objectives. Therefore, this chapter focuses on (a) the explicit modeling of uncertainties and their correlations in reliability analysis and (b) the integration of such an analysis into a framework for system-level **DSE!**. The techniques proposed are not tailored to a specific abstraction layer, but can be best classified as combining

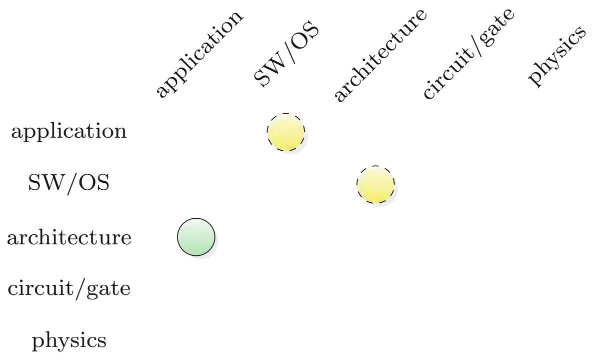


Fig. 3 Main abstraction layers of embedded systems and this chapter's major (green, solid) and minor (yellow, dashed) cross-layer contributions

architecture and application layers according to the embedded system abstraction layers as depicted in Fig. 3.

The rest of this chapter is organized as follows: Sect. 2 reviews related work and introduces required fundamentals. Section 3 introduces a formal **CRA!** framework and its application using a case study. An explicit modeling of uncertainty in reliability analysis and optimization is given in Sect. 4. Finally, Sect. 5 concludes the chapter.

2 Related Work and Fundamentals

2.1 Reliability Analysis and Optimization

Reliability analysis and optimization are thoroughly studied research topics that are of great importance fornearly every safety-critical system [12], especially embedded systems [36]. However, one can observe that the different areas raise significantly diverse needs for the applied analysis techniques. An overview of well-known reliability analysis techniques can be found in [35].

Up to now, several approaches have been presented for analyzing the reliability of embedded systems at system level which are typically integrated into system-level **DSE!**. In [16], fault-tolerant schedules are synthesized using task re-execution, rollback recovery, and active replication. The authors of [47] try to maximize reliability by selectively introducing redundancy while treating area consumption and latency as constraints. Reliability is introduced as an objective into system-level design in [13]. However, the employed reliability analysis techniques are restricted to series-parallel system structures which render them infeasible for typical embedded systems where processing and communication resources have to be shared. On the other hand, reliability analysis at low levels of abstraction has been studied thoroughly, e. g., transistor level [42] or for prospective switching devices like carbon nanotubes [32].

So far, few systematic approaches have been proposed to upscale the knowledge gathered at low abstraction levels to the system level. The work in [1] proposes a system-level dynamic temperature management scheme to prevent thermal hot spots through a run-time application re-mapping, and to efficiently mitigate aging effects in a many-core architecture. In [23], thermal effects in a Multiprocessor System-on-Chip (MPSoC) on its reliability are propagated into a scheduling and binding optimization at system level. Their analysis is based on a simulation of the MPSoC and given relations between the temperature profile and the resulting reliability. Similar reliability analysis techniques are used in [34] in order to optimize the lifetime of MPSoCs using the so-called *slack allocation*. However, these techniques are able to capture thermal effects only, without investigating the possibility to include and propagate these effects into a more holistic analysis that also takes into account, e. g., soft errors or a complex system structure like a networked embedded system consisting of several interconnected processors or MPSoCs.

2.2 Compositional Approaches to Reliability Analysis

A first attempt to close the gap on accurate power models for reliability analysis between the **ESL!** (**ESL!**) and the gate level is presented in [40]. While the approach sounds promising in modeling thermal effects on the component reliability, it fails to offer a formal framework that allows to integrate different analysis techniques cross level. Herkersdorf et al. [24] [RAP-Chap.] propose a framework for probabilistic fault abstraction and error propagation and show that all physically induced faults manifest in higher abstraction levels as a single or multiple bit flip(s). Similarly, the proposed **CRA!** model aims to propagate the effects of uncertainty and the resulting faults originating from lower levels of abstraction into the system-level analysis by incorporating appropriate reliability analysis techniques for each relevant *error model* at a specific level of abstraction. As a result, the developed concepts become independent of an actual error model since it abstracts from the actual source of unreliability during *upscaling*, i. e., the propagation of data from lower levels to higher levels by means of abstraction and data conversion. **CRA!** approaches that consider component-based software are presented in [37]. Although these approaches try to develop a more general compositional analysis scheme, they miss a well-defined mathematical underpinning and do not focus on automatic analysis as needed during **DSE!**.

The use of composition and decomposition in well-defined formal models that allow abstraction to avoid state space explosion has been addressed in, e.g., [25]. An especially interesting and formally sound approach can be found in [9]. In this chapter, we develop a formal approach, inspired by techniques from the verification area, for **CRA!**. A particular challenge will be the consideration and explicit modeling of uncertainties in the formal model where there is no similar technique or need in the area of verification given.

2.3 Uncertainty Considerations

There exist intense studies on the effect of uncertainties on the system reliability for general engineering problems, cf. [38], as well as circuits and microarchitectures, cf. [27]. However, few studies focus on the cross-level reliability analysis of embedded systems in the presence of uncertainty arising from manufacturing tolerances, etc.

Uncertainty-Aware Analysis One example is a cross-level adaptive reliability prediction technique proposed in [17] that derives information from different levels of abstraction and allows to consider the simultaneous effects of process, voltage, temperature and aging variations, and soft errors on a processor. The authors of [18] propose a cross-level framework to analyze the combined impact of aging and process variation on the **SER!** (**SER!**) and static noise margin of memory arrays in near threshold voltage regimes. This framework enables to explore workload, as instruction per second, and cache configuration, as cache size and associativity, in order to minimize **SER!** and its variations for 6T and 8T SRAM cells. Contrary to all mentioned approaches, this chapter explicitly treats each effect of uncertainty during reliability analysis of a system. Proposed is an analysis technique that obtains the range of reliability that is achievable for a system given its configuration and the uncertainties of its components.

Uncertainty-Aware Optimization Optimization problems may be affected by various sources of uncertainty including perturbation of decision variables as well as effects of noise and approximation on objective functions [26]. In this work, uncertainty is explicitly modeled as variations in component failure rates and costs. The uncertainty propagates through reliability analysis and cost evaluation at system level and renders design objectives to be uncertain as well. To make correct decisions when comparing and discriminating implementations during **DSE!**, the employed optimization algorithm needs to take the uncertainty of the design objectives into account as well. The work in [44] proposes a mathematical approach to calculate the probability of an implementation dominating another, given all uncertain objectives follow either uniform or any discrete distributions. However, extending this approach to consider diversely distributed uncertain objectives requires solving difficult integrals demanding a huge computational effort. To this end, approximate simulation-based approaches, e. g., in [30], provide trade-offs between execution time and accuracy of calculating this probability. In [33], it is proposed to compare uncertain objectives with respect to their lower and upper bounds. However, this approach fails to distinguish largely overlapping intervals with even significantly different distributions. A lot of work has been proposed for problems with continuous search spaces and linear objective functions, see e. g.,[15]. However, typical embedded system design problems have discrete search spaces, non-linear and often not differentiable objective functions, and have to cope with stringent constraints. Thus, these optimization techniques cannot be applied without further investigation and modification. In [39], an approach based on an uncertainty-aware ***MOEA! (MOEA!)*** that targets reliability as one design objective is presented. The approach takes into account the uncertainty of the reliability of each system component and tries to maximize the robustness of the system. This chapter presents a novel uncertainty-aware multi-objective optimization approach applicable for **DSE!** of reliable systems at system level, see Sect. 4.

2.4 System-Level Design Fundamentals

This chapter targets the system-level design of embedded MPSoCs, typically specified by an application graph, a resource graph, and a set of possible task-to-resource mappings. The application graph includes a set of tasks to be executed and specifies the data and control flow among them. The resource graph consists of hardware resources, namely, processors and accelerators connected by communication infrastructures such as buses or networks-on-a-chip. The mappings specify which tasks can be executed on which resources. Figure 4 shows an example specification with three tasks t_i, $i \in [0 \ldots 2]$, five resources r_j, $j \in [0 \ldots 4]$, and eight mappings $m_{i,j}$ from t_i to r_j.

Implementation candidates are derived via system-level synthesis [10] performing the steps: (a) *Resource allocation* selects a subset of resources that are part of the implementation. (b) *Task binding* associates at least one instance of each task to an allocated resource by activating the respective task-to-resource mapping. (c) *Scheduling* determines a feasible start time for each task instance. An implementation is *feasible* if and only if all constraints regarding, e. g., communication, timing, or utilization are fulfilled. Figure 4 highlights a possible feasible implementation

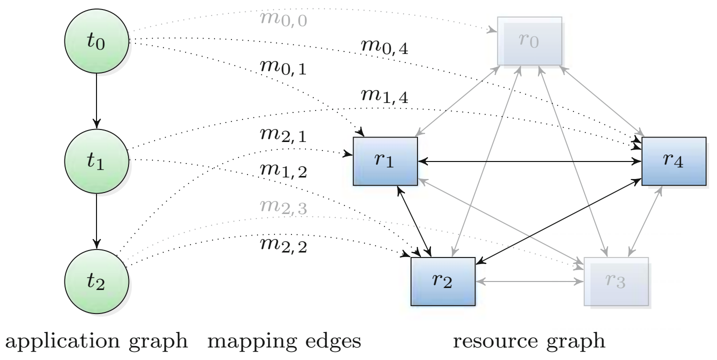


Fig. 4 A specification comprising (**a**) an application graph where edges indicate data dependencies of tasks, (**b**) a resource graph with edges representing dedicated communication between resources, and (**c**) a set of task-to-resource mappings which model possible execution of tasks on resources. A possible implementation candidate obtained by system-level synthesis is depicted with non-allocated resources and non-active bindings being grayed out

with non-allocated resources and non-activated mappings being grayed out. More details of the underlying system synthesis and **DSE!** in the context of reliability analysis and optimization can be found in [22, 43].

3 Compositional Reliability Analysis (CRA)

This section introduces models and methods for **CRA!** as proposed in [21]. Figure 5 shows a schematic view of **CRA!** and its required mechanisms. To realize a cross-level analysis, it encapsulates existing reliability analysis techniques in **CRN!**s (**CRN!**s) at multiple **RAL!**s (**RAL!**s). It tames analysis complexity within a certain **RAL!** using composition and decomposition and connects different **RAL!**s through adapters. Each **CRN!** applies an analysis step $\mathcal{Y}(t) = X(S)$ at a specific **RAL!** where X is a concrete analysis technique and S is a (sub)system. A **CRN!** derives a specific measure $\mathcal{Y}$ over time t. A **RAL!** in **CRA!** may combine several (design) abstraction levels where the same errors and, especially, their causes are significant.

Adjacent **RAL!**s are connected by the concept of *adapters* that have to perform three tasks: (a) *refinement* provides the data required for the analysis in the lower **RAL!**, (b) *data conversion* transforms the output measures from the lower **RAL!** to the input required at the higher **RAL!**, and (c) *abstraction* during both refinement and data conversion tames analysis complexity. A concrete example of **CRA!** describing a temperature-reliability adapter for MPSoCs is presented in Sect. 3.1.

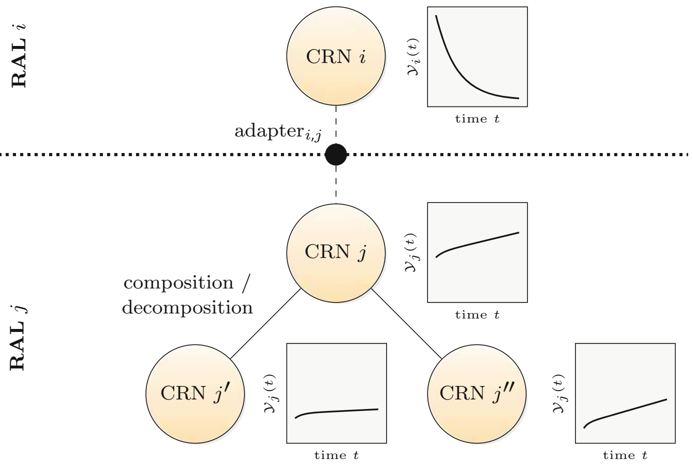


Fig. 5 A schematic view of **CRA!**

Another important aspect of **CRA!** concerns the feasibility of composition and decomposition with respect to reliability analysis. While, of course, composition and decomposition should reduce the complexity of the analysis, errors caused by approximation or abstraction should be bounded. For example, a rule to bound the approximation error of a decomposition D of a given system S into n subsystems $S_1, \ldots, S_n$ is as follows:

$$D(S) = \{S_1, \ldots, S_n\} \text{ is feasible, if } \exists\epsilon : |X(S) - \big(X(S_1) \circ \ldots \circ X(S_n)\big)| \leq \epsilon$$

with $\circ$ being an analysis-dependent operator, e. g., multiplication, and ϵ being the maximum approximation error. A special focus of these investigations is the proper handling of decomposed nodes that influence each other. Nowadays, hardly any subsystem of an embedded system is truly independent of all other subsystems. Thus, this rule should be extended as follows to consider both the truly independent individual properties of the decomposed nodes and their dependencies during composition C:

$$\begin{gathered} D(S) = \{S_1, \ldots, S_n\} \text{ is feasible, if } \exists\epsilon : \\ |X(S) - C\big(X(S_1) \circ \ldots \circ X(S_n), P(\{S_1, \ldots, S_n\})\big)| \leq \epsilon. \end{gathered} \tag{1}$$

In this case, the composition C not only takes into account the parts of the subsystem that can be analyzed independently, but also performs a corrective postprocessing P to take into account their interactions.

Similarly, we have developed rules for the connection of different **RAL!**s. The task of an adapter is to convert the measure $\mathcal{Y}$ used at the lower **RAL!** into the measure $\mathcal{Y}'$ used at the higher **RAL!**s, for example, $\mathcal{Y}' = A(\mathcal{Y})$. Especially because of the models and methods needed for converting from one **RAL!** to another, a thorough analysis of the function A needs to be carried out. In most cases, this function will not provide an exact result, but will require an abstraction such as by the determination of tight upper and lower bounds. Thus, the developed rules will define requirements for the functions in the adapter used for abstraction and data conversion.

3.1 CRA Case Study and Uncertainty Investigations

In [21], a concrete application of **CRA!** to realize a temperature-aware redundant task mapping approach is presented. In the following, a brief summary of the case study is given with focus being put on the aspect of uncertainty introduced due to the application of composition and decomposition. This further motivates the need to develop techniques to explicitly model and consider uncertainty during analysis and optimization as is presented in Sect. 4.

3.1.1 CRA Case Study

In the context of system-level design and especially the design of reliable embedded systems as introduced in Sect. 2, deploying redundant (software) tasks can be considered a rather cost-efficient technique to enhance system reliability. However, the resulting additional workload may lead to increased temperature and, thus, a reliability degradation of the (hardware) components executing the tasks. The case study in [21] combines three different techniques on three **RAL!**s: At the highest **RAL!**, a reliability analysis based on **BDD!** (**BDD!**), see, e. g.,[20], computes the system reliability of a complete 8-core MPSoC and requires the reliability function of each component (core) in the system. To determine the latter, an intermediate **RAL!** uses the behavioral analysis approach **RTC!** (**RTC!**) [45] to derive the upper bound for the workload of each core over time. This workload is passed to the lowest **RAL!** where this information is used to carry out a temperature simulation based on HotSpot [41] to deliver a temperature profile of each core. Using these temperature profiles and assuming electromigration as a fault model, [21] proposes an adapter that—based on the works in [14, 46]—delivers a temperature-aware reliability function for each core back to the highest **RAL!** in order to complete the system analysis.

3.1.2 Uncertainty Investigations

As given in Eq. 1, composition/decomposition may result in an imprecision ϵ_o of an output measure $o \in O$. In [19], we present techniques for formal decomposition and composition for **CRN!**s that describe the system via Boolean formulas, typically used by **BDD!**s, Fault Trees, etc. Here, functional correlations between components are fully captured in the Boolean formulas, and we propose an exact composition/decomposition scheme on the basis of early quantification. However, correlations are typically non-functional, with heat dissipation between adjacent cores being a prominent one. Consider again the case study described before and Fig. 6: Not decomposing the system into individual cores results in a temperature simulation of all cores at the lowest level, implicitly including the effect of heat dissipation in-between cores, see Fig. 6 (top-left). A naive decomposition could decompose the system into independent cores such that the workload of each core is determined and a reliability function would be gathered by per-core temperature simulations on the lowest level, see Fig. 6 (middle-left). This, however, would completely neglect the effect of heat dissipation between cores. As a third option, [21] investigates a *corrective postprocessing* within the adapter between the lower levels where the workload of cores and the temperature simulation are analyzed independently, while a simple model that considers the distance and steady-state temperature of each core is used to approximate the respective heat flow, see Fig. 6 (bottom-left).

The imprecision resulting from the three discussed decomposition variants is given in Fig. 6 (right), derived from ≈8000 different system implementations

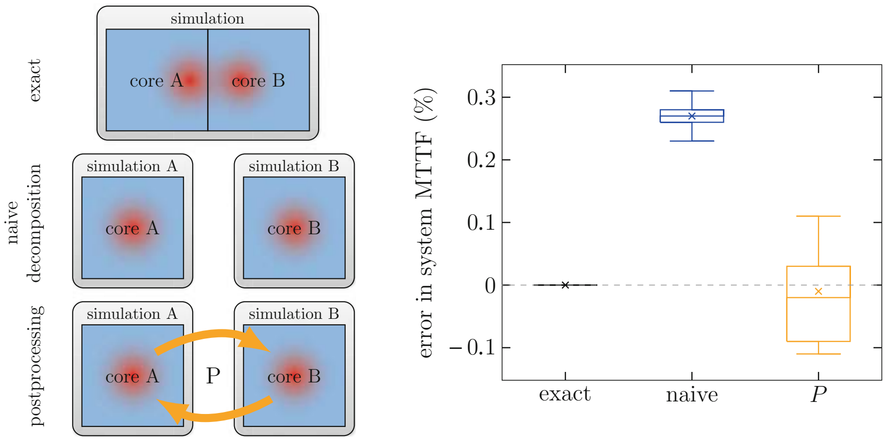


Fig. 6 A simulation of two cores captures heat dissipation (exact, top) while a decomposition (naive, middle) is unable to capture heat dissipation between cores. A corrective postprocessing (P, bottom) enables to reduce analysis complexity while providing a basic notion of heat dissipation. The resulting imprecisions in percentage on system-wide **MTTF!** are depicted on the right

analyzed as part of a **DSE!**: While no decomposition is treated as an exact base value—with respect to heat dissipation being considered and not the overall exactness of the simulation—the naive decomposition constantly overestimates the system-wide **MTTF!** by ≈26%. On the other hand, the corrective postprocessing delivers results with a rather good match in terms of the median and average error, but also shows that the correction may come at errors of up to ≈10%. At the same time, compared to the complete simulation, the decomposition including corrective postprocessing achieves a ≈2 × average speed-up. These results further motivate the need for analysis and optimization techniques—as presented in the next section—that can explicitly model uncertainty such as the shown imprecision.

4 Uncertainty in Reliability Analysis and Optimization

To design and optimize systems for reliability, existing uncertainties in their environment and internal states, see Fig. 1, must be explicitly integrated into reliability analysis techniques. Implicit uncertainty modeling hides the effects of controllable and non-controllable uncertainties, e. g., into a single reliability function, and fails to distinguish between them. On the other hand, explicit modeling determines the range of achievable reliability of a component or subsystem, e. g., using upper and lower bound functions.

We introduce two solutions for uncertainty modeling: (a) using upper and lower bounding curves for the achievable reliability and (b) abstracting various uncertainties into a finite set of typical use cases and providing a system reliability

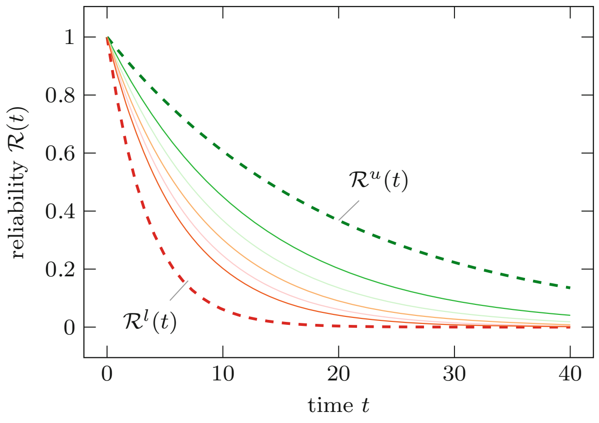


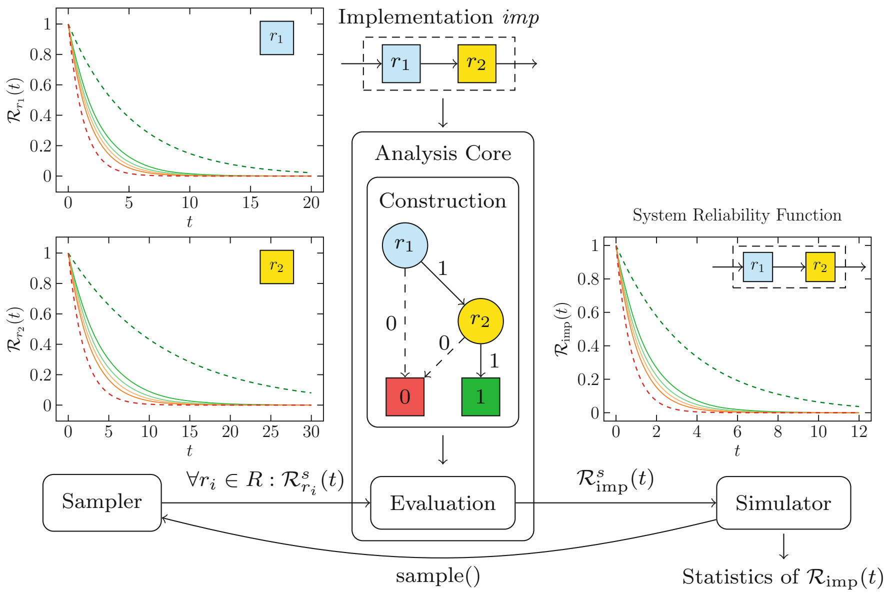
Fig. 7 Reliability functions $\mathcal{R}(t)$ that result from uncertainties in the internal heat dissipation of a silicon system that arise from, e. g., changing task binding on a processing unit. Shown is a range that is determined by an upper $\mathcal{R}^u(t)$ and a lower bound $\mathcal{R}^l(t)$ reliability function and the reliability functions for 5 use cases, e. g., 5 favored task schedules that show a distribution within the range

function for each case, see Fig. 7. While the former offers a range and abstracts from the distributions in between bounds, the latter variant explicitly determines important cases in that range, but of course, comes with an increased complexity. This section covers both approaches and assumes that uncertainty obtained from lower abstraction levels is available at higher levels as known distributions or sampled data.

As introduced earlier, incorporating reliability-increasing techniques into a system at design time may deteriorate other design objectives. Due to the explicit modeling of uncertainties, a multi-objective uncertainty-aware optimization becomes necessary. Given that the system reliability is no more a single value, optimization algorithms must be able to handle uncertain objectives given as probability distributions, a set of samples or upper and lower bound curves, and allow for a quantitative comparison of different designs.

4.1 Uncertainty-Aware Reliability Analysis

The uncertainty-aware reliability analysis technique introduced in the following is originally proposed in [29]. It models the reliability of a component r with uncertain characteristics $\mathcal{U}_r$ using reliability functions $\mathcal{R}_r(t)$ that are distributed within given lower and upper bound reliability functions, i. e., $\mathcal{U}_r = \left[\mathcal{R}_r^l(t), \mathcal{R}_r^u(t)\right]$. A *sampler* is used to take $\mathcal{U}_r$ as input and deliver a sampled reliability function $\mathcal{R}_r^s(t)$ with $\mathcal{R}_r^l(t) \leq \mathcal{R}_r^s(t) \leq \mathcal{R}_r^u(t)$. It ensures that the sampled reliability functions follow the intended distribution within the given bounds, and enables



$\mathcal{R}_{r_1}(t)$
r_1
t
$\mathcal{R}_{r_2}(t)$
r_2
t
Implementation *imp*
r_1
r_2
Analysis Core
Construction
r_1
r_2
0
1
Evaluation
System Reliability Function
$\mathcal{R}_{\text{imp}}(t)$
t
Sampler
$\forall r_i \in R : \mathcal{R}^s_{r_i}(t)$
$\mathcal{R}^s_{\text{imp}}(t)$
Simulator
sample()
Statistics of $\mathcal{R}_{\text{imp}}(t)$

Fig. 8 An overview of the proposed uncertainty-aware reliability analysis

the consideration of arbitrary distributions, in particular well-known discrete and continuous distributions.

In practice, component reliability is typically derived from measurements that are fitted to closed-form exponential ($\mathcal{R}_r(t) = e^{-\lambda_r \cdot t}$) and Weibull ($\mathcal{R}_r(t) = e^{-\lambda_r \cdot t^{\beta_r}}$) reliability functions with λ being the component's failure rate. The uncertainty model $\mathcal{U}_r$ includes a set of uncertain parameters P_r, distributed within the bounds $[P_r^l, P_r^u]$. The sampler takes a sample from each parameter $p_r \in P_r$ and constructs a sample reliability function. For example, for an exponential distribution with bounds $[\lambda_r^l, \lambda_r^u] = [0.0095, 0.0099]$, a sample reliability function $\mathcal{R}_r^s(t) = e^{-0.0098\, t}$ can be generated.

The overall flow of the analysis approach is shown in Fig. 8 and includes the following steps: (a) The sampler samples a reliability function $\mathcal{R}_r^s(t)$ from the uncertainty distribution of each component r, (b) an analysis core uses these samples and calculates a sample reliability function for the given system implementation $\mathcal{R}^s_{\text{imp}}(t)$, and (c) a statistical *simulator* collects a set of sampled reliability functions $\Phi_{\text{imp}} = \bigcup_{s=1}^{n} \{\mathcal{R}^s_{\text{imp}}(t)\}$ and constructs the uncertainty distribution of the system reliability.

The analysis core can be realized by any existing technique that requires a reliability function of each component and calculates the system reliability function. In the concrete case, Fig. 8 shows a formal technique based on **BDD!**s that models the reliability of a system implementation with two components in series. The statistical simulator determines the number of required samples n to later obtain desired statistics like mean and quantiles from Φ_{imp} with a guaranteed confidence level.[1] As an example, for each sample $\mathcal{R}^s_{\text{imp}}(t)$ in Φ_{imp}, the **MTTF!** can be calculated using the integration below:

$$\textbf{MTTF!}_s = \int_0^\infty t \cdot f^s(t)\,\mathrm{d}t \quad \text{where} \quad f^s(t) = -\frac{\mathrm{d}\mathcal{R}^s_{\text{imp}}(t)}{\mathrm{d}t}. \tag{2}$$

Using sample **MTTF!**s, design objectives such as the best-, worst-, and average-case **MTTF!** can be derived.

4.1.1 Uncertainty Correlation

To model any existing correlation between uncertain parameters of system components, we investigate whether they are exposed to common uncertainty sources, and are, thus, subject to correlative variations. Take temperature as an example: Components that are fabricated in the same package may be exposed to the same temperature, which means their reliability characteristics can be considered in a *correlation group*, whereas components in different packages might be considered independent. Assuming that the uncertainty sources and the correlation groups are given, we introduce models for obtaining correlated samples from the uncertainty distribution of component reliability functions in [29, 31]: To sample from an uncertain parameter p, we check if it is a member of any correlation group or not. If p is a member of G, we first generate a random probability g for the group G at the beginning of each implementation evaluation step and then calculate a sample from p using the inverse **CDF!** (**CDF!**) of the probability distribution of p at point g. Otherwise, a sample is taken independently from the distribution of p. Note that since the uncertain parameters in a correlation group might be differently distributed, returning the same quantile g from their distributions does not necessarily yield the same value, see Fig. 9. Thus, through sampling, the uncertain parameters in G vary together, and their variations are independent of those of the parameters outside G.

4.2 Uncertainty-Aware Multi-Objective Optimization

Finally, to enable the optimization of system implementations with multiple uncertain objectives, we propose an uncertainty-aware framework in [29]. It extends a state-of-the-art **DSE!** [43] and employs a **MOEA!** as the optimization core. These techniques introduce *dominance criteria* to compare different implementations and select which one to store in an *archive* and vary for the next iteration.

[1] Efficient sampling techniques [8] can be used to reduce the number of required samples.

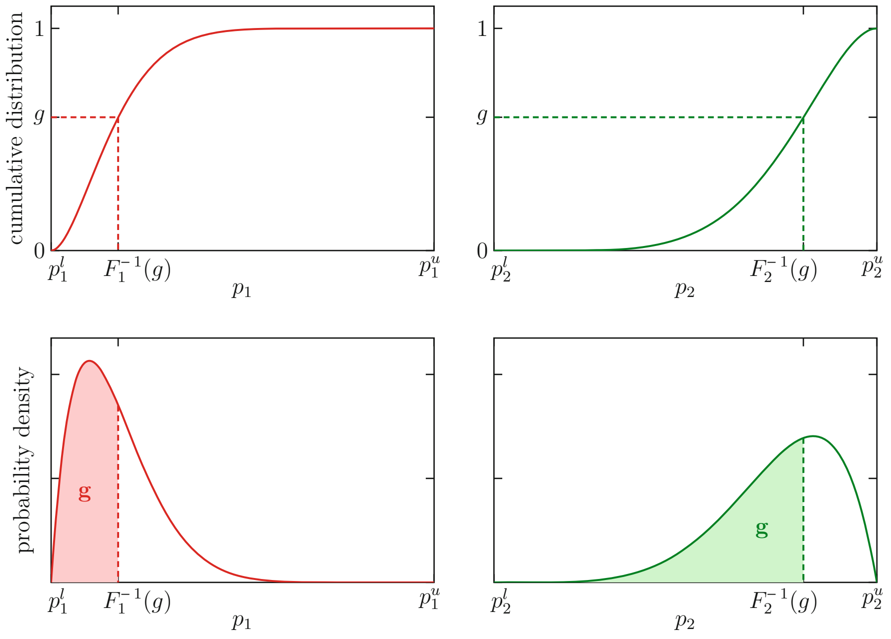


Fig. 9 Generating samples for correlated uncertain parameters p_1 and p_2

To maximize m objectives $O_1, \ldots, O_m$, each being a single value, the dominance of two implementations A and B is defined as follows:

$$A \succ B \iff \forall i \in [1, v] : O_A(i) \geq O_B(i) \ \wedge \ \exists j \in [1, v] : O_A(j) > O_B(j) \tag{3}$$

Here, $A \succ B$ means "A dominates B" and $O_A(j) > O_B(j)$ means "A is better than B in the j-th objective." Since this dominance criterion compares each of the m objectives independently, we refer to $O(i)$ as O for brevity.

The proposed uncertainty-aware optimization compares uncertain objectives using the following three-stage algorithm: (a) If the intervals of O, specified by the lower bound O^l and upper bound O^u, of two implementations A and B do not overlap, one is trivially better ($>$) than the other. (b) If the intervals overlap, we check if one objective is significantly better with respect to an *average criterion*, e. g., mean, mode, or median. (c) If the average criterion does not find a preference, a *spread criterion* compares objectives based on their deviation, e. g., standard deviation, variance, or quantile intervals, and judges whether one is considerably better. In case none of the three stages determines that one objective is better, the objectives are considered equal. The flow of this comparison operator is illustrated in Fig. 10.

To find if one uncertain objective has significantly better average O^{avg} or deviation O^{dev} compared to the other, we use two configurable threshold values

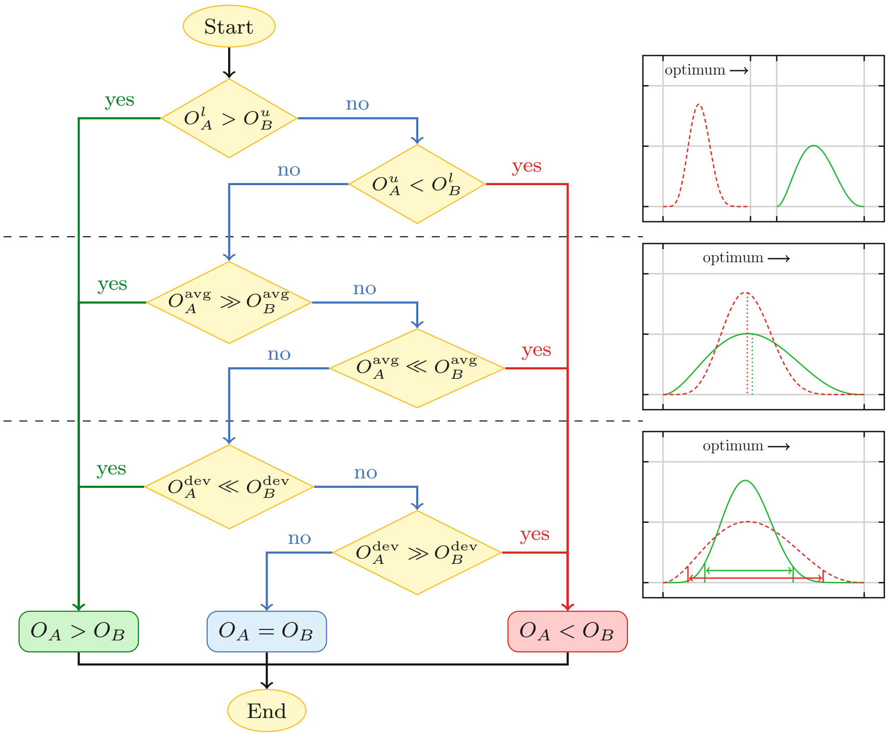

Start
$O_A^l > O_B^u$
yes
no
$O_A^u < O_B^l$
no
yes
$O_A^{\text{avg}} \gg O_B^{\text{avg}}$
yes
no
$O_A^{\text{avg}} \ll O_B^{\text{avg}}$
no
yes
$O_A^{\text{dev}} \ll O_B^{\text{dev}}$
yes
no
$O_A^{\text{dev}} \gg O_B^{\text{dev}}$
no
yes
$O_A > O_B$
$O_A = O_B$
$O_A < O_B$
End
optimum ⟶
optimum ⟶
optimum ⟶

Fig. 10 The flow of the proposed three-stage comparison operator

ε^{avg} and ε^{dev}, respectively. For the average criterion, a configurable threshold value ε^{avg} determines if the difference of the considered average-case objective values is significant with respect to the given objective bounds. This enables to control the sensitivity of the second stage of the comparison:

$$\frac{O_A^{\text{avg}} - O_B^{\text{avg}}}{\max\left(O_A^u, O_B^u\right) - \min\left(O_A^l, O_B^l\right)} \geq \varepsilon^{\text{avg}}. \tag{4}$$

Here, $\varepsilon^{\text{avg}} = 0$ always prefers the objective with better average case, while $\varepsilon^{\text{avg}} = 1$ renders the average criterion ineffective since the left-hand side of Eq. (4) is always less than one. Thus, the scope of ε^{avg} must be carefully selected based on the objective's criticality to guarantee a required precision.

The spread criterion prefers the objective value with smaller deviation and uses a threshold value ε^{dev} to control the sensitivity of the comparison, i. e.,

$$\frac{O_B^{\text{dev}} - O_A^{\text{dev}}}{O_A^{\text{dev}} + O_B^{\text{dev}}} \geq \varepsilon^{\text{dev}} \Rightarrow O_A > O_B. \tag{5}$$

Given $\varepsilon^{\text{dev}} = 0$, any small difference between O_A^{dev} and O_B^{dev} is reckoned, which can lead to crowding in the solution archive. It causes solution A which is indeed weakly dominated by another solution B to be regarded as non-dominated because one of its uncertain objectives has a slightly better deviation than the corresponding objective of B. On the other hand, the spread criterion becomes ineffective if $\varepsilon^{\text{dev}} = 1$ and any significant difference between deviations of two uncertain objectives would be overlooked. Therefore, the value of ε^{dev} must be carefully selected.

Note that the statistics of an uncertain objective O required in the proposed comparison operator are calculated using samples from its distribution. Given a set of n samples for O, its variance can be calculated as follows:

$$O^{\sigma^2} = \frac{1}{n}\sum_{i=j}^{n}\left(O^j - O^\mu\right)^2 \text{ where } O^\mu = \frac{1}{n}\sum_{i=j}^{n} O^j, \tag{6}$$

with O^μ denoting the mean of the distribution of O. Moreover, to find the q^{th} quantile of this distribution, we use the *inverse empirical distribution function* which traverses the samples in the ascending order and returns the very first sample after the $q\%$ smallest samples.

Figure 11 shows the resulting Pareto fronts for optimizing **MTTF!** and cost of an H.264 specification using the proposed comparison operator vs. a common uncertainty-oblivious approach that compares instances of uncertain objectives with respect to their mean values. The specification incorporates 15 resources, 66 tasks, and 275 mappings. The proposed operator uses mean and 95% quantile interval as the average and spread criteria, respectively. Depicted are the mean values,

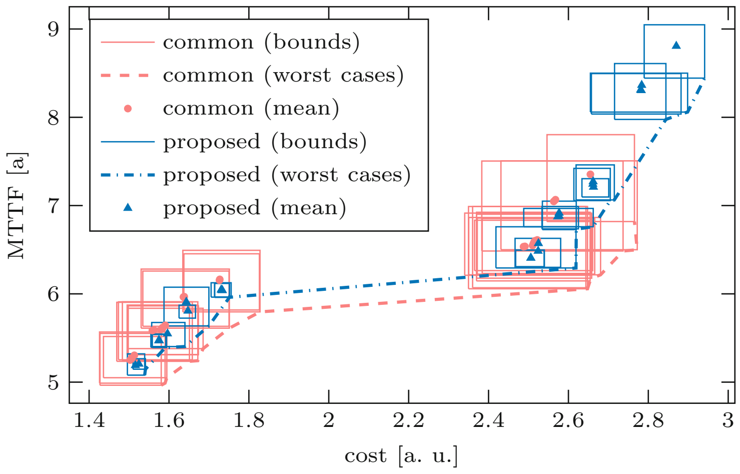


Fig. 11 Pareto fronts when optimizing **MTTF!** and cost of an H.264 encoder/decoder using a common uncertainty-oblivious approach vs. the proposed comparison operator

boxes enclosing the uncertainty distributions, and lines connecting the worst cases. The results show that the proposed comparison operator enables the **DSE!** to find implementation candidates of smaller uncertainty, and yet comparable quality in the average case.

5 Conclusion

Progressive shrinkage in electronic devices has brought them vulnerabilities to manufacturing tolerances as well as environmental and operational changes. The induced uncertainty in component reliability might propagate to system level, which necessitates uncertainty-aware cross-level reliability analysis. This chapter presents a cross-level reliability analysis methodology that enables handling the ever increasing analysis complexity of embedded systems under the impact of different uncertainties. It combines various reliability analysis techniques across different abstraction levels by introducing mechanisms for (a) the composition and decomposition of the system during analysis and (b) converting analysis data over abstraction levels through adapters. It also provides an explicit modeling of uncertainties and their correlations. The proposed methodology is incorporated in an automatic reliability analysis tool that enables the evaluation of reliability-increasing techniques within a **DSE!** framework. The **DSE!** employs meta-heuristic optimization algorithms and is capable of comparing system implementation candidates with objectives regarded as probability distributions.

Acknowledgments This work is supported in part by the German Research Foundation (DFG) as associated project CRAU (GL 819/1-2 & TE 163/16) of the priority program Dependable Embedded Systems (SPP 1500).

References

1. Al Faruque, M., Jahn, J., Ebi, T., Henkel, J.: Runtime thermal management using software agents for multi-and many-core architectures. IEEE Design Test Comput. **27**(6), 58–68 (2010)
2. Aliee, H.: Reliability analysis and optimization of embedded systems using stochastic logic and importance measures. Doctoral Thesis, Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) (2017)
3. Aliee, H., Glaß, M., Reimann, F., Teich, J.: Automatic success tree-based reliability analysis for the consideration of transient and permanent faults. In: Design, Automation & Test in Europe (DATE), pp. 1621–1626 (2013)
4. Aliee, H., Glaß, M., Khosravi, F., Teich, J.: An efficient technique for computing importance measures in automatic design of dependable embedded systems. In: Hardware/Software Codesign and System Synthesis (CODES+ISSS), pp. 34:1–34:10 (2014)
5. Aliee, H., Borgonovo, E., Glaß, M., Teich, J.: Importance measures in time-dependent reliability analysis and system design. In: European Safety and Reliability Conference (ESREL) (2015)

6. Aliee, H., Borgonovo, E., Glaß, M., Teich, J.: On the Boolean extension of the Birnbaum importance to non-coherent systems. Reliab. Eng. Syst. Safe. **160**, 191–200 (2016)
7. Aliee, H., Banaiyianmofrad, A., Glaß, M., Teich, J., Dutt, N.: Redundancy-aware design space exploration for memory reliability in many-cores. In: Methoden und Beschreibungssprachen zur Modellierung und Verifikation von Schaltungen und Systemen (MBMW) (2017)
8. Aliee, H., Khosravi, F., Teich, J.: Efficient treatment of uncertainty in system reliability analysis using importance measures. In: 2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN) (2019)
9. Bensalem, S., Bozga, M., Sifakis, J., Nguyen, T.: Compositional verification for component-based systems and application. In: Automated Technology for Verification and Analysis (ATVA), pp. 64–79 (2008)
10. Blickle, T., Teich, J., Thiele, L.: System-level synthesis using evolutionary algorithms. Des. Autom. Embed. Syst. **3**(1), 23–58 (1998)
11. Borgonovo, E., Aliee, H., Glaß, M., Teich, J.: A new time-independent reliability importance measure. Eur. J. Oper. Res. **254**(2), 427–442 (2016)
12. Bowen, J., Stavridou, V.: Safety-critical systems, formal methods and standards. Softw. Eng. J. **8**, 189–189 (1993)
13. Coit, D.W., Smith, A.E.: Reliability optimization of series-parallel systems using a genetic algorithm. IEEE Trans. Reliab. **45**(1), 254–260 (1996)
14. Council, J.E.D.E.: Failure Mechanisms and Models for Semiconductor Devices. JEDEC Publication JEP122-B (2003)
15. Deb, K., Gupta, H.: Searching for robust Pareto-optimal solutions in multi-objective optimization. In: Evolutionary Multi-Criterion Optimization (EMO), pp. 150–164 (2005)
16. Eles, P., Izosimov, V., Pop, P., Peng, Z.: Synthesis of fault-tolerant embedded systems. In: Design, Automation & Test in Europe (DATE), pp. 1117–1122 (2008)
17. Farahani, B., Safari, S.: A cross-layer approach to online adaptive reliability prediction of transient faults. In: Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFTS), pp. 215–220 (2015)
18. Gebregiorgis, A., Kiamehr, S., Oboril, F., Bishnoi, R., Tahoori, M.B.: A cross-layer analysis of soft error, aging and process variation in near threshold computing. In: Design, Automation & Test in Europe (DATE), pp. 205–210 (2016)
19. Glaß, M., Lukasiewycz, M., Haubelt, C., Teich, J.: Towards scalable system-level reliability analysis. In: Design Automation Conference (DAC), pp. 234–239 (2010)
20. Glaß, M., Lukasiewycz, M., Reimann, F., Haubelt, C., Teich, J.: Symbolic system level reliability analysis. In: International Conference on Computer-Aided Design (ICCAD), pp. 185–189 (2010)
21. Glaß, M., Yu, H., Reimann, F., Teich, J.: Cross-level compositional reliability analysis for embedded systems. In: International Conference on Computer Safety, Reliability and Security (SAFECOMP), pp. 111–124 (2012)
22. Glaß, M., Teich, J., Lukasiewycz, M., Reimann, F.: Hybrid Optimization Techniques for System-Level Design Space Exploration, pp. 1–31. Springer, Dordrecht (2017)
23. Gu, Z., Zhu, C., Shang, L., Dick, R.: Application-specific MPSoC reliability optimization. IEEE Trans. Very Large Scale Integr. Syst. **16**(5), 603 (2008)
24. Herkersdorf, A., Aliee, H., Engel, M., Glaß, M., Gimmler-Dumont, C., Henkel, J., Kleeberger, V., Kochte, M., Kühn, J., Mueller-Gritschneider, D., Nassif, S., Rauchfuss, H., Rosenstiel, W., Schlichtmann, U., Shafique, M., Tahoori, M., Teich, J., Wehn, N., Weis, C., Wunderlich, H.: Resilience articulation point (RAP): cross-layer dependability modeling for nanometer system-on-chip resilience. Microelectr. Reliab. **54**(6–7), 1066–1074 (2014)
25. Hooman, J.: Specification and Compositional Verification of Real-Time Systems. Springer, Berlin (1991)
26. Jin, Y., Branke, J.: Evolutionary optimization in uncertain environments–a survey. IEEE Trans. Evol. Comput. **9**(3), 303–317 (2005)
27. Jung, S., Lee, J., Kim, J.: Variability-aware, discrete optimization for analog circuits. IEEE Trans. Comput. Aided Design Integr. Circuits Syst. **33**(8), 1117–1130 (2014)

28. Khosravi, F., Reimann, F., Glaß, M., Teich, J.: Multi-objective local-search optimization using reliability importance measuring. In: Design Automation Conference (DAC), pp. 1–6 (2014)
29. Khosravi, F., Müller, M., Glaß, M., Teich, J.: Uncertainty-aware reliability analysis and optimization. In: Design, Automation & Test in Europe (DATE), pp. 97–102 (2015)
30. Khosravi, F., Borst, M., Teich, J.: Probabilistic dominance in robust multi-objective optimization. In: IEEE Congress on Evolutionary Computation (CEC), pp. 1597–1604 (2018)
31. Khosravi, F., Müller, M., Glaß, M., Teich, J.: Simulation-based uncertainty correlation modeling in reliability analysis. The Institution of Mechanical Engineers, Part O J. Risk Reliab. **232**, 725–737 (2018)
32. Liang, S., Zhang, Z., Pei, T., Li, R., Li, Y., Peng, L.: Reliability tests and improvements for Sc-contacted n-type carbon nanotube transistors. Nano Res. **6**(7), 535–545 (2013)
33. Limbourg, P.: Multi-objective optimization of problems with epistemic uncertainty. In: Evolutionary Multi-Criterion Optimization (EMO), pp. 413–427 (2005)
34. Meyer, B., Hartman, A., Thomas, D.: Cost-effective slack allocation for lifetime improvement in NoC-based MPSoCs. In: Design, Automation & Test in Europe (DATE), pp. 1596–1601 (2010)
35. Misra, K.B.: Reliability Analysis and Prediction: A Methodology Oriented Treatment, vol. 15. Elsevier, Amsterdam (2012)
36. Narayanan, V., Xie, Y.: Reliability concerns in embedded system designs. Computer **39**, 118–120 (2006)
37. Pham, T.T., Defago, X., Huynh, Q.T.: Reliability prediction for component-based software systems: dealing with concurrent and propagating errors. Sci. Comput. Program. **97**, 426–457 (2015)
38. Rakshit, P., Konar, A., Das, S.: Noisy evolutionary optimization algorithms–a comprehensive survey. Swarm Evol. Comput. **33**, 18–45 (2017)
39. Salazar, A.D., Rocco, S.C.: Solving advanced multi-objective robust designs by means of multiple objective evolutionary algorithms (MOEA): a reliability application. Reliab. Eng. Syst. Safe. **92**(6), 697–706 (2007)
40. Sander, B., Schnerr, J., Bringmann, O.: ESL power analysis of embedded processors for temperature and reliability estimations. In: Hardware/Software Codesign and System Synthesis (CODES+ISSS), pp. 239–248 (2009)
41. Skadron, K., Stan, M.R., Huang, W., Velusamy, S., Sankaranarayanan, K., Tarjan, D.: Temperature-aware microarchitecture. In: 30th Annual International Symposium on Computer Architecture (ISCA), pp. 2–13 (2003)
42. Stathis, J.: Reliability limits for the gate insulator in CMOS technology. IBM J. Res. Dev. **46**(2–3), 265–286 (2002)
43. Streichert, T., Glaß, M., Haubelt, C., Teich, J.: Design space exploration of reliable networked embedded systems. J. Syst. Architect. **53**(10), 751–763 (2007)
44. Teich, J.: Pareto-front exploration with uncertain objectives. In: Evolutionary Multi-Criterion Optimization (EMO), pp. 314–328 (2001)
45. Wandeler, E., Thiele, L.: Real-Time Calculus (RTC) Toolbox (2006). http://www.mpa.ethz.ch/Rtctoolbox
46. Xiang, Y., Chantem, T., Dick, R.P., Hu, X.S., Shang, L.: System-level reliability modeling for MPSoCs. In: Hardware/Software Codesign and System Synthesis (CODES+ISSS), pp. 297–306 (2010)
47. Xie, Y., Li, L., Kandemir, M., Vijaykrishnan, N., Irwin, M.: Reliability-aware co-synthesis for embedded systems. J. VLSI Signal Proce. **49**(1), 87–99 (2007)

Robust Computing for Machine Learning-Based Systems

Muhammad Abdullah Hanif, Faiq Khalid, Rachmad Vidya Wicaksana Putra, Mohammad Taghi Teimoori, Florian Kriebel, Jeff (Jun) Zhang, Kang Liu, Semeen Rehman, Theocharis Theocharides, Alessandro Artusi, Siddharth Garg, and Muhammad Shafique

1 Introduction

Machine learning (ML) has emerged as the principal tool for performing complex tasks which are impractical (if not impossible) to code by humans. ML techniques provide machines the capability to learn from experience and thereby learn to perform complex tasks without much (if any) human intervention. Over the past decades, many ML algorithms have been proposed. However, Deep Learning (DL), using Deep Neural Networks (DNNs), has shown state-of-the-art accuracy, even surpassing human-level accuracy in some cases, for many applications [31]. These applications include, but are not limited to, object detection and localization, speech recognition, language translation, and video processing [31].

The state-of-the-art performance of the DL-based methods has also led to the use of DNNs in complex safety-critical applications, for example, autonomous driving [11] and smart healthcare [10]. DNNs are intrinsically computationally

M. A. Hanif (✉) · F. Khalid · R. V. W. Putra · M. T. Teimoori · F. Kriebel · S. Rehman
M. Shafique
Technische Universität Wien (TU Wien), Vienna, Austria
e-mail: muhammad.hanif@tuwien.ac.at; faiq.khalid@tuwien.ac.at; rachmad.putra@tuwien.ac.at; florian.kriebel@tuwien.ac.at; semeen.rehman@tuwien.ac.at; muhammad.shafique@tuwien.ac.at

J. Zhang · K. Liu · S. Garg
New York University, New York, NY, USA
e-mail: jeffjunzhang@nyu.edu; kang.liu@nyu.edu; sg175@nyu.edu

T. Theocharides
University of Cyprus, Nicosia, Cyprus
e-mail: ttheocharides@ucy.ac.cy

A. Artusi
University of Cyprus, Nicosia, Cyprus

MRG DeepCamera RISE, Nicosia, Cyprus

J. Henkel, N. Dutt (eds.), *Dependable Embedded Systems*, Embedded Systems,
https://doi.org/10.1007/978-3-030-52017-5_20

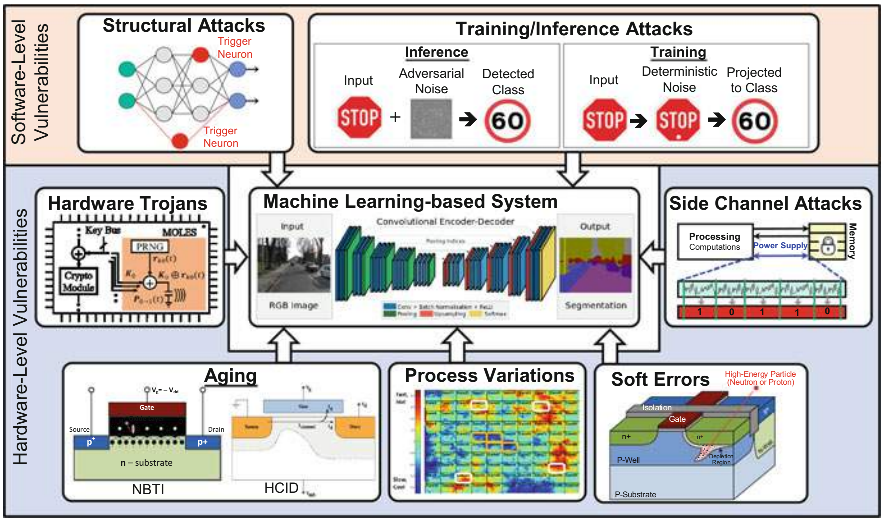
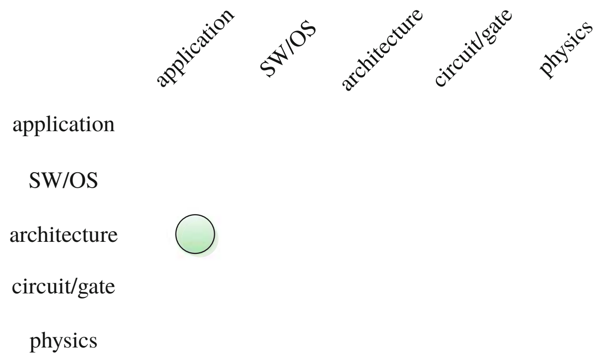


Fig. 1 Overview of different reliability and security vulnerabilities to machine learning-based systems. (Picture sources: [47, 49])

Fig. 2 Main abstraction layers of embedded systems and this chapter's major (green, solid) contributions



intensive and also require high memory resources [53]. Current research mainly focuses on the development of less computationally intensive and resource-efficient DNNs that can offer high accuracy, and energy and performance efficient DNN accelerators for ML-based applications [1, 18, 23, 29, 34, 36, 37, 44, 53]. *However, when considered for safety-critical applications, the robustness of these DNN-based systems to different reliability and security vulnerabilities also becomes one of the foremost objectives.* An overview of different types of vulnerabilities in ML-based systems is shown in Fig. 1, which are discussed from the architectural- and application-layer perspective in this chapter. Figure 2 shows the abstraction layers in the context of the SPP 1500 covered in this chapter.

Reliability Threats: In hardware design, reliability is the ability of the hardware to perform as intended for a specified duration, i.e., the lifetime of the hardware. There are a number of hardware related vulnerabilities that can disrupt the functionality of a digital system in its lifetime.

1. **Soft Errors** are transient faults caused by high energy particle strikes. These faults surface at hardware-layer as bit-flips and can propagate to the application layer resulting in incorrect output.
2. **Aging** is the gradual degradation of the hardware due to different physical phenomena like Hot carrier Injection (HCI), Negative-Bias Temperature Instability (NBTI), and Electromigration (EM). It leads to timing errors and eventually can also lead to permanent faults [56].
3. **Process variations** are the imperfections caused by the variations in the fabrication process of the chips. This can lead to variations in the timing and leakage power characteristics within a chip as well as across different chips [45].

Apart from the above-listed vulnerabilities, environmental conditions can also affect the reliability of a system. Such factors include temperature, altitude, high electric fields, etc.

A number of techniques have been proposed for improving the resilience of the systems against the reliability threats. However, most of these mitigation techniques are based on redundancy, for example, DMR: dual modular redundancy [58] and TMR: triple modular redundancy [35]. The redundancy based approaches, although considered to be very effective for other application domains [19], are highly inefficient for DNN-based systems because of the compute intensive nature of the DNNs [48], and may incur significant area, power/energy, and performance overheads. *Hence, a completely new set of resource-efficient reliability mechanisms is required for robust machine learning systems.* A list of techniques proposed for improving the reliability of DNN-based systems, which are later discussed in the following sections of the chapter, are mentioned in Fig. 3.

Security Threats: In system design, security is defined as the property of a system to ensure the confidentiality, integrity, and availability of the hardware and the data while performing the assigned tasks. There are several security vulnerabilities that can be exploited to perform security attacks.

1. **Data Manipulation:** The input data or data during inter-/intra-module communication in a system can be manipulated to perform several security attacks. For example, in DNNs, the training dataset and the inference data can be manipulated to perform misclassification or confidence reduction attacks [17, 24, 26, 27, 43, 51].
2. **Denial-of-Service:** A tiny piece of code/hardware or flooding the communication channels can be used to trigger the malfunctioning or failure of the system. For example, in DNNs, adding an extra neuron/set of neurons [17] or introducing the kill switch in DNN-based hardware can lead to system failure or malfunctioning, i.e., misclassification.

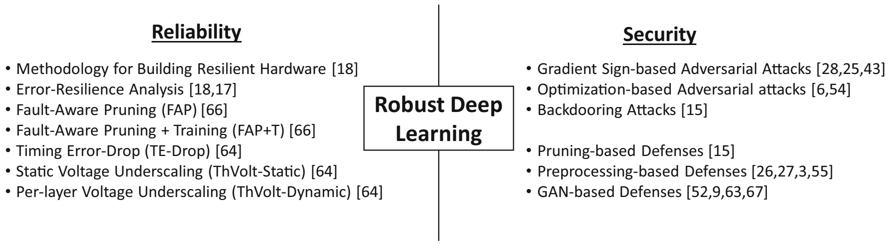

Reliability
• Methodology for Building Resilient Hardware [18]
• Error-Resilience Analysis [18,17]
• Fault-Aware Pruning (FAP) [66]
• Fault-Aware Pruning + Training (FAP+T) [66]
• Timing Error-Drop (TE-Drop) [64]
• Static Voltage Underscaling (ThVolt-Static) [64]
• Per-layer Voltage Underscaling (ThVolt-Dynamic) [64]
Robust Deep Learning
Security
• Gradient Sign-based Adversarial Attacks [28,25,43]
• Optimization-based Adversarial attacks [6,54]
• Backdooring Attacks [15]
• Pruning-based Defenses [15]
• Preprocessing-based Defenses [26,27,3,55]
• GAN-based Defenses [52,9,63,67]

Fig. 3 Overview of the works discussed in this chapter for addressing reliability and security vulnerabilities of deep learning-based systems

3. **Data/IP Stealing:** The side-channel information (in hardware, power, timing, and loopholes or behavior leaking properties of the algorithms) can be exploited to steal the confidential information. For example, in DNNs, the gradient information can be used to steal trained model [50, 57, 60].

Several countermeasures have been developed to address these threats, but most of these defenses are either based on obfuscation or run-time monitoring [3, 22]. These techniques are very effective for traditional systems, however, DNN-based systems require different approaches because of their unique security vulnerabilities, i.e., training/inference data manipulation. Some of the techniques proposed for addressing the security of DNN-based systems are listed in Fig. 3 and are later discussed in the chapter.

In the following sections, we discuss:

1. A brief *overview of DNNs and the hardware accelerators* used for efficiently processing these networks.
2. In Sect. 3, we present our *methodology for building reliable systems* and discuss *techniques for mitigating permanent and timing errors*.
3. The *security vulnerabilities* in different types of DNNs are discussed in Sect. 4.
4. *Open challenges* and further *research opportunities* for building robust systems for ML-based safety-critical applications

2 Preliminaries

2.1 Deep Neural Networks

A neural network can be described as a network of interconnected neurons. *Neurons* are the fundamental computational units in a neural network where each neuron performs a weighted sum of inputs (dot-product operation), using the inputs and the weights associated with each input connection of the neuron. Each output is then (optionally) passed through an *activation function* which introduces non-linearity and thereby allows the network to learn complex classification boundaries.

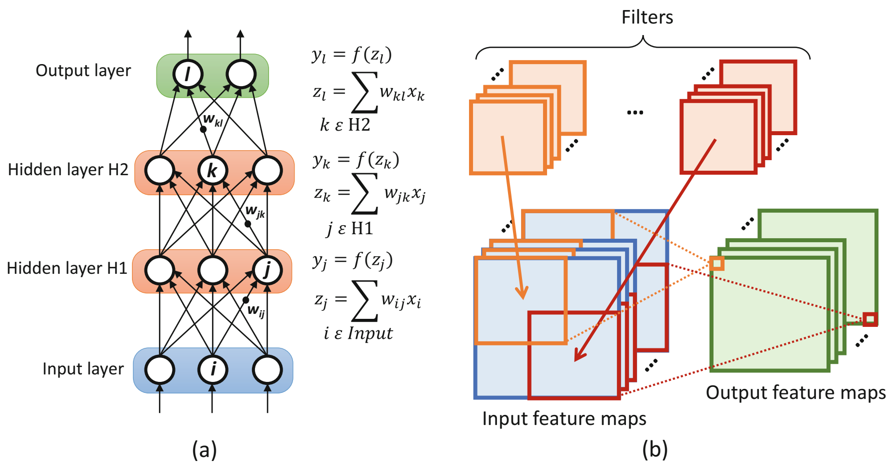


Fig. 4 Illustration of (**a**) a multi-layer perceptron and (**b**) a convolutional layer

In neural networks, neurons are arranged in the form of *layers*. There are several types of NNs, for instance, Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Multi-Layer Perceptrons (MLPs) [31]. Although the techniques discussed in the following sections are not limited to a specific type of NNs, in this chapter, we mainly focus on feed-forward neural networks (i.e., CNNs and MLPs) because of their widespread use in many artificial intelligence applications.

An MLP is a type of NN that is composed of multiple fully-connected layers. In a fully-connected layer, each neuron is connected to all the neurons in the neighboring layers. An example illustration of a three layer MLP is shown in Fig. 4a.

A CNN is a type of NN that is composed of several convolutional layers and the fully-connected layers. An example illustration of a *convolutional layer* is shown in Fig. 4b. The layer is composed of multiple filters which are convolved with the input feature maps to generate the output feature maps. The depth of the filters and the input feature maps is the same. Each filter results in one *output feature map* and, therefore, the number of output feature maps is equal to the number of filters in a convolutional layer. These input and output feature maps are also referred to as *activation maps*. A detailed description of CNNs can be found in [53].

2.2 *Hardware Accelerators for Deep Neural Networks*

To enable the use of DNNs in energy-/power-constraint scenarios as well as in high performance applications, several different hardware architectures for DNN acceleration have been proposed. While all the accelerators provide some unique features and support some specific dataflows in a more efficient manner, *systolic array-based designs are considered among the promising ones* [18, 23, 37, 61].

A systolic array is a homogeneous network of processing elements (PEs), which are tightly coupled together. Each PE in the network receives data from its nearest neighbors, performs some function, and passes on the result and data to the neighboring PE/s. The systolic array-based architectures alleviate the memory bottleneck issue by *locally reusing the data*, without the need of expensive memory read and write operations. Moreover, the systolic arrays are intrinsically efficient at performing matrix multiplications, which is the core operation of neural networks. Therefore, many accelerators use these arrays at their core for accelerating the neural networks [18, 23, 37, 61]. The *Tensor Processing Unit (TPU)*, a DNN accelerator that is currently in use in the datacenters of Google, is a systolic array-based architecture that uses an array of 256×256 multiply-and-accumulate (MAC) units. The TPU provides $15 \times -30\times$ faster execution, and $30 \times -80\times$ more efficient (in terms of performance/Watt) performance than the K80 GPU and the Haswell CPU [23].

Figure 5 illustrates a design overview of an exemplar DNN accelerator which is based on the TPU architecture. The design is used as the basic architecture in the following section. The architecture is composed of a systolic array of MAC units, similar to that in the TPU. Prior to the computations, the weights are pre-loaded in the PEs from the weight memory in a manner that the weights from the same filter/neuron are loaded in the same column of the array. During processing, the

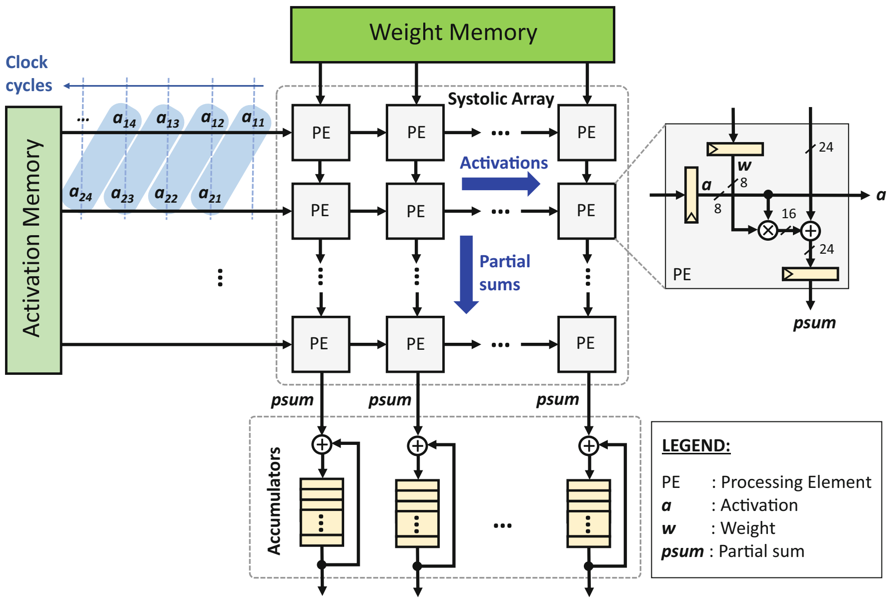


Fig. 5 A systolic array-based DNN accelerator architecture (adapted from [65])

weights are held *stationary* inside the PEs and the activations are streamed in from the activation memory. At each clock cycle, the activations are passed-on from left to right while the partial sums are moved downstream. The activations across rows are aligned such that the activations corresponding to a particular output reaches a particular PE at the same instance when its partial sum reaches that PE. In case, the size of a filter/neuron is larger than the number of rows in the array, each output computation related to the filter/neuron is divided into multiple portions and the accumulators at the bottom are used for temporarily holding the partial sums while rest of the corresponding partial sums are computed by the array. A more detailed explanation of the architecture can be found in [65].

3 Reliable Deep Learning

In this section, we present our methodology for building reliable hardware for DNN-based applications. We also highlight a few case studies, targeting different types of reliability threats, for building reliable yet efficient hardware for DNN-based applications.

3.1 Our Methodology for Designing Reliable DNN Systems

Figure 6 presents our design flow for developing reliable hardware for DNN-based applications [17]. The methodology is composed of two parts: (1) Design-time steps; and (2) Run-time steps.

The **design-time** steps focus on proposing a hardware architecture which is capable of mitigating different types of reliability faults that arise due to process variations and aging, as well as aggressive voltage scaling (i.e., permanent faults

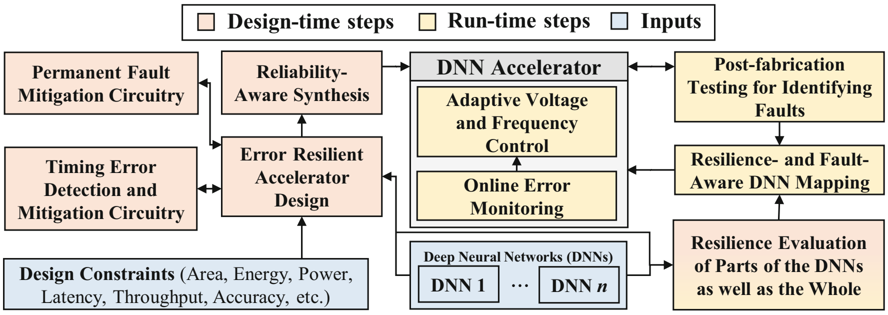

Design-time steps
Run-time steps
Inputs
Permanent Fault Mitigation Circuitry
Reliability-Aware Synthesis
DNN Accelerator
Adaptive Voltage and Frequency Control
Online Error Monitoring
Post-fabrication Testing for Identifying Faults
Timing Error Detection and Mitigation Circuitry
Error Resilient Accelerator Design
Resilience- and Fault-Aware DNN Mapping
Design Constraints (Area, Energy, Power, Latency, Throughput, Accuracy, etc.)
Deep Neural Networks (DNNs)
DNN 1
DNN n
Resilience Evaluation of Parts of the DNNs as well as the Whole

Fig. 6 Our methodology for designing reliable hardware for DNN-based applications (adapted from [17])

and timing errors). Provided a set of design constraints, representative DNN models, and resilience of the DNNs to different types of reliability threats and errors, a baseline hardware architecture is designed. We then reinforce it with different architectural enhancements for mitigating permanent faults (see Sect. 3.3) and handling timing errors (see Sect. 3.4). *The architectural enhancements are performed in a manner that they do not significantly affect the resource efficiency of the baseline architecture.* Once the architecture is finalized, the hardware is synthesized using reliability-aware synthesis techniques, for example, by using standard cells to selectively harden vulnerable nodes in the hardware [33], to harden the more vulnerable parts of the hardware design.

The **run-time** steps focus on proposing *mapping policies* for mapping DNN computations to the synthesized hardware. The mapping policies are decided based on the fault maps generated using post-fabrication and testing, and the error resilience of the DNNs. *Techniques like error injection can be used for the resilience analysis* [16, 46]. *Fault-aware training of DNNs* can also be used for designing/modifying network architecture/parameters (see Sect. 3.3). Moreover, adaptive voltage scaling can be employed for trading off reliability with energy efficiency based on the error resilience of the DNNs. If required, software-level redundancy can also be employed to further improve the reliability by performing the computations related to critical neurons/filters multiple times.

3.2 Resilience of DNNs to Reliability Threats

Neural Networks are assumed to be inherently error resilient [12]. However, different types of errors can have different impact on the output of a DNN. This section presents the accuracy analysis of DNNs in the presence of different types of reliability faults.

3.2.1 Resilience of DNNs to Permanent Faults

This section highlights the resilience of DNNs to permanent faults by empirically analyzing the effects of *stuck-at* permanent faults in the TPU-based accelerator (presented in Fig. 5) on the classification accuracy of different DNNs. The datasets (i.e., *MNIST* and *TIMIT*) and the corresponding network architectures used for this analysis are listed in Table 1. To study the resilience, the TPU with a systolic array of 256×256 MAC units is synthesized using 45 nm OSU PDK to generate a gate-level netlist and then stuck-at faults are inserted at internal nodes in the netlist. For this analysis, faults only in the data-path were considered as the faults in the memory components can be mitigated using Error Correction Codes (ECC) and faults in control-path can lead to undesirable results.

Figure 7a shows the impact of using a faulty TPU for two different classification tasks, i.e., *image classification using the MNIST dataset* and *speech recognition*

Table 1 Datasets and the corresponding 8-bit DNNs used for evaluation (adapted from [65])

Dataset	Network architecture	Accuracy(%)
MNIST [30]	Fully-connected (L1–L4): 784×256×256×256×10	98.15
TIMIT [4]	Fully-connected (L1–L4): 1845×2000×2000×2000×183	73.91
ImageNet [7]	Convolutional (L1–L2): (224, 224, 3)×(27, 27, 64)×(13, 13, 192) Convolutional (L3–L5): (13, 13, 384)×(13, 13, 256)×(6, 6, 256) Fully-connected (L6–L8): 4096×4096×1000	76.33 (Top-5)

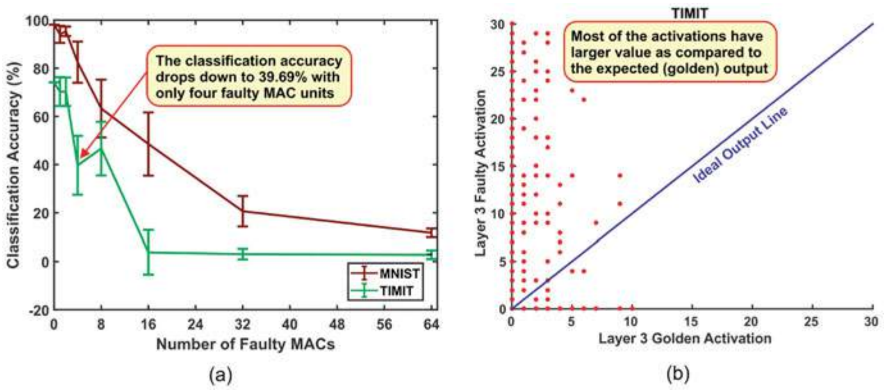


Fig. 7 Impact of stuck-at-faults in the baseline TPU-based architecture on DNN applications. (**a**) Classification accuracy drop due to stuck-at-fault MACs. (**b**) Impact of TPU stuck-at-faults on DNN applications (adapted from [66])

using the TIMIT dataset. It can be seen in the figure that the classification accuracy of both the tasks decreases significantly with the increase in the number of faulty PEs in the hardware. For example, the classification accuracy for the *TIMIT* dataset drops from 74.13 to 39.69% when only four (out of 256 × 256) MAC units are faulty and is almost 0% when the number of faulty MACs increases to 16 or more.

The reason for the significant drop in accuracy can be understood by comparing the golden (fault-free) output of the neurons of a particular layer with the outputs computed by the faulty TPU. Figure 7b shows that the computed output of the final layer of the network used for the *TIMIT* dataset in most of the cases has higher activation value as compared to the expected. This is mainly because of the fact that *stuck-at faults, in some of the cases, affect the higher order bits of the MACs output*. This highlights the need for permanent fault mitigation in the hardware to increase the yield as hardware with permanent faults cannot be used for ML-based applications, specifically for the safety-critical applications.

3.2.2 Resilience of DNNs to Timing Faults

Timing failures in high performance nanometer technology-based digital circuits are a major reliability concern and are caused by various mechanisms, e.g., power supply disturbance, crosstalk, process variations, as well as aging. Moreover, the operating conditions, which play a vital role in defining the performance and energy efficiency of the hardware, also have a significant impact on the frequency of the timing errors. Although it is assumed that the critical paths, which are more vulnerable to timing errors, are rarely exercised, the timing errors can significantly affect the functionality of an application. Here, we highlight this for DNN-based applications by analyzing the energy-quality trade-off achieved using voltage underscaling. We show the analysis for two widely accepted types of timing error mitigation techniques: (1) *timing error detection and recovery* (TED) [9]; and (2) *timing error propagation* (TEP) [41, 62]. The TED makes use of additional components (e.g., using Razor flip-flops [9]) for detecting timing errors, and recovers by reliably re-executing the function in case of errors. On the other hand, TEP allows errors to propagate through to the application layer in the hope that the application is error resilient.

For this analysis, the TPU-based hardware architecture discussed in Sect. 2.2 is considered. The architecture is assumed to be composed of a 256×256 MAC array. The terms *Local Timing Error* and *Global Timing Error* are used to characterize the resilience. The local timing error is used to denote the error in a single MAC unit. The global timing error defines the error in the complete systolic array. Figure 8b shows the impact on the classification accuracy for the *MNIST* dataset with voltage underscaling when the timing errors are allowed to propagate through to the application layer. It can be seen from the figure that as soon as the timing errors start occurring, i.e., below the voltage underscaling ratio of $r = 0.9$ (as shown in Fig. 8b), the classification accuracy of the DNN for TEP drops sharply.

As mentioned above, the TED-based approaches work on the principle of error detection and recovery. The recovery phase in TED defines its limitation for huge systolic array-based systems as, for synchronization of the data flow, the complete systolic array has to be stalled to recover the error in a single PE. This limitation of the TED-based approach can be highlighted using Fig. 8a which shows the impact of voltage underscaling on the overall energy consumption of the TPU-based hardware architecture for generating accurate outputs. It can be noted from the figure that *the overall energy consumption for a recovery based technique starts increasing as soon as errors start appearing*, which is the case for even the most naive type of error recovery mechanism, i.e., single cycle recovery.

3.2.3 Resilience of DNNs to Memory Faults

To illustrate the importance of memory faults, we presented an analysis in [17] where we injected random faults at bit-level in the weight memory (i.e., the memory storing the network parameters) and studied the impact of those faults on the

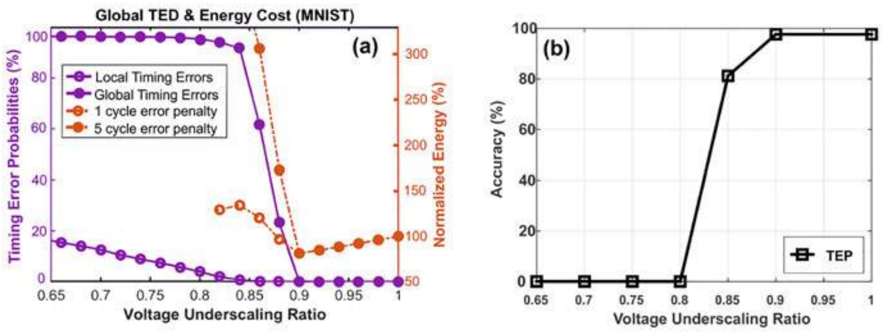


Fig. 8 (**a**) Timing error probabilities versus voltage underscaling ratio, and the corresponding energy cost for global TED. (**b**) DNN accuracy on the MNIST versus voltage underscaling for TEP. (Adapted from [65])

accuracy of a DNN. The analysis concluded that, for the higher significance bits of the weights, the accuracy of the DNNs drop sharply with the increase in error rate. We also studied the impact of different types of bit-flips, i.e., from 0 to 1 bit-flips and from 1 to 0 bit-flips, and found that the 0 to 1 bit-flips result in erroneous output while the 1 to 0 bit-flips do not impact the accuracy much. This is inline with the concept of dropout [20] and dropconnect [59] in the sense that in case of 1 to 0 bit-flips the erroneous output is leaned towards 0 value, whereas in case of 0 to 1 bit-flips the error can increase significantly if the bit-flip occurs in any of the higher significance bits. This analysis was performed on the AlexNet network using the ImageNet dataset. Similar, fault injection methods, e.g., [16] and [46], can also be used for analyzing the resilience of DNNs, as a whole as well as of individual layers/neurons of the networks.

3.3 Permanent Fault Mitigation

To mitigate permanent faults in the computing units of the hardware, two different methods have been proposed: (1) Fault-Aware Pruning (FAP); and (2) Fault-Aware Pruning + Training (FAP+T).

The **Fault-Aware Pruning (FAP)** works on the principle of pruning the weights (i.e., setting them to zero) that have to be mapped on faulty MAC units. *The principle is inline with the concepts of dropout [20] and dropconnect [59] which are commonly used for regularization and avoiding over-fitting*. For this work, the TPU architecture shown in Fig. 5 with static mapping policy is assumed. The static mapping policy means that each weight is mapped to a specific PE while multiple weights can be mapped to the same PE at different time instances. Moreover, it is also assumed that post-fabrication tests are performed on each TPU chip to extract the fault map which indicates the faulty PEs.

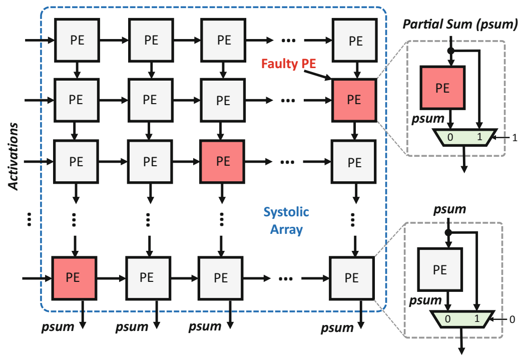


Fig. 9 Systolic array-based architecture for permanent fault mitigation (adapted from [66])

Figure 9 shows an implementation that can be used to realize the concept where a bypass path is provided for each MAC unit [66]. *The bypass path enables to skip the contribution of a specific partial sum in case the specific PE is faulty, which is equivalent to setting the weight to zero.* The area overhead of the modified design is only around 9% [66].

The **Fault-Aware Pruning + Training (FAP+T)** technique starts with the FAP approach, however, it additionally retrains the unpruned weights while forcing the pruned weights to zero to optimize the network parameters. One drawback of this approach is that the fault map of each chip can be different which means that a network has to be retained for each chip based on its own fault map.

Figure 10 shows the impact on the classification accuracy versus the percentage of faulty MAC units for three different classification problems mentioned in Table 1. The results show that both the techniques show significant resilience to the permanent faults. Moreover, the FAP+T technique outperforms FAP because of the involved optimization of the network parameters and allows the DNN-based system to run with negligible accuracy loss even when 50% of its MAC units are faulty. However, in cases where FAP+T is impractical FAP can also provide reasonable accuracy, specifically in cases where the number of faulty units is less.

3.4 Timing Fault Mitigation

As mentioned in Sect. 3.2.2, the conventional TED approaches have significant overheads when used for DNN accelerators. Here, we discuss the new architectural innovations proposed in Thundervolt [65] for mitigating timing errors in DNN accelerators in a performance efficient manner.

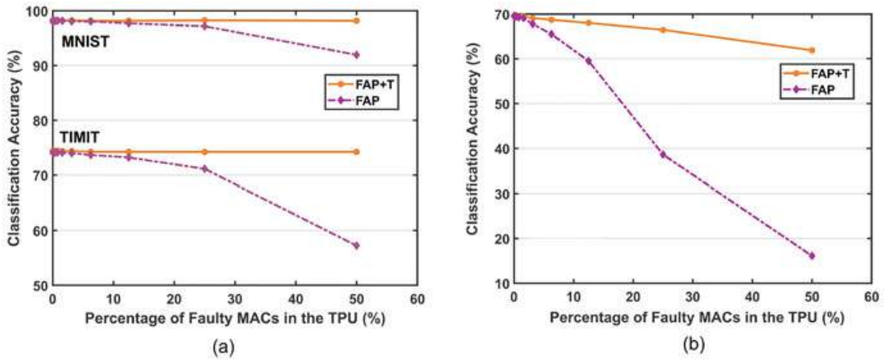


Fig. 10 Classification accuracy versus percentage of faulty MACs using FAP and FAP+T for the networks used corresponding to (**a**) MNIST and TIMIT; and (**b**) ImageNet datasets (adapted from [66])

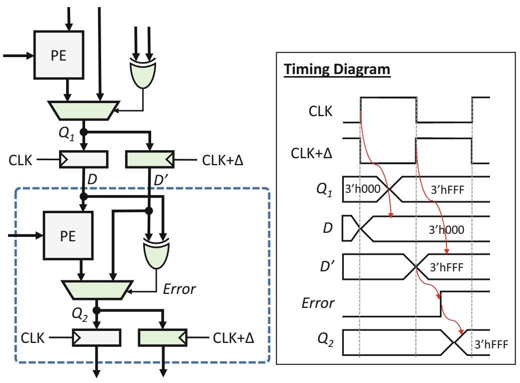


Fig. 11 A block-level diagram illustrating the architectural modifications for TE-Drop and the impact of timing errors on the computation of a neuron (adapted from [65])

3.4.1 TE-Drop

Thundervolt [65] proposed a novel technique to deal with timing errors in a systolic array-based DNN accelerator, i.e., TE-Drop. *TE-Drop utilizes the Razor flip-flops to detect timing errors, however, it does not re-execute erroneous MAC operations.* Similar to the FAP techniques, TE-Drop also works on the principle that the contribution of each individual MAC output to the output of a neuron in DNNs is small. Hence, a few MAC operations can be ignored without significantly affecting the overall accuracy of the network. In case of a timing error, TE-Drop allows the MAC unit to sample the correctly computed output to an alternate register operating on a delayed clock. The succeeding PE is then bypassed and the correctly computed output is provided instead. The architectural modifications required to realize the concept are shown in Fig. 11.

Figure 11 illustrates the functionality of the TE-Drop with the help of a timing diagram. Here, it is assumed that the shadow clock is delayed by 50% of the clock

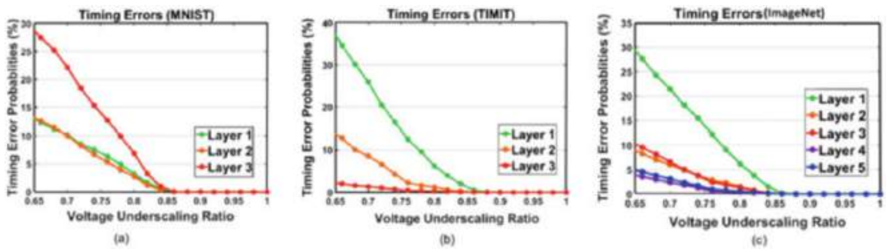


Fig. 12 Timing error probabilities for each layer of the networks used corresponding to (**a**) MNIST. (**b**) TIMIT, and (**c**) ImageNet datasets (adapted from [65])

period. It is assumed that the clock frequency is defined such that the error signal and correct partial sum from the erroneous MAC become available after this much duration. Note that the error signal is obtained by OR-ing the bitwise XOR of all the individual Razor flip-flop at the output of the MAC unit.

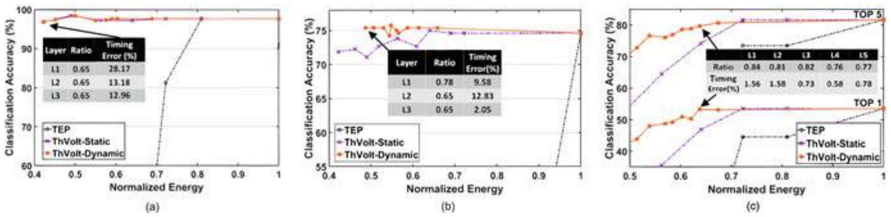
3.4.2 Per-Layer Voltage Underscaling

In most of the accelerators, it is assumed that the layers of a DNN are executed in a serial fashion (i.e., one after the other), where processing of each layer can take thousands of clock cycles, depending on the size of the layer. Figure 12 shows the timing error rate versus voltage underscaling ratio plots for each individual layer of three DNN architectures mentioned in Table 1. It can be seen from the figures that the error rate varies significantly across layers. Based on this observation, a per-layer voltage underscaling scheme was proposed in Thundervolt [65] that distributes the total timing error budget equally among the layers of a network to ensure that the more sensitive layers should not consume a significant part of the budget and limits the achievable efficiency gains.

Figure 13 compares two versions of Thundervolt:

1. **ThVolt-Static** where each voltage underscaling ratio is kept the same throughout a DNN execution.
2. **ThVolt-Dynamic** that utilizes per-layer voltage underscaling based on the sensitivity of each layer.

For the baseline, the results of the TEP scheme are also shown. The plot for ThVolt-Static is obtained by sweeping voltage underscaling ratios, and that of ThVolt-Dynamic is obtained by sweeping the total timing error budget. The figures show that for each case Thundervolt outperforms TEP scheme, and for complex tasks (e.g., image classification on the ImageNet dataset) the **ThVolt-Dynamic** outperforms the **ThVolt-Static** approach.



Layer	Ratio	Timing Error (%)
L1	0.65	28.17
L2	0.65	13.18
L3	0.65	12.96

Layer	Ratio	Timing Error(%)
L1	0.78	9.58
L2	0.65	12.83
L3	0.65	2.05

	L1	L2	L3	L4	L5
Ratio	0.84	0.81	0.82	0.76	0.77
Timing Error(%)	1.56	1.58	0.73	0.58	0.78

Fig. 13 Accuracy versus energy trade-off using Thundervolt [65] on validation data. (**a**) MNIST. (**b**) TIMIT (**c**) ImageNet (adapted from [65])

4 Secure Deep Learning

In this section, we present different security attacks on DNNs and potential countermeasures.

4.1 Security Attacks on DNNs

Several security attacks have been proposed by exploiting the security vulnerabilities, especially data dependency and unpredicted behavior of intermediate layers of DNN-algorithms during training as well as inference. However, *adversarial and backdooring attacks are some of the most effective and popular attacks for DNNs*. Therefore, in the following subsections, we analyze the state-of-the-art adversarial attacks and proposed backdoor attacks.

4.1.1 Adversarial Perturbation Attacks

It can be defined as the crafted imperceptible noise to perform *targeted* or *untargeted misclassification* in a DNN-based system. In these attacks, an attacker's objective can be summarized as follows: given an image x with a classification label $y = \text{classifier}(x)$, where classifier is the function of the neural network. The attacker aims to find an image x' whose classification label is y', such that $y' = \text{classifier}(x') \neq y$, and $\|x' - x\| \leq \delta$, where δ is an upper bound of the distortion from x to x'. For example, some input adversarial attacks are shown in Fig. 14.

Several attacks have proposed to exploit the adversarial vulnerabilities in DNN-based systems. However, based on the attack methodology, these attacks can broadly be categorized into Gradient Sign Methods and Optimization-based approaches.

1. **Gradient Sign Methods:** These attacks exploit the derivatives and backpropagation algorithm to generate the attack images with imperceptible crafted noise. The main goal of these attacks is to minimize the prediction probability of the true label so as to mislead the network to output a different label (can be targeted or untargeted) other than the ground truth. Some of the most commonly proposed

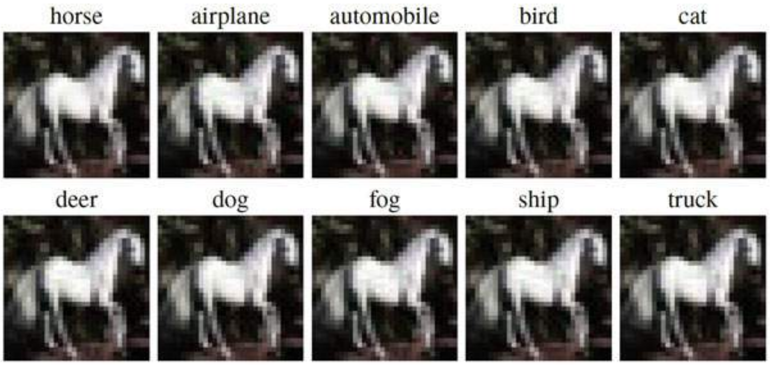


Fig. 14 Clean and adversarial images with different prediction labels, where the clean image of a horse and its adversarial images remain extremely similar, however, their prediction labels are quite distinct and each targets a totally different class

attacks are Fast Gradient Sign (FGS), Iterative Fast Gradient Sign (IFGS), and Jacobian-based saliency map attack (JSMA) methods [42]. Based on the similar principle, there are following attacks which do not require training data and also have less convergence time (in terms of queries):

- *TrISec:* This attack exploits the backpropagation algorithm to identify the small change (attack noise) in input pixels with respect to misclassification at the output, while ensuring the imperceptibility [25].
- *RED-Attack:* Most of the state-of-the-art attacks require a large number of queries to generate an imperceptible attack. However, in resource-constraint scenarios, these attacks may fail, therefore, we proposed a methodology that generates an attack image with imperceptible noise while requiring a very less number of queries [26].

2. **Optimization-based Approaches:** Unlike the gradient-based approaches, these attacks redefine the loss function (i.e., the cost function used for optimization) by adding extra constraints with respect to targeted or untargeted misclassification, and then propose different optimization algorithms to generate adversarial images. For example, Limited Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) [54] and Carlini and Wagner (CW) [5] attacks use the box-constrained L-BFGS algorithm with single and multi-objective optimization, respectively.

Other types of neural networks, i.e., Capsule Networks and Spiking Neural Networks, are emerging as an alternative because of their robustness to affine transformations and potential for offering higher energy efficiency, respectively. However, recent works showed that these networks are also vulnerable to adversarial attacks [38, 39].

4.1.2 Backdoor Attacks

Due to expensive and computationally intensive training, the training (or fine-tuning) of DNNs is usually outsourced which opens the new frontiers of security threats, e.g., *backdoored neural networks (BadNets [14])*. These threats arise due to the involvement of untrusted third party service providers that can insert backdoors by training the ML models on compromised training data, or by altering the DNN structure. The untrusted third party also ensures the required accuracy of the backdoored model on most validation and testing inputs, but cause targeted misclassification or confidence reduction based on *backdoor trigger*. For example, in case of autonomous driving use case, an attacker can introduce the backdoor in a street sign detector while ensuring the required accuracy for classifying street signs in most of the cases, however, it can perform either targeted or untargeted misclassification, i.e., classifies stop signs with a particular sticker as speed limit signs or any other sign different from stop sign. This kind of misclassification can lead to catastrophic effects, e.g., in case of misclassification of a stop sign, autonomous vehicle does not stop at the intersection which can result in an accident.

4.2 Defences Against Security Attacks on DNNs

Several countermeasures have been proposed to defend against the adversarial attacks, i.e., *DNN masking*, *gradient masking*, *training for known adversarial attacks*, and *pre-processing of the CNN inputs* [6]. For examples, Fig. 15 shows

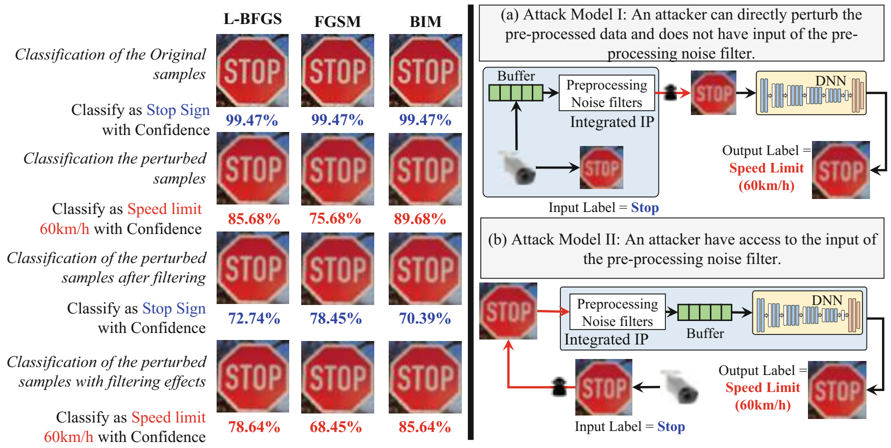

L-BFGS
FGSM
BIM
Classification of the Original samples
Classify as Stop Sign with Confidence
99.47%
99.47%
99.47%
Classification the perturbed samples
Classify as Speed limit 60km/h with Confidence
85.68%
75.68%
89.68%
Classification of the perturbed samples after filtering
Classify as Stop Sign with Confidence
72.74%
78.45%
70.39%
Classification of the perturbed samples with filtering effects
Classify as Speed limit 60km/h with Confidence
78.64%
68.45%
85.64%
(a) Attack Model I: An attacker can directly perturb the pre-processed data and does not have input of the pre-processing noise filter.
Buffer
Preprocessing Noise filters
Integrated IP
DNN
Output Label = Speed Limit (60km/h)
Input Label = Stop
(b) Attack Model II: An attacker have access to the input of the pre-processing noise filter.
Preprocessing Noise filters
Buffer
Integrated IP
DNN
Output Label = Speed Limit (60km/h)
Input Label = Stop

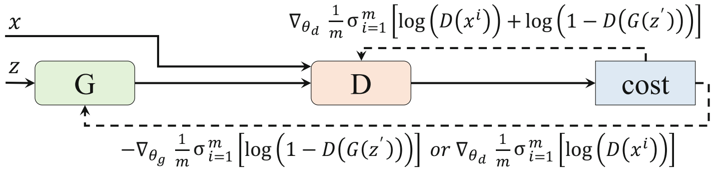
Fig. 15 Impact of the pre-processing filtering on the state-of-the-art adversarial attacks with different attack models with and without the access of filters. (**a**) Attack model I: an attacker can directly perturb the pre-processed data and does not have input of the pre-processing noise filter. (**b**) Attack model II: an attacker have access to the input of the pre-processing noise filter (adapted from [24, 27])

that *low-pass pre-processing filters that can nullify the adversarial attacks if they are not known to the attacker* [24, 27]. Therefore, based on this analysis, we have proposed to utilize the pre-processing quantization to improve the perceptibility of the attack noise [2]. Similarly, Sobel-filers can also be used to decrease the attack strength [55].

However, these defences are not applicable to backdoor-based attacks because the backdoor attacks intrude the networks and are activated through a specific trigger. Therefore, to address these attacks, we propose to use *pruning* as a natural defense because it eliminates the neurons that are dormant on clean inputs, consequently disabling backdoor behavior [14]. Although these defenses are effective, most of them provide defense against known adversarial and backdoor attacks. Therefore, one of the most important problems in designing secure machine learning systems is *the ability to define threats, and model them sufficiently so that any learning system can be trained to be able to identify such threats*.

4.2.1 Generative Adversarial Networks

To address the above-mentioned challenge, Generative Adversarial Networks (GANs) have emerged as one of the prime solutions because of their ability to generate the model by learning to mimic actual models [13]. In particular, GANs is a framework to estimate generative models where simultaneously two models are trained, *generator* (G) and *discriminator* (D) (see Fig. 16). This is achieved through an adversarial process where the two models are competing with each other for achieving two opposite goals. Simply speaking, D is trying to distinguish real images from fake images and G is trying to create images as close as possible to real images so as D will not be able to distinguish them, as illustrated in Fig. 16. When dealing with inference scenarios, the challenge is to provide a training set which includes attack-generated data patterns labeled of course correctly as attacks. For example, an autonomous system may rely on visual information to orient and steer itself or to undertake significant decisions. However, white or patterned noise can be maliciously inserted into a camera feed that may fool the system, and thus results in potentially catastrophic scenarios. The problem with modeling these types of attacks is that the attack models are hard to mathematically formulate, and thus



x
z
G
D
cost
$\nabla_{\theta_d} \frac{1}{m} \sigma_{i=1}^{m} \left[\log\left(D(x^i)\right) + \log\left(1 - D(G(z'))\right)\right]$
$-\nabla_{\theta_g} \frac{1}{m} \sigma_{i=1}^{m} \left[\log\left(1 - D(G(z'))\right)\right]$ or $\nabla_{\theta_d} \frac{1}{m} \sigma_{i=1}^{m} \left[\log\left(D(x^i)\right)\right]$

Fig. 16 GANs framework: an illustration of how G and D are trained, adapted from [21]

hard, if not impossible, to replicate and therefore, train the system to recognize them as attacks. Hence, GANs provide us with this capability, as we can utilize the G model to generate-and-evaluate threat models and train the D model to differentiate between what we consider an attack or not. However, GAN-based threat comes with the following challenges [21, 40]:

1. **Collapsing:** In this case G produces only a single sample or set of similar samples, regardless the type of input given to it.
2. **Convergence:** Since G and D models are competing towards achieving two opposite goals, this may make the model parameters to oscillate, destabilizing the training process.
3. **Gradient Vanish:** If one of the two models becomes more powerful than the other, the learning signal is becoming useless, making the system incapable to learn.
4. **Over-Fitting:** This is typically due to the unbalance optimization of G and the D models, e.g., if too much time is spent on minimizing G, then D will most likely collapse to a few states.
5. **Sensitive:** It is characterized by being highly sensitive to the selection of the hyperparameters, i.e., learning rate, momentum, etc.; making the training process much more tedious.

4.2.2 Case Study: Noisy Visual Data, and How GANs Can be Used to Remove Noise and Provide Robustness

To illustrate how a GAN-based framework can be used to define threats and subsequently to provide robustness in a DNN-based system, we use computer vision as an example because security threats in computer vision applications may arise from either physical attacks, cyber attacks or a combination of both. We use the hazing in images to model such threats. To remove this threat, we use the GANs because of their capability in preserving fine details in images and producing results that look perceptually convincing [28]. The goal is to translate the input image with haze, into a haze-free output image. In this case, the noise distribution z is the noisy image, and it is given as input to the GANs, i.e., haze input image. Afterwards, a new sample image F is generated by G. D will receive as input the generated image F and the ground truth haze-free image, to be trained to distinguish between real and artificially generated images (see Fig. 17).

This approach has recently been used for haze removal [8, 52, 63]. These methods mainly differ from each other, based on the utilized deep learning structure for G and D, i.e., using three types of G to solve the optimization of the haze removal problem [63], using the concept of cycle GAN introduced in [67]. Also they may differ for the type of loss function used for the training process, where the overall objective functions is constrained to preserve certain features or priors. However, they provide a solution that, in most of the cases, is capable to improve the quality performances of the state-of-the-art haze removal methods for single image, so as making this quite a promising area.

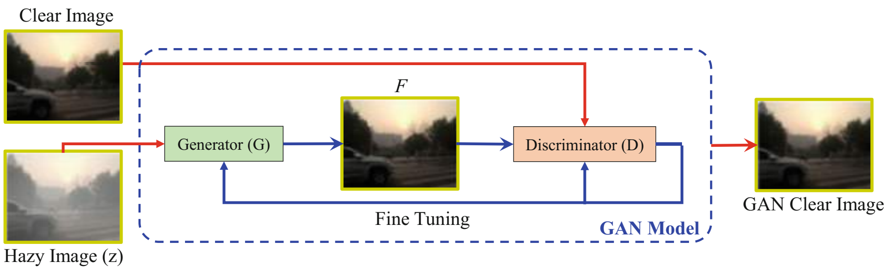


Fig. 17 Example of GANs used for removing haze noise from a single image. The haze image is input to *G* that generate an output image *F* and *D* receives as input both the generated image *F* and the free-haze image. Input images taken from [32]

5 Open Research Challenges

Machine learning has paved its way to a majority of the fields that involve data processing. However, regardless of all the work which has been carried out in interpreting the neural networks and making the ML-based systems reliable, there are still quite some challenges which are to be addressed before ML algorithms (specifically, DNNs) can be widely accepted for complex safety-critical applications. Following is a list of a few of the main challenges in this direction.

- **Error-Resilience Evaluation Frameworks:** One approach towards this for timing error estimation is proposed in [64]. However, more sophisticated frameworks are required to study the impact of multiple types of reliability threats and their interdependence in a time efficient manner.
- **Methodologies for Designing Robust and Resource-Efficient DNNs:** Retraining a DNN in the presence of hardware-induced faults [15] can improve their resilience. However, there is a need to investigate the types of DNN architectures which are inherently resilient to most (if not all) of the reliability threats. Furthermore, there is a need to investigate frameworks to develop robust ML systems by synergistically investigating reliability and security vulnerabilities.
- **Reliable and Resource-Efficient Hardware Architectures:** With all the security and reliability challenges highlighted in the chapter, there is a dire need to re-think the way current DNN hardware is designed, such that the vulnerabilities that cannot be addressed at the software-level have to be addressed through a robust DNN hardware.
- **Interpretability of Deep Neural Networks:** Developing interpretable DNNs is a challenge, however, it has to be addressed in order to better understand the functionality of the DNNs. This will help us in improving the learning capabilities of the DNNs, as well as in uncovering their true vulnerabilities and thereby will help is developing more efficient and robust network architectures.
- **Practicality of the Attacks:** With the ongoing pace of the research in ML, new methods and types of network architectures are surfacing, e.g., CapsuleNets.

Also, the focus of the community is shifting more towards semi-/un-supervised learning methods as they overcome the need for large labeled datasets. Therefore, there is a dire need to align the focus with the current trends in the ML community. Also, the attacks should be designed considering the constraints of the real systems, i.e., without making unrealistic assumptions about the number of queries and the energy/power resources available to generate an attack. An early work in this direction by our group can be found at [26].

Acknowledgments This work was supported in parts by the German Research Foundation (DFG) as part of the priority program "Dependable Embedded Systems" (SPP 1500—spp1500.itec.kit.edu) and in parts by the National Science Foundation under Grant 1801495.

References

1. Ahmad, H., Tanvir, M., Abdullah, M., Javed, M.U., Hafiz, R., Shafique, M.: Systimator: a design space exploration methodology for systolic array based CNNs acceleration on the FPGA-based edge nodes (2018). arXiv:1901.04986
2. Ali, H., Tariq, H., Hanif, M.A., Khalid, F., Rehman, S., Ahmed, R., Shafique, M.: QuSecNets: quantization-based defense mechanism for securing deep neural network against adversarial attacks (2018). arXiv:1811.01437
3. Athalye, A., Carlini, N., Wagner, D.: Obfuscated gradients give a false sense of security: circumventing defenses to adversarial examples (2018). arXiv:1802.00420
4. Ba, J., Caruana, R.: Do deep nets really need to be deep? In: Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N.D., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems, vol. 27, pp. 2654–2662. Curran Associates, New York (2014). http://papers.nips.cc/paper/5484-do-deep-nets-really-need-to-be-deep.pdf
5. Carlini, N., Wagner, D.: Towards evaluating the robustness of neural networks (2016). arXiv:1608.04644
6. Chakraborty, A., Alam, M., Dey, V., Chattopadhyay, A., Mukhopadhyay, D.: Adversarial attacks and defences: a survey (2018). arXiv:1810.00069
7. Deng, J., Dong, W., Socher, R., Li, L.: ImageNet: a large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition, pp. 248–255 (2009). https://doi.org/10.1109/CVPR.2009.5206848
8. Engin, D., Genç, A., Ekenel, H.K.: Cycle-Dehaze: enhanced cycleGAN for single image dehazing. In: 2018 IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2018, Salt Lake City, June 18–22, 2018, pp. 825–833 (2018). https://doi.org/10.1109/CVPRW.2018.00127. http://openaccess.thecvf.com/content_cvpr_2018_workshops/w13/html/Engin_Cycle-Dehaze_Enhanced_CycleGAN_CVPR_2018_paper.html
9. Ernst, D., Das, S., Lee, S., Blaauw, D., Austin, T., Mudge, T., Kim, N.S., Flautner, K.: Razor: circuit-level correction of timing errors for low-power operation. IEEE Micro **24**(6), 10–20 (2004)
10. Esteva, A., Robicquet, A., Ramsundar, B., Kuleshov, V., DePristo, M., Chou, K., Cui, C., Corrado, G., Thrun, S., Dean, J.: A guide to deep learning in healthcare. Nat. Med. **25**(1), 24 (2019)
11. Fink, M., Liu, Y., Engstle, A., Schneider, S.A.: Deep learning-based multi-scale multi-object detection and classification for autonomous driving. In: Fahrerassistenzsysteme 2018, pp. 233–242. Springer, Berlin (2019)

12. Gebregiorgis, A., Kiamehr, S., Tahoori, M.B.: Error propagation aware timing relaxation for approximate near threshold computing. In: Proceedings of the 54th Annual Design Automation Conference 2017, p. 77. ACM, New York (2017)
13. Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2, NIPS'14, pp. 2672–2680. MIT Press, Cambridge (2014). http://dl.acm.org/citation.cfm?id=2969033.2969125
14. Gu, T., Dolan-Gavitt, B., Garg, S.: BadNets: identifying vulnerabilities in the machine learning model supply chain (2017). arXiv:1708.06733
15. Hacene, G.B., Leduc-Primeau, F., Soussia, A.B., Gripon, V., Gagnon, F.: Training modern deep neural networks for memory-fault robustness. In: 2019 IEEE International Symposium on Circuits and Systems (ISCAS), pp. 1–5. IEEE, Piscataway (2019)
16. Hanif, M.A., Hafiz, R., Shafique, M.: Error resilience analysis for systematically employing approximate computing in convolutional neural networks. In: 2018 Design, Automation and Test in Europe Conference and Exhibition (DATE), pp. 913–916. IEEE, Piscataway (2018)
17. Hanif, M.A., Khalid, F., Putra, R.V.W., Rehman, S., Shafique, M.: Robust machine learning systems: reliability and security for deep neural networks. In: 2018 IEEE 24th International Symposium on On-Line Testing and Robust System Design (IOLTS), pp. 257–260. IEEE, Piscataway (2018)
18. Hanif, M.A., Putra, R.V.W., Tanvir, M., Hafiz, R., Rehman, S., Shafique, M.: MPNA: a massively-parallel neural array accelerator with dataflow optimization for convolutional neural networks (2018). arXiv:1810.12910
19. Henkel, J., Bauer, L., Dutt, N., Gupta, P., Nassif, S., Shafique, M., Tahoori, M., Wehn, N.: Reliable on-chip systems in the nano-era: lessons learnt and future trends. In: Proceedings of the 50th Annual Design Automation Conference, p. 99. ACM, New York (2013)
20. Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.R.: Improving neural networks by preventing co-adaptation of feature detectors (2012). arXiv:1207.0580
21. Hui, J.: Gan why it is so hard to train generative adversarial networks! Elsevier, Amsterdam (2018). https://medium.com/@jonathan_hui/gan-why-it-is-so-hard-to-train-generative-advisory-networks-819a86b3750b
22. Jia, J., Gong, N.Z.: Attriguard: a practical defense against attribute inference attacks via adversarial machine learning. In: 27th {USENIX} Security Symposium ({USENIX} Security 18), pp. 513–529 (2018)
23. Jouppi, N.P., Young, C., Patil, N., Patterson, D., Agrawal, G., Bajwa, R., Bates, S., Bhatia, S., Boden, N., Borchers, A., et al.: In-datacenter performance analysis of a tensor processing unit. In: 2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA), pp. 1–12. IEEE, Piscataway (2017)
24. Khalid, F., Hanif, M.A., Rehman, S., Qadir, J., Shafique, M.: Fademl: understanding the impact of pre-processing noise filtering on adversarial machine learning (2018). arXiv:1811.01444
25. Khalid, F., Hanif, M.A., Rehman, S., Shafique, M.: ISA4ML: training data-unaware imperceptible security attacks on machine learning modules of autonomous vehicles (2018). arXiv:1811.01031
26. Khalid, F., Ali, H., Hanif, M.A., Rehman, S., Ahmed, R., Shafique, M.: Red-attack: resource efficient decision based attack for machine learning (2019). arXiv:1901.10258
27. Khalid, F., Hanif, M.A., Rehman, S., Qadir, J., Shafique, M.: FAdeML: understanding the impact of pre-processing noise filtering on adversarial machine learning. In: Design, Automation and Test in Europe. IEEE, Piscataway (2019)
28. Kupyn, O., Budzan, V., Mykhailych, M., Mishkin, D., Matas, J.: Deblurgan: blind motion deblurring using conditional adversarial networks. In: Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2018)
29. Kwon, H., Samajdar, A., Krishna, T.: MAERI: enabling flexible dataflow mapping over DNN accelerators via reconfigurable interconnects. In: Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems, pp. 461–475. ACM, New York (2018)

30. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998). https://doi.org/10.1109/5.726791
31. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436 (2015)
32. Li, B., Ren, W., Fu, D., Tao, D., Feng, D., Zeng, W., Wang, Z.: Benchmarking single-image dehazing and beyond. IEEE Trans. Image Process. **28**(1), 492–505 (2019)
33. Limbrick, D.B., Mahatme, N.N., Robinson, W.H., Bhuva, B.L.: Reliability-aware synthesis of combinational logic with minimal performance penalty. IEEE Trans. Nuclear Sci. **60**(4), 2776–2781 (2013)
34. Lu, W., Yan, G., Li, J., Gong, S., Han, Y., Li, X.: FlexFlow: a flexible dataflow accelerator architecture for convolutional neural networks. In: 2017 IEEE International Symposium on High Performance Computer Architecture (HPCA), pp. 553–564. IEEE, Piscataway (2017)
35. Lyons, R.E., Vanderkulk, W.: The use of triple-modular redundancy to improve computer reliability. IBM J. Res. Development **6**(2), 200–209 (1962)
36. Marchisio, A., Shafique, M.: Capstore: energy-efficient design and management of the on-chip memory for CapsuleNet inference accelerators (2019). arXiv:1902.01151
37. Marchisio, A., Hanif, M.A., Shafique, M.: CapsAcc: an efficient hardware accelerator for CapsuleNets with data reuse (2018). arXiv:1811.08932
38. Marchisio, A., Nanfa, G., Khalid, F., Hanif, M.A., Martina, M., Shafique, M.: Capsattacks: robust and imperceptible adversarial attacks on capsule networks (2019). arXiv:1901.09878
39. Marchisio, A., Nanfa, G., Khalid, F., Hanif, M.A., Martina, M., Shafique, M.: SNN under attack: are spiking deep belief networks vulnerable to adversarial examples? (2019). arXiv:1902.01147
40. Metz, L., Poole, B., Pfau, D., Sohl-Dickstein, J.: Unrolled generative adversarial networks. In: Proceedings of the International Conference on Learning Representations (ICLR'17) (2017)
41. Nakhaee, F., Kamal, M., Afzali-Kusha, A., Pedram, M., Fakhraie, S.M. Dorosti, H.: Lifetime improvement by exploiting aggressive voltage scaling during runtime of error-resilient applications. Integr. VLSI J. **61**, 29–38 (2018)
42. Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z.B., Swami, A.: The limitations of deep learning in adversarial settings. In: 2016 IEEE European Symposium on Security and Privacy (EuroS&P), pp. 372–387. IEEE, Piscataway (2016)
43. Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z.B., Swami, A.: Practical black-box attacks against machine learning. In: Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, pp. 506–519. ACM, New York (2017)
44. Putra, R.V.W., Hanif, M.A., Shafique, M.: ROMANet: fine-grained reuse-driven data organization and off-chip memory access management for deep neural network accelerators (2019). arXiv:1902.10222
45. Raghunathan, B., Turakhia, Y., Garg, S., Marculescu, D.: Cherry-picking: exploiting process variations in dark-silicon homogeneous chip multi-processors. In: 2013 Design, Automation and Test in Europe Conference and Exhibition (DATE), pp. 39–44. IEEE, Piscataway (2013)
46. Reagen, B., Gupta, U., Pentecost, L., Whatmough, P., Lee, S.K., Mulholland, N., Brooks, D., Wei, G.Y.: Ares: a framework for quantifying the resilience of deep neural networks. In: 2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC), pp. 1–6. IEEE, Piscataway (2018)
47. Rehman, S., Shafique, M., Henkel, J.: Reliable Software for Unreliable Hardware: A Cross Layer Perspective. Springer, Berlin (2016)
48. Schorn, C., Guntoro, A., Ascheid, G.: Efficient on-line error detection and mitigation for deep neural network accelerators. In: International Conference on Computer Safety, Reliability, and Security, pp. 205–219. Springer, Berlin (2018)
49. Shafique, M., Garg, S., Henkel, J., Marculescu, D.: The EDA challenges in the dark silicon era: temperature, reliability, and variability perspectives. In: Proceedings of the 51st Annual Design Automation Conference, pp. 1–6. ACM, New York (2014)
50. Shokri, R., Stronati, M., Song, C., Shmatikov, V.: Membership inference attacks against machine learning models. In: 2017 IEEE Symposium on Security and Privacy (SP), pp. 3–18. IEEE, Piscataway (2017)

51. Suciu, O., Marginean, R., Kaya, Y., Daume III, H., Dumitras, T.: When does machine learning {FAIL}? Generalized transferability for evasion and poisoning attacks. In: 27th {USENIX} Security Symposium ({USENIX} Security'18), pp. 1299–1316 (2018)
52. Swami, K., Das, S.K.: Candy: conditional adversarial networks based fully end-to-end system for single image haze removal (2018). https://arxiv.org/abs/1801.02892v2
53. Sze, V., Chen, Y.H., Yang, T.J., Emer, J.S.: Efficient processing of deep neural networks: a tutorial and survey. Proc. IEEE **105**(12), 2295–2329 (2017)
54. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R.: Intriguing properties of neural networks (2013). arXiv:1312.6199
55. Tariq, H., Ali, H., Hanif, M.A., Khalid, F., Rehman, S., Ahmed, R., Shafique, M.: SSCNets: a selective sobel convolution-based technique to enhance the robustness of deep neural networks against security attacks (2018). arXiv:1811.01443
56. Tiwari, A., Torrellas, J.: Facelift: hiding and slowing down aging in multicores. In: Proceedings of the 41st Annual IEEE/ACM International Symposium on Microarchitecture, pp. 129–140. IEEE Computer Society, Washington (2008)
57. Tramèr, F., Zhang, F., Juels, A., Reiter, M.K., Ristenpart, T.: Stealing machine learning models via prediction APIs. In: 25th {USENIX} Security Symposium ({USENIX} Security'16), pp. 601–618 (2016)
58. Vadlamani, R., Zhao, J., Burleson, W., Tessier, R.: Multicore soft error rate stabilization using adaptive dual modular redundancy. In: Proceedings of the Conference on Design, Automation and Test in Europe, pp. 27–32. European Design and Automation Association, Leuven (2010)
59. Wan, L., Zeiler, M., Zhang, S., Le Cun, Y., Fergus, R.: Regularization of neural networks using dropconnect. In: International Conference on Machine Learning, pp. 1058–1066 (2013)
60. Wang, B., Gong, N.Z.: Stealing hyperparameters in machine learning. In: 2018 IEEE Symposium on Security and Privacy (SP), pp. 36–52. IEEE, Piscataway (2018)
61. Wei, X., Yu, C.H., Zhang, P., Chen, Y., Wang, Y., Hu, H., Liang, Y., Cong, J.: Automated systolic array architecture synthesis for high throughput CNN inference on FPGAs. In: Proceedings of the 54th Annual Design Automation Conference 2017, p. 29. ACM, New York (2017)
62. Whatmough, P.N., Das, S., Bull, D.M., Darwazeh, I.: Circuit-level timing error tolerance for low-power DSP filters and transforms. IEEE Trans. Very Large Scale Integration Syst. **21**(6), 989–999 (2013)
63. Yang, X., Xu, Z., Luo, J.: Towards perceptual image dehazing by physics-based disentanglement and adversarial training. In: Thirty-Second AAAI Conference on Artificial Intelligence (AAAI) (2018)
64. Zhang, J.J., Garg, S.: Fate: fast and accurate timing error prediction framework for low power DNN accelerator design. In: Proceedings of the International Conference on Computer-Aided Design, p. 24. ACM, New York (2018)
65. Zhang, J., Rangineni, K., Ghodsi, Z., Garg, S.: ThUnderVolt: enabling aggressive voltage underscaling and timing error resilience for energy efficient deep neural network accelerators (2018). arXiv:1802.03806
66. Zhang, J.J., Gu, T., Basu, K., Garg, S.: Analyzing and mitigating the impact of permanent faults on a systolic array based neural network accelerator. In: 2018 IEEE 36th VLSI Test Symposium (VTS), pp. 1–6. IEEE, Piscataway (2018)
67. Zhu, J., Park, T., Isola, P., Efros, A.A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. In: 2017 IEEE International Conference on Computer Vision (ICCV), pp. 2242–2251 (2017). https://doi.org/10.1109/ICCV.2017.244

Exploiting Memory Resilience for Emerging Technologies: An Energy-Aware Resilience Exemplar for STT-RAM Memories

Amir Mahdi Hosseini Monazzah, Amir M. Rahmani, Antonio Miele, and Nikil Dutt

1 Introduction

In the recent years, the aggressive progress in technology scaling has allowed to integrate a larger number of processing cores in the same chip thus leading to the fabrication of multicore and manycore devices. To efficiently exploit such processing power, it is imperative to proportionally increase the performance and bandwidth of the on-chip cache memory sub-system.

Unfortunately, the commonly-used static RAM (SRAM) technology imposes fundamental limitations for the quest of high memory performance in the next generation computing systems. Indeed, the low density of SRAM cells forces to dedicate approximately 60% of the area of today's chips to the cache memories [7, 19]. Moreover, SRAM memories present a considerably high leakage power consumption becoming a considerable issue with the continuous technology scaling

A. M. H. Monazzah (✉)
Iran University of Science and Technology (IUST), Tehran, Iran

Institute for Research in Fundamental Sciences (IPM), Tehran, Iran
e-mail: monazzah@iust.ac.ir

A. M. Rahmani
University of California, Irvine (UCI), Irvine, CA, USA
e-mail: a.rahmani@uci.edu

Institute of Computer Technology, TU Wien, Austria
e-mail: antonio.miele@polimi.it

A. Miele
Politecnico di Milano, Milano, Italy
e-mail: dutt@uci.edu

N. Dutt
University of California, Irvine (UCI), Irvine, CA, USA

beyond 40 nm leading to the leakage power to contribute up to 80% of the overall energy consumption of the cache memories [7].

For these reasons, a recent trend is to consider Non-Volatile Memory (NVM) technologies, such as Phase Change Memory (PCM), Resistive RAM (ReRAM), Ferroelectric RAM (FeRAM), and Spin Transfer Torque Magnetic RAM (STT-MRAM or STT-RAM) as alternative solutions to SRAMs in multicore and many-core chips [23]. Among them, various studies [2, 3, 11, 17, 20] have observed that STT-RAM represents the most promising technology for on-chip cache memories. In particular, STT-RAM allows to increase the memory size, thanks to its higher density, and outperforms the SRAM counterpart in terms of energy consumption. In fact, since STT-RAM is a non-volatile technology, its leakage power consumption is negligible.

Unfortunately, STT-RAM technology suffers from a different set of reliability issues that has to be carefully addressed in order to realize its deployment in commercial products. STT-RAMs present a high susceptibility to failure both during write/read operations and in idle status [2, 23]. More precisely, the memory cell may suffer from *retention errors*, which are caused by thermal noises that lead to an unintentional bit flip of the value stored in an idle cell. Moreover, during read operations, it may occur that the cell content incorrectly flips leading to the so-called *read disturbance*, or the returned value has an undistinguished state, dubbed as *false read*. Finally, also the write operation may suffer from *write errors*, which are caused by thermal fluctuations in the magnetization process that lead to storing a wrong value in the cell w.r.t. the one in the input. Among these reliability threats, write errors impose the most challenging issue [2, 7].

A circuit-level strategy to eliminate write errors is to increase the current (voltage) applied during the write operation [23]. However, it mainly leads to higher energy consumption, and secondarily to a higher probability of permanent failures of the device. Indeed, higher current implies an increase in the temperature and in turn to an increase in the probability of junction barrier breakdown [7]. An alternative design strategy is the use of Error Correction Codes (ECCs) to harden the architecture of the cache memory [1]. However, if it is applied in a naive way, it may result in a significant area overhead. For instance, when Bose–Chaudhuri–Hocquenghem (BCH) 7 ECC is employed, such overhead can be as high as 15% of the data block area [2]. Such additional area causes a significant energy overhead. Given such rationale, several studies [2, 3, 6, 7, 17, 18, 20, 21, 24] have individually investigated these techniques and proposed strategies to improve their effectiveness. Given this background, we believe that there is an opportunity for a larger improvement of these hardening schemes by holistically amalgamating these two techniques. Moreover, an opportunistic integration and tuning of these two techniques can also lead to a considerable improvement in energy consumption of the cache memory architecture while at the same time guaranteeing the error rate threshold.

This chapter proposes FlexRel, a reliability improvement technique which utilizes the STT-RAM write current as an actuation knob and multi-level ECC

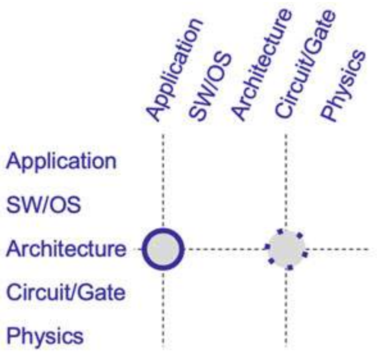


Fig. 1 Positioning of the proposed approach in the overall cross-layer vision of the book

protection scheme to conduct an optimal trade-off between reliability and energy consumption in STT-RAM cache memories. Targeting an overall block write error rate threshold that should be guaranteed in applications running on a platform, FlexRel proposes a cache way partitioning scheme that utilizes different combinations of write currents and ECC protection codes in each partition to satisfy that threshold. Then, during the run-time of applications, the FlexRel controller redirects the more vulnerable blocks to more robust partitions to keep the write error rate below the write error threshold. Within the overall cross-layer vision of the book, the main contribution of this chapter can be primarily classified as an architecture-to-application cross-layer approach, also using gate/circuit-level actuation, as illustrated in Fig. 1. In fact, this approach exploits application-level profiled information as an input to an architecture-level memory hardening technique based on ECC while using current tuning at circuit-level, with the final goal of optimizing energy consumption and application reliability.

We evaluate FlexRel using gem5 simulator [4] running SPEC CPU2006 [9] workloads. We compare the efficiency of FlexRel against an optimized uniform protection (OUP) scheme from reliability, energy, area, and performance perspectives. The simulation results show that, while FlexRel meets the write error rate threshold, it outperforms OUP scheme in terms of energy and area by up to 13.2 and 7.9%, respectively. Furthermore, The restriction of write traffics to specific partitions in FlexRel incurs only a 1.7% performance overhead to the system, on average.

The rest of the chapter is organized as follows. Section 2 introduces the necessary background presenting the basic architecture of a STT-RAM cell and the energy/error rate issues of this technology. Section 3 briefly surveys the previous approaches for hardening STT-RAMs highlighting the adopted strategies and differentiating them from the proposed approach. Section 4 is the core of this chapter and presents our proposed FlexRel approach, consisting of an enhanced memory architecture capable of trading off reliability and energy consumption. The proposed solution has been experimentally validated and results are discussed in Sect. 5. Finally, Sect. 6 draws conclusions.

2 STT-RAMs and Their Energy-Reliability Challenges

This section introduces the basics of STT-RAM technology, its architecture and how read/write operations are performed. The second part of the section focuses on energy vs. write error issues in this type of memory. This discussion represents the preliminaries for the proposed energy-aware error-tolerant scheme for STT-RAM.

2.1 Basic Architecture of STT-RAM

Figure 2 shows the basic cell structure of a STT-RAM, called 1 Transistor 1 Magnitude Tunnel Junction (MTJ), shortly 1T-1J. The cell is constructed from an MTJ element and an access NMOS transistor. MTJ itself includes three layers which are a MgO-based barrier (called *tunneling oxide barrier*), a ferromagnetic layer with fixed magnetic field direction (called *reference layer*), and a ferromagnetic layer with free magnetic field direction (called *free layer*). The MgO-based barrier layer is sandwiched between two ferromagnetic layers. STT-RAM works based on the relative magnetic field direction of the free layer and the reference layer (parallel or anti-parallel states). As shown in the figure, the parallel state will represent a logic value ”0” while the anti-parallel one a logic value ”1.” The different relative ferromagnetic field directions lead to different resistances in MTJ, i.e. R_{High} and R_{Low} (R_H and R_L) [5]. In the following, we explore the read and write operation mechanisms in a STT-RAM cell.

The read operation in a STT-RAM cell is initiated by setting the word line (WL in the figure) to turn on the access NMOS transistor. Then, a small read current I_R (or read voltage, V_R) is applied to the MTJ from the source line (SL in the figure) through an access transistor [30]. By applying I_R to MTJ, a current (or voltage) is

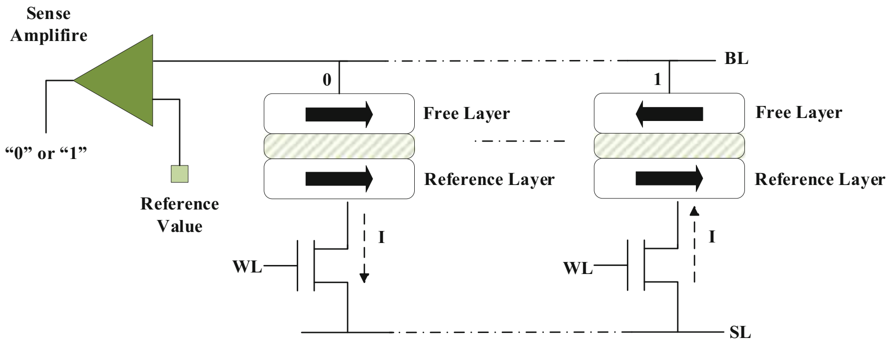


Fig. 2 A typical STT-RAM cell structure: on the left side, a parallel magnetic field direction represents a logic value 0, while on the right side, an anti-parallel magnetic field direction a logic value “1”

sensed on the bit line (BL in the figure). Based on the sensed values of current or voltage in the BL during the read operations,the resistance of MTJ is determined as high (R_H) or low (R_L). The STT-RAM cell value is determined by calculating the Tunneling Magneto Resistance (TMR), a ratio parameter defined as

$$\text{TMR} = \frac{(R_H - R_L)}{R_L} \tag{1}$$

The parallel (anti-parallel) state of MTJ leads to low (high) resistance of MTJ, calculated by means of the following equations:

$$V_{BL_L} = I_R \times (R_L + R_{NMOS}) \tag{2}$$

$$V_{BL_H} = I_R \times (R_H + R_{NMOS}) \tag{3}$$

If the sense amplifier which calculates the voltage (current) of bit line decides the sensed voltage is the same as V_{BL_L}, the MTJ value is logically “0,” otherwise, if the sensed voltage is the same as V_{BL_H}, MTJ value is logically “1.”

To perform a write operation and modify the value stored in MTJ, we need to change the magnetic field direction of MTJ free layer. By changing the magnetic field direction of the free layer, the resistance of MTJ will change [5, 10]. To this end, again the access transistor should be turned on by setting the word line. Then, a write current is applied from the source line to the baseline or vice versa. The direction of applied write current determines the magnetic field direction of the free layer. By applying the write pulse to the MTJ, when the amount of spin polarized current exceeds a threshold value, the magnetic field direction of the free layer flips.

2.2 *Error Rate vs. Energy Consumption Trade-Off*

One of the main issues in STT-RAM is its stochastic switching nature caused by the effects of thermal fluctuations. Among the side-effects of stochastic switching, write failure is the most important reliability challenge [2, 7]. More precisely, write failure occurs during the write operation and its effect is that the value stored in the MTJ will be different from the one provided as data input. From a physical point of view, it happens according to the stochastic behavior of STT-RAM cell when the magnetic field direction of the free layer could not change during the pre-determined write pulse width [12, 27]. There are many parameters that contribute to switching the MTJ state during the write operations, e.g., MTJ switching current, process variations, thermal fluctuations, and switching pulse width. According to [16], the write failure probability can be calculated using the following equation:

$$P_{wf}\left(t_w\right) = \exp\left(-t_w \cdot \frac{2\mu_B p\left(I_w - I_{C_0}\right)}{\left(c + \ln\left(\frac{\Pi^2\Delta}{4}\right)\right) \cdot \left(em\left(1 + p^2\right)\right)}\right) \tag{4}$$

where Δ is the thermal stability factor, I_{C_0} is the critical MTJ switching current at 0 °K, c is the Euler constant, e is the magnitude of electron charge, m denotes the magnetic momentum of the free layer, p is the tunneling spin polarization, μ_B is the Bohr magneton, I_w is the write current, and t_w is the write pulse width.

While STT-RAM technology is a promising candidate to resolve the static energy challenge of SRAM technology in on-chip memories, from the dynamic energy consumption perspective, it imposes considerable energy consumption for a reliable write operation due to its stochastic switching feature; the higher the I_w, the lower the write error rate of STT-RAM will be. For example, we used NVSim [8] to experimentally compare two alternative implementations, in SRAM and STT-RAM technologies, with the same 32 KB cache architecture with 64 Byte word lines implemented in 45 nm. Our results show that from the leakage power point of view, the SRAM cache imposes 41.896 mW power consumption, while STT-RAM cache only charges 9.066 mW power consumption to the design. On the other hand, each read operation in SRAM and STT-RAM implementations uses 11.421 and 82.493 pJ dynamic energy consumption, respectively. Finally, each write operation in SRAM and STT-RAM technologies enforces 5.712 and 534.375 pJ dynamic energy consumption to the design, responsively. As a conclusion, I_w in Eq. 4 is the main contributor for dynamic energy consumption in STT-RAM. Accordingly, I_w is one of the effective circuit-level knobs available to control the reliability-energy trade-off during a write operation. Generally, a lower I_w decreases the write energy, but it also amplifies the probability of write failure.

To systematically analyze this aspect, we performed a quantitative evaluation of the write error rate of STT-RAM at different write current amplitudes by using the STT-RAM SPICE model introduced in [14]. In particular, we characterized a STT-RAM cell by using parameters reported in Table 1 and ran several Monte Carlo simulations. Figure 3 depicts the write error rate of the STT-RAM cell when the write current is varied and the cell is flipped from $0 \rightarrow 1$ (in red) or vice versa (in blue). As shown in Fig. 3, we retrieved the trend lines of error rate patterns in both directions to generate the STT-RAM write error rate formulas. These formulas are useful to estimate the write error rates of STT-RAM at any write current. We therefore conclude that I_w is an effective circuit-level knob available to control the

Table 1 STT-RAM HSPICE model configurations

Parameter	Value ($\mu \pm 3\sigma$)
MTJ length	32 nm
MTJ width	96 nm
MTJ thickness	2.44 nm
Relative initial angle	$0 \pm 35° / 180 \pm 35°$
Transistors technology size	32 ± 1 nm

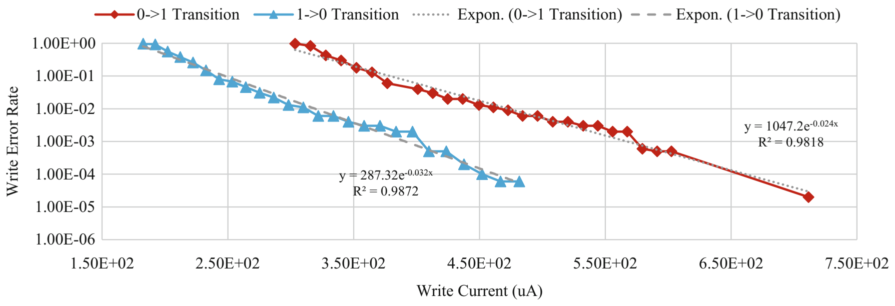


Fig. 3 STT-RAM cell write error rate vs. write current in different MTJ state transitions (write pulse width fixed at 10 ns)

quality-energy trade-off during a write operation. Moreover, it can be noted that the write error rate is asymmetric and $0 \rightarrow 1$ is the most critical transition. This is due to the fact that the initial MTJ state affects the total energy required to change its resistance [29]. For $0 \rightarrow 1$ bit transition, to change the MTJ state from parallel (low resistance) to anti-parallel (high resistance), more energy (power$cdot$time) needs to be spent compared to the amount of energy needed for the transition in the opposite direction ($1 \rightarrow 0$) [28]. As a consequence, the MTJ state transition is asymmetric from the error rate vs. energy consumption perspective.

Finally, it is worth mentioning that unlike the provided knob in SRAM technology (memory bank voltage scaling) that is coarse-grained and affects a large portion of data in memory, STT-RAM exploits I_w which is fine-grained and can be tuned for granularity of a data block in memory. As we show in the following, this feature offered by STT-RAM provides a unique opportunity for flexible adjustment of the energy-reliability knob.

3 Related Work on STT-RAM Reliability

In recent years, several studies have addressed reliability issues of STT-RAM. In [23], the authors present a survey of the preliminary approaches addressing faults in various non-volatile memory technologies with the focus on both permanent and transient faults. Regarding permanent faults, the basic strategies are to (1) increase the current during write operations and (2) augment the architecture by using Error Correction Codes (ECCs). Indeed these strategies are the ones later used by many subsequent approaches.

In [22], the authors propose a strategy, Verify and Correct (VnC), which consists of reading each value immediately after the write operation in the STT-RAM cache to assess its correctness. Since read delay is negligible, such an approach may lead to performance degradation in case of high failure rate. The approach is later enhanced by combining VnC with a limited ECC to reduce the need of rewriting upon failures. Ahn et al. [1] propose a scheme where ECC is shared among several cache blocks thus reducing its hardware cost. Cheshmikhani et al. [6] and Azad et al. [3] present ECC schemes with an optimized interleaved bit selection mechanism to minimize codewords' vulnerability variations. Finally, the authors of [25] introduce adaptiveness in the hardening scheme; where two different levels of ECC can be selectively chosen for each incoming block in the STT-RAM cache. The level of protection is selected based on the vulnerability of the incoming block which is calculated by enumerating the number of $0 \rightarrow 1$ bit transitions. The idea of using several ECC scheme in an adaptive way is further explored in [2] by defining an STT-RAM cache architecture, called A^2PT using several ECC levels and integrating a specific hardware module which selects the replacement candidate in order to minimize the Hamming distance between the stored block the newly incoming one.

A different strategy is proposed in [11] where the classical Least-Recently-Used (LRU) cache replacement policy is substituted with a new algorithm which performs a Least-Error-Rate (LER) replacement. To reduce the probability of write errors in STT-RAM, this algorithm tries to write the incoming block in a location which imposes the least number of $0 \rightarrow 1$ bit transitions among the victim block candidates. In [7], the authors observe that the stochastic switching in write operations is mainly caused by the device heating. Therefore, to reduce the write errors, they propose to replace the LRU policy with a thermal-aware counterpart, which tries to write the incoming block in a location which imposes the least temperature increase among the victim block candidates.

As discussed in Sect. 2, acting on the current applied during the write operation sensibly affects the correctness of the stored value; moreover, $0 \rightarrow 1$ is the most susceptible bit transition. For these reasons, Kim et al. [13] propose two different circuit design techniques applied at each single bit-line to balance out the asymmetric write current and optimize the memory design in terms of write-power and reliability. In a similar manner, Monazzah et al. [17] exploit the tuning of the write current to explicitly trade memory reliability for energy saving in the context of approximate computing. The approach considers software applications capable of tolerating a certain degree of errors in the results, such as image processing applications. Therefore, for each write operation, the current to be applied is dynamically selected based on the reliability requirement annotated in the application source code as well as the Hamming distance between the block to be written and the candidate to be replaced. In such a way, the energy consumption is minimized under a predefined number of errors that can be ignored in the application output.

The main contribution of our approach presented in this chapter is to holistically integrate ECC deployment and write current tuning based on our prior works presented in [2] and [17]. The main property of our approach is its self-adaptiveness to dynamically tune the system operating point to the characteristics of the running applications to optimize the energy consumption of the system while keeping the observed error rate under control.

4 FlexRel: An Energy-Aware Reliability Improvement Approach for STT-RAM Caches

In this section, we present our proposed approach called FlexRel. As a preliminary discussion, first we explore the conventional ECC deficiency in tolerating the write failures of STT-RAM caches. Then, we observe how different data patterns lead to different write error rates in cache blocks. At the end, we discuss in detail the FlexRel approach for the STT-RAM caches which utilizes the STT-RAM write current actuation knob and multi-level ECC protection scheme to conduct an optimum trade-off between reliability and energy consumption.

4.1 The Effects of Write Patterns on ECC Protection Level

As mentioned in previous sections, the write error rate in the STT-RAM cache depends on the bit differences between the contents of data that was previously stored in the cache block and the contents of the new incoming block. Indeed, during a write operation while failure may occur in the bit locations that should be toggled, for the other bit locations that will not experience any toggle, we do not observe any write error. In Sect. 2.2, we observed that in STT-RAM the write error rates of $0 \rightarrow 1$ bit transitions is higher than $1 \rightarrow 0$ bit transitions by about two orders of magnitude for the same current amplitude in both directions. Accordingly, FlexRel will mainly focus on $0 \rightarrow 1$ bit transitions since they represent the main contributor to write error rate in STT-RAM.

Generally, the total number of $0 \rightarrow 1$ bit transitions in a STT-RAM cache block is proportional to the Hamming Weight (HW) of the new incoming block, that is the total number of 1 in the bit representation of each block [2, 26]. On the other hand, the maximum number of $0 \rightarrow 1$ bit transitions in a cache block write operation happens when all of the bit locations storing value "1" in the new incoming block should store on bit locations that previously contained "0." Considering this fact, for a STT-RAM cache that is protected with an ECC code with t-bit error correction capability, we can estimate the Block Error Rate (BER) of a STT-RAM cache write operation according to Eq. 5 [26]:

$$\mathrm{BER}(w, t) \approx 1 - \sum_{i=0}^{t} C_w^i P_{ER\,0\rightarrow 1}^i (1 - P_{ER\,0\rightarrow 1})^{w-i} \tag{5}$$

where, $P_{ER\,0\rightarrow 1}$ is the bit failure rate in $0 \rightarrow 1$ switching, t the error correction capability of ECC, w the HW of the incoming data, and C_w^i the combination of HW taken i at a time.

We conducted an experimental evaluation of the BER of STT-RAM cache write operations for the incoming blocks when varying HW. In particular, we configured

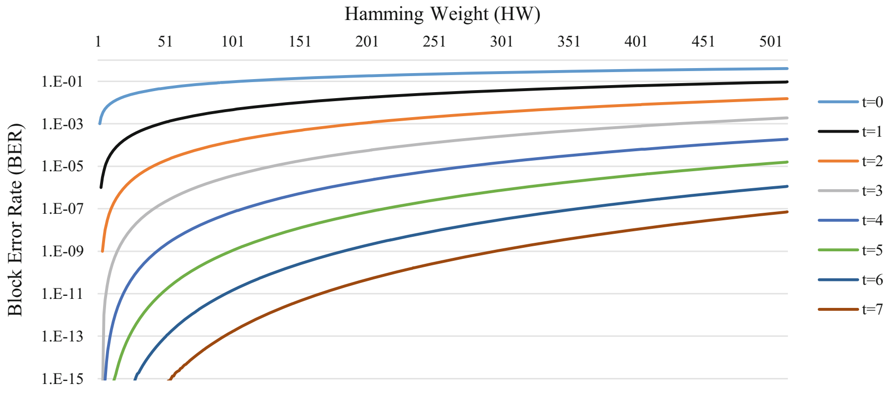


Fig. 4 Block Error Rate (BER) in different ECC schemes for various Hamming Weight (HW) data

a 512-bit STT-RAM cache block, with $0 \rightarrow 1$ write error rate, i.e., $P_{ER\,0\rightarrow 1}$ of 10^{-3} as reported in [1, 22, 26]. Figure 4 depicts the results of the experiments which verify that the BER of the cache blocks with different ECC protection scheme are considerably affected by the HW of the incoming data. For example, if we consider ECC with protection capability of t bit errors in a cache block, for HW of 25 we observed the BER of 10^{-15}, while for HW of 50 and 100 this protection scheme delivers BER of 10^{-13} and 10^{-11}, respectively, which is considerably different.

The results shown in Fig. 4 are calculated based on Bose, Chaudhuri, and Hocquenghem (BCH) coding scheme [15] which is a well-known scheme in protecting memory architectures. In BCH code, ECC converts to k-bit data and $(n - k)$ ECC check bits. The complexity of the peripherals that is required to protect the k-bit data is also increased with the increase in the ECC protection capability. Generally, for protecting a $k = 512$ bit cache line using BCH code with $(t + 1)$ bit error correction capability, we require $(10t + 1)$ check bits [2]. The results demonstrated in Fig. 4 is calculated for various coding schemes: SEC-DED (Single Error Correction-Double Error Detection), DEC-TED (Double Error Correction-Triple Error Detection), 3EC4ED, 4EC5ED, 5EC6ED, 6EC7ED, and 7ED8EC codes with $t = 1$, $t = 2$, $t = 3$, $t = 4$, $t = 5$, $t = 6$, and $t = 7$ error(s) correction capabilities, respectively.

With the emergence of ECC protection scheme in the cache memories to protect the data, conventionally, all the cache blocks are protected with the same ECC protection capability (t). This conventional ECC protection scheme is called Uniform, i.e., all of the blocks in the cache utilize the same ECC protection level. However, as we can see in Fig. 4, the different HWs in write requests lead to various BER for each cache block during the execution time. Accordingly, in the Uniform protection scheme the highest HW needs to be considered to select an ECC protection level that satisfies the write error rate threshold. As an example, considering Fig. 4, the ECC protection level $t = 6$ should be selected to satisfy the

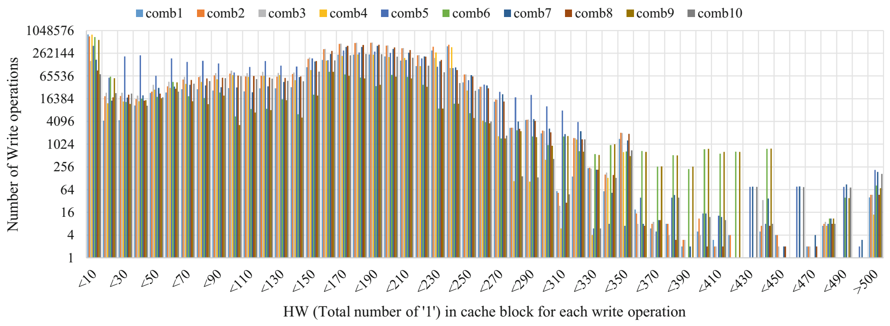


Fig. 5 The distribution of write operations based on Hamming Weight (HW) across a 4-MB 16-way shared L2 cache

write error rate threshold of 10^{-6} in the worst case where all of the bit locations in the target cache block experience toggle for the new incoming write request. However, satisfying a write request that all of its bit locations experience $0 \rightarrow 1$ bit transition is a very rare event. Therefore, using ECC with $t = 6$ correction capability is usually more than enough.

Accordingly, we investigated the distribution of HW across a 4-MByte 16-way set associative shared L2 cache during the execution of workloads. To this end, we ran combinations of benchmarks that are selected from SPEC CPU2006 [9] benchmark suite.[1] Figure 5 depicts the distribution of write requests' HWs across the shared L2 cache. Figure 5 illustrates that most of the write requests had less than 350 HW, while for the uniform full-protection ECC schemes we need to consider the worst-case HW (512 in 512 bit cache line size) for reliable write operations leading to significant under-utilization of resources.

One of the main concerns in utilizing uniform full-protection ECC configuration for the caches is the amount of energy consumption that is imposed to the system. Indeed, as illustrated in Figs. 4 and 5, while utilizing the full-protection configuration guarantees the reliable write operations, it imposes high energy consumption for most of the time that the write operations contain lower number of value 1 than the considered ECC-related threshold. For this reason, FlexRel exploits a non-uniform multi-protection level ECCs scheme to save energy. In addition, it improves the hardening scheme by deploying different write current levels introducing different STT-RAM write error rates.

[1]The details of simulator configurations and workload combinations will be mentioned later in Sect. 5, in particular in Tables 3 and 4.

4.2 FlexRel Organization

In FlexRel, we divide a cache memory into several protection level zones. Unlike the previous studies that benefit only from different ECC codes to conduct multi-level protection scheme (e.g. [2, 3, 25]), in FlexRel we consider a combination of write current level and ECC protection level to satisfy a pre-determined write error rate threshold at each zone. The main strategy that is considered in FlexRel for cache partitioning is to assign the lowest possible write current for the zones that face high amounts of write operations to alleviate STT-RAM high write energy consumption and instead apply stronger ECC protection codes in these zones to satisfy the write error threshold. On the other hand, for the zones that experience low amounts of write operations we consider high write current with weaker ECC protection codes to alleviate the static energy consumption of ECC parts of the cache ways. For the sake of better intuition, in the following we explain the FlexRel approach considering a STT-RAM L2 cache memory architecture being 16-way set-associative and having a 64 Byte (512 bit) cache line as our case study example, while in general, FlexRel approach is applicable to all associative STT-RAM caches with any configuration and at any memory abstraction level.

The first step in designing a FlexRel-equipped cache is to classify the write requests of the cache based on the HW (which shows the vulnerability of write requests) and the portion of write requests. The partitioning in FlexRel applies at way granularity. Here, as an example, we consider four protection levels in our case study FlexRel-equipped cache. The straightforward approach to assign the cache ways to one of the four protection levels is the uniform assignment (in case of 16-way set associative cache it implies to assign four ways to each protection level). Considering the efficiency challenge that was mentioned for the uniform ECC protection technique, this straightforward assignment may face a considerable waste of resources. Therefore, our approach to enhance FlexRel has been to consider once again the write request patterns depicted in Fig. 5 to configure the portion of cache ways in each protection level. For this decision, we need to consider the cumulative amount of write requests in each protection level to provide enough space for them and keep the system performance as high as possible.

Thus, we partition the 16-way FlexRel-equipped cache to four protection levels as depicted in Fig. 6. According to the figure, we assign half of the cache ways to protection level 2 ($101 \leq \text{HW} \leq 250$) which should serve the most amount of write requests. Furthermore, protection level 1 ($\text{HW} \leq 100$) which should serve the second most amount of write requests benefits from a quarter of cache ways, while each of protection levels 3 and 4 only utilizes two ways to serve their low-intensive write requests. Figure 7 depicts the proposed FlexRel scheme for our case study example which includes four zones regarding protection levels.

After the way partitioning of FlexRel-equipped cache is completed, we should determine combinations of STT-RAM write current and ECC code to deliver a reliable write operation in each zone. To select these combinations, first we should consider a write error rate threshold to meet during the write operations in FlexRel-

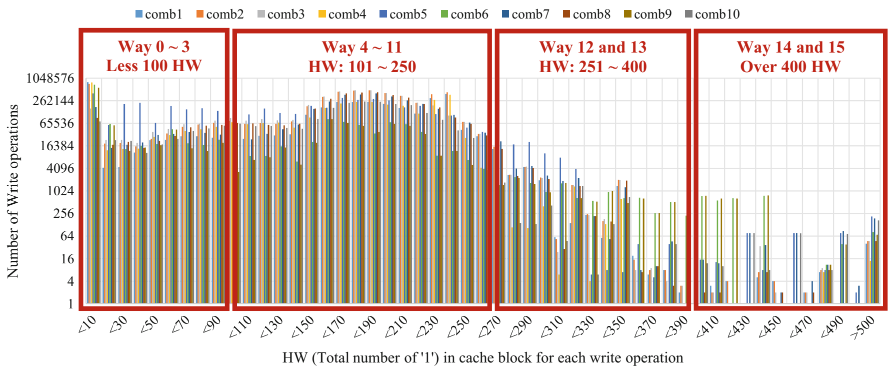


Fig. 6 Way partitioning across a 4-MB 16-way FlexRel-equipped shared L2 cache

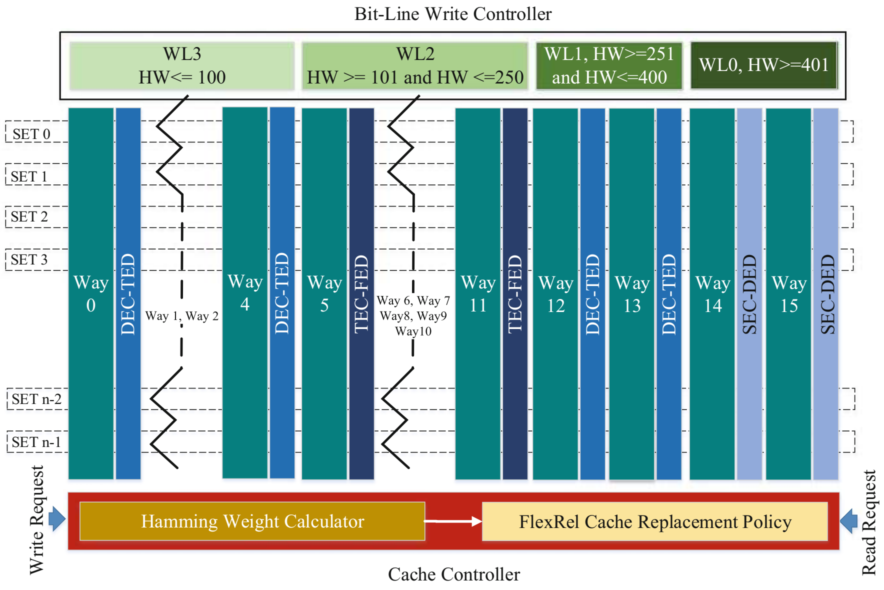


Fig. 7 A 16-way set associative FlexRel-equipped cache

equipped cache. To guarantee this threshold, FlexRel can either increase the write voltage and decrease the protection level of ECCs or vice versa. As an example, we consider 10^{-8} as write error rate threshold that should be met in all protection levels considering the write requests' HWs.

Consequently, Table 2 depicts a transducer map considered for FlexRel. As shown in the table, based on the configuration of write currents and ECC protection capabilities different dynamic energies and static powers are consumed in FlexRel-equipped cache ways. The last row of the table shows the amount of dynamic energy and static power of a uniform full-protection scheme with the same write error rate

Table 2 Transducer map for a 256 KB way embedded in a 4 MB 16-way set-associative FlexRel-equipped cache

				Cache 0 → 1 block error rate	Way dynamic energy	Way static energy		
Protection level	Write current	0 → 1 Error rate	ECC protection level (t)	Lowest HW	Highest HW	Read	Write	
1	706.2 µA	3×10^{-5}	2	None	4.4×10^{-9}	67.3 pJ	4.79 nJ	50.6 mW
2	666.9 µA	8×10^{-5}	3	1.5×10^{-10}	6.4×10^{-9}	68.5 pJ	4.61 nJ	53.1 mW
3	778.2 µA	5×10^{-6}	2	3.2×10^{-10}	1.3×10^{-9}	67.5 pJ	5.27 nJ	50.6 mW
4	935.0 µA	1×10^{-7}	1	7.9×10^{-10}	1.3×10^{-9}	66.6 pJ	6.19 nJ	48.5 mW
OUP	586.0 µA	6×10^{-4}	7	None	1.42×10^{-9}	73.3 pJ	4.37 nJ	60.6 mW

The energy consumption and Error rates depicted in this table are retrieved with the aid of NVSim [8] and HSPICE Monte Carlo® simulations threshold mentioned for FlexRel-equipped cache. It is worthy of mentioning that in each protection level, FlexRel should provide the required facilities so that the highest HW write request in that level meets write error rate thresholds. The write currents at different protection levels in Table 2 reveal the mentioned strategy in FlexRel way partitioning. For example, comparing protection level 2 (with high amounts of write operations) with protection level 4 (with low amounts of write operations), for level 2 we considered lower write current (666.9 μA) with stronger ECC code ($t = 3$) to alleviate the high write energy consumption of STT-RAMs. On the other hand, for level 4, we consider higher write current (935.0 μA) with weaker ECC code ($t = 1$) to alleviate the ECC static energy consumption.

Now that the data storage architecture of FlexRel is explored, the final element that we should consider in the architecture of FlexRel-enable cache is to design a mechanism to redirect the write request to their corresponding ways based on their HW during the execution. As depicted in Fig. 7, we developed a new replacement policy to perform this redirection.

Considering our case study example, Algorithm 1 depicts the traffic controller and replacement policy defined for FlexRel-equipped cache. This algorithm can be easily modified to apply to any other FlexRel-equipped cache with different configurations than the case study example. The FlexRel controller is responsible for calculating the HW of the incoming write request (Line 1). Then, if the write request is hit in the cache (Lines 3–10), FlexRel controller will verify the possibility of writing the new request to the hit block. To this end, FlexRel controller checks the HW of the new incoming request with the HW boundaries of the hit block's way protection level (Line 4). If the new incoming block satisfies the HW boundaries of the hit block's way protection level, FlexRel controller will satisfy the write request (Lines 5–7). Otherwise, the hit block will become invalid, and a cache miss signal will be triggered for the new incoming write request (Lines 7–10).

On the other hand, if the new incoming write request is missed in the cache (Line 11–24), based on the HW of this request that was calculated previously (Line 1), the FlexRel replacement policy should select the appropriate protection level for this request and evict a block from the protection level's assigned ways. It should be noted that FlexRel replacement policy uses LRU replacement policy in each protection level to evict the blocks (Lines 13, 16, 19, and 22).

5 Experimental Results

To explore the effectiveness of FlexRel in saving the energy consumption while meeting the reliability constraints we conducted a set of simulations. To this end, we used gem5 [4] simulating a quad-core ARM processor. The frequency of this processor is set to 1 GHz. The details of simulation configurations are summarized in Table 3. We extracted the dynamic and leakage power of STT-RAM cache ways from NVSim [8] with the aid of HSPICE. SPEC CPU2006 benchmark suites [9]

Algorithm 1: FlexRel controller and replacement policy for 16-way set associative cache

```
input  : New incoming write request (WR)
output : The target block in cache for satisfying request
1  HW = calculate_HW(WR);
2  blk = Hit(WR);
   /* Check the availability of the requested block.          */
3  if blk then
       /* The requested blk is found in the cache (hit).      */
4      satisfy_request = check_way_boundary(blk→way,HW);
5      if satisfy_request then
6          return(blk);
7      else
8          Invalidate(blk);
9          blk = null;
           /* After generating a miss signal for this request,
              FlexRel replacement policy will decide about the new
              location of this block.                          */
10     end
11 else
       /* The requested blk is not found in the cache (miss). */
12     if $HW \leq 100$ then
13         blk = LRU(way0 $\sim$ way3);
14         return(blk);
15     else if $101 \leq HW \leq 250$ then
16         blk = LRU(way4 $\sim$ way11);
17         return(blk);
18     else if $251 \leq Hw \leq 400$ then
19         blk = LRU(way12 and way13);
20         return(blk);
21     else
22         blk = LRU(way14 and way15);
23         return(blk);
24     end
25 end
```

were used as the workloads in this study. Table 4 depicts the combination of benchmarks in each workload.

It should be noted that, for the sake of improving the accuracy of the experiments, all of the simulation results were retrieved after skipping the L2 cache warm-up phase. During the experiments, we implemented and compared the following schemes:

- **Optimized Uniform Protection**—In this scheme, L2 cache ways were protected with uniform 7EC8ED BCH code with low write current mentioned in the last row of Table 2 (OUP). This uniform protection satisfied the considered block write error rate considered in this study (i.e., 10^{-8}).

Table 3 Experimental setup for gem5 simulations

Parameter	Value
ISA	ARMv7-A
No. of cores	4
L1 $ size, assoc.	32 KB, 4
L2 $ size, assoc.	4 MB, 16
Cache configuration	L1 (Private)
	L2 (Shared, FlexRel-enabled)
Cache block size	64B
Cache warm-up instructions	100 million
No. simulated instructions	100 million

Table 4 Workload combinations

Combination	Core 0	Core 1	Core 2	Core 3
Comb1	perlbench	bzip2	mcf	soplex
Comb2	perlbench	bzip2	omnetpp	xalancbmk
Comb3	perlbench	mcf	omnetpp	xalancbmk
Comb4	bzip2	mcf	soplex	xalancbmk
Comb5	gcc	bwaves	mcf	cactusADM
Comb6	namd	dealII	soplex	calculix
Comb7	perlbench	gcc	mcf	namd
Comb8	perlbench	namd	soplex	xalancbmk
Comb9	bwaves	dealII	namd	calculix
Comb10	gcc	bwaves	soplex	xalancbmk

- **FlexRel** —In this scheme, L2 cache ways were protected with variable strength ECCs and write currents mentioned in Table 2, according to the discussed cache structure in FlexRel.

Figure 8 depicts the normalized energy consumption of FlexRel-equipped shared L2 cache compared with the optimized uniform scheme. We evaluated the efficiency of FlexRel in terms of energy consumption from three perspectives, i.e., dynamic energy consumption, static energy consumption, and overall energy consumption. According to Fig. 8, FlexRel increased the dynamic energy consumption of ways by up to 19% in *comb1* which intensively used the protection levels that consume high write energy consumption. It is worth noting that since the optimized uniform scheme utilizes the least write current, the FlexRel scheme will impose more dynamic energy. On the other hand, since FlexRel utilizes low protection ECCs in comparison with optimized uniform ECC scheme, it significantly improves the static energy consumption in almost all of the combinations. Indeed, while the leakage power of the ways in the FlexRel-equipped cache was significantly lower than the optimized uniform scheme, the high-performance overhead experienced in *comb2* (see Fig. 9) led to the same static energy consumption in both schemes, and further increased the overall energy consumption by 3%.

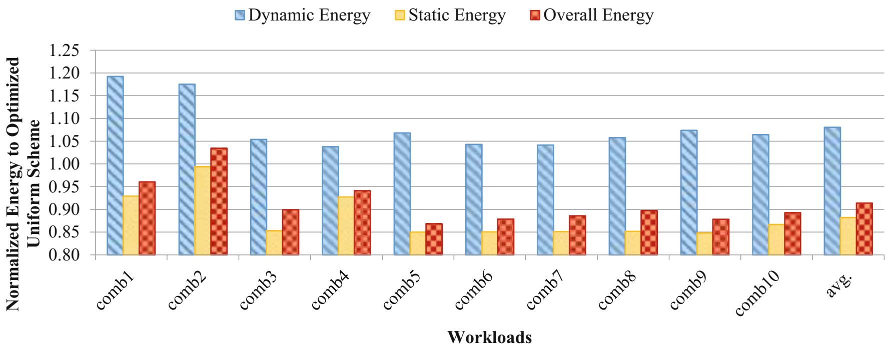


Fig. 8 Energy consumption of the ways of 4 MB shared FlexRel-equipped L2 cache normalized to optimized uniform scheme

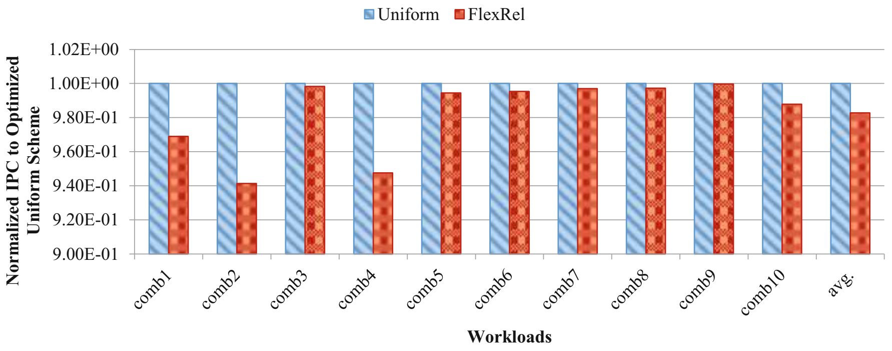


Fig. 9 The Instruction per Clock (IPC) of a system includes a 4 MB shared FlexRel-equipped L2 cache normalized to the IPC of a system that includes optimized uniform scheme in its L2 cache

In general, since static energy consumption is the main contributor in the energy consumption of the cache ways, FlexRel achieves a considerable improvement in the overall energy consumption (static + dynamic) of ways in the L2 cache. On average, while FlexRel increases the dynamic energy consumption of the ways by 8%, it saves the static energy consumption and overall energy consumption of L2 cache ways by 12% and 9%, respectively. The calculated amount of saved static power in each FlexRel-equipped cache's way is about 70.6 mW. With this amount of power, we will be able to supply the static power of more than seven 32 KB L1 caches (like the ones considered in this study) each consuming 8.9 mW static power.

Figure 9 shows the Instruction per Clock (IPC) of a system using a 4 MB shared FlexRel-equipped L2 cache normalized to the IPC of a system that incorporates the optimized uniform scheme in its L2 cache. Since FlexRel modifies the replacement policy of the L2 cache to redirect the write requests to their corresponding protection level's ways it may impose some performance penalty to the system in the situations when (1) the target protection level's ways face intensive write requests from

Table 5 The amount of way area saving in FlexRel at different protection levels compared with optimized uniform scheme

Protection Level	ECC protection level (t)	ECC check bits	% of area improvement
1	2	21	8.5
2	3	31	6.6
3	2	21	8.5
4	1	11	10.2
% of average area improvement	7.9		

different addresses or (2) we should evict a hit block because the total number of “1”s in this block forces FlexRel to change its location.

According to Fig. 9, in the worst case scenario, i.e., *comb2* that the miss penalty in FlexRel approach is considerable, the IPC of the system is degraded by 6%. This result illustrates the importance of cache partitioning in the performance of the system. In other words, while the considered partitioning provides reasonable performance for most of the workloads, for *comb1*, *comb2*, and *comb4*, the considered partitioning utilized by Algorithm 1 led to high-performance overheads for these workloads. Previously we mentioned that these performance overheads even affect the energy efficiency of FlexRel for *comb1*, *comb2*, and *comb4* workloads. On average, FlexRel decreases the IPC of the system by a negligible 1.7%.

Finally, w.r.t. area, ECC check bits assigned at each protection level are the main contributor. Accordingly, Table 5 reports ECC check-bits, and the area saving at different protection levels of FlexRel compared with the optimized uniform protection scheme. In general, considering a 16-way set associative L2 cache, the flexible scheme provided by FlexRel could save the occupied ways' area by about 7.9%, on average.

6 Conclusions and Future Work

In this chapter, we proposed FlexRel, an energy-aware reliability improvement architectural scheme for STT-RAM cache memories. FlexRel is an architecture-to-application cross-layer approach that considers a memory architecture provided with Error Correction Codes (ECCs) and a custom current regulator for the various cache ways and conducts a trade-off between reliability and energy consumption. The FlexRel cache controller dynamically profiles the number of $0 \rightarrow 1$ bit transitions of each write operation and, based on this critical parameter it selects the most-suitable cache way and current level to deliver the necessary reliability level (in terms of occurred write errors) while minimizing the energy consumption.

The results of evaluating FlexRel show that, while the scheme satisfies the reliability requirements, it delivers up to 13.2% energy saving and up to 10.2% cache ways' area saving, compared with the most efficient uniform protection scheme. The performance overhead imposed by FlexRel to the system due to the modifications of cache ways' access traffics is 1.7%, on average.

As future work, we will further improve and refine the FlexRel in two aspects. First, we will focus on the proposed replacement policy to improve the performance of the system for workloads that face significant block evictions due to HW boundary violations. To minimize the performance overhead, we will attempt to devise a dynamic cache partitioning scheme capable of changing the configuration of the FlexRel-equipped cache when considerable performance degradation is observed.

References

1. Ahn, J., Choi, K.: Selectively protecting error-correcting code for area-efficient and reliable STT-RAM caches. In: Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC), pp. 285–290 (2013)
2. Azad, Z., Farbeh, H., Monazzah, A.M.H., Miremadi, S.G.: An efficient protection technique for last level STT-RAM caches in multi-core processors. IEEE Trans. Parallel Distr. Syst. **28**(6),1564–1577 (2017)
3. Azad, Z., Farbeh, H., Monazzah, A.M.H.: ORIENT: Organized interleaved ECCs for new STT-MRAM caches. In: Proceedings of the Design, Automation Test in Europe Conference and Exhibition (DATE), pp. 1187–1190 (2018)
4. Binkert, N., Beckmann, B., Black, G., Reinhardt, S.K., Saidi, A., Basu, A., Hestness, J., Hower, D.R., Krishna, T., Sardashti, S., et al.: The gem5 simulator. ACM SIGARCH Comput. Archit. News **39**(2), 1–7 (2011)
5. Chen, Y., Li, H., Wang, X., Zhu, W., Xu, W., Zhang, T.: A nondestructive self-reference scheme for spin-transfer torque random access memory (STT-RAM). In: Proceedings of the Design, Automation Test in Europe Conference and Exhibition (DATE), pages 148–153, 2010.
6. Cheshmikhani, E., Farbeh, H., Asadi, H.: ROBIN: Incremental oblique interleaved ECC for reliability improvement in STT-MRAM caches. In: Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC), pp. 173–178 (2019)
7. Cheshmikhani, E., Farbeh, H., Miremadi, S.G., Asadi, H.: TA-LRW: A replacement policy for error rate reduction in STT-MRAM caches. IEEE Trans. Comput. **68**(3), 455–470 (2019)
8. Dong, X., Xu, C., Xie, Y., Jouppi, N.: NVSim: A circuit-level performance, energy, and area model for emerging nonvolatile memory. IEEE Trans. Comput. Aided Design Integr. Circuits Syst. **31**(7), 994–1007 (2012)
9. Henning, J.L.: SPEC CPU2006 benchmark descriptions. ACM SIGARCH Comput. Archit. News **34**(4), 1–17 (2006)
10. Hosomi, M., Yamagishi, H., Yamamoto, T., Bessho, K., Higo, Y., Yamane, K., Yamada, H., Shoji, M., Hachino, H., Fukumoto, C., Nagao, H., Kano, H.: A novel nonvolatile memory with spin torque transfer magnetization switching: Spin-ram. In: Proceedings of the IEEE International Electron Devices Meeting (IEDM), pp. 459–462 (2005)
11. Monazzah, A.M.H., Farbeh, H., Miremadi, S.G.: LER: Least-error-rate replacement algorithm for emerging STT-RAM caches. IEEE Trans. Device Mat. Rel. **16**(2), 220–226 (2016)
12. Jin, Y., Shihab, M., Jung, M.: Area power and latency considerations of STT-MRAM to substitute for main memory. In: Proceedings of the Symposium on Computer Architecture (ISCA) (2014)

13. Kim, Y., Gupta, S., Park, S., Panagopoulos, G., Roy, K.: Write-optimized reliable design of STT MRAM. In: Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED), ISLPED ’12, pp. 3–8 (2012)
14. Kim, J., Chen, A., Behin-Aein, B., Kumar, S., Wang, J., Kim, C.: A technology-agnostic MTJ SPICE model with user-defined dimensions for STT-MRAM scalability studies. In: Proceedings of the Custom Integrated Circuits Conference (CICC), pp. 1–4 (2015)
15. Lin, S., Costello, D.: Error Control Coding: Fundamentals and Applications. Prentice Hall, Upper Saddle River (1983)
16. Marins de Castro, M., Sousa, R.C., Bandiera, S., Ducruet, C., Chavent, A., Auffret, S., Papusoi, C., Prejbeanu, I.L., Portemont, C., Vila, L., et al.: Precessional spin-transfer switching in a magnetic tunnel junction with a synthetic antiferromagnetic perpendicular polarizer. J. Appl. Phys. **111**(7), 07C912 (2012)
17. Monazzah, A.M.H., Shoushtari, M., Miremadi, S.G., Rahmani, A.M., Dutt, N.: QuARK: Quality-configurable approximate STT-MRAM cache by fine-grained tuning of reliability-energy knobs. In: Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED), pp. 1–6 (2017)
18. Oboril, F., Shirvanian, A., Tahoori, M.: Fault tolerant approximate computing using emerging non-volatile spintronic memories. In: Proceedings of the VLSI Test Symposium (VTS), pp. 1–1 (2016)
19. Rahmani, A.M., Liljeberg, P., Hemani, A., Jantsch, A., Tenhunen, H.: The Dark Side of Silicon, 1st edn. Springer, Berlin (2016)
20. Ranjan, A., Venkataramani, S., Fong, X., Roy, K., Raghunathan, A.: Approximate storage for energy efficient spintronic memories. In: Proceedings of the Design Automation Conference (DAC), pp. 1–6 (2015)
21. Shoushtari, M., Rahmani, A.M., Dutt, N.: Quality-configurable memory hierarchy through approximation: Special session. In: Proceedings of the International Conference on Compilers, Architectures and Synthesis for Embedded Systems (CASES), pp. 1–2 (2017)
22. Sun, H., Liu, C., Zheng, N., Min, T., Zhang, T.: Design techniques to improve the device write margin for MRAM-based cache memory. In: Proceedings of the Great Lakes Symposium on VLSI (GLSVLSI), pp. 97–102 (2011)
23. Swami, S., Mohanram, K.: Reliable nonvolatile memories: techniques and measures. IEEE Design Test **34**(3), 31–41 (2017)
24. Teimoori, M.T., Hanif, M.A., Ejlali, A., Shafique, M.: Adam: adaptive approximation management for the non-volatile memory hierarchies. In: Proceedings of the Design, Automation Test in Europe Conference and Exhibition (DATE), pp. 785–790 (2018)
25. Wang, X., Mao, M., Eken, E., Wen, W., Li, H., Chen, Y.: Sliding basket: An adaptive ECC scheme for runtime write failure suppression of STT-RAM cache. In: Proceedings of the Design, Automation Test in Europe Conference and Exhibition (DATE), pp. 762–767 (2016)
26. Wen, W., Mao, M., Zhu, X., Kang, S., Wang, D., Chen, Y.: CD-ECC: Content-dependent error correction codes for combating asymmetric nonvolatile memory operation errors. In: Proceedings of the International Conference on Computer-Aided Design (ICCAD), pp. 1–8. IEEE, New York (2013)
27. Xu, W., Sun, H., Wang, X., Chen, Y., Zhang, T.: Design of Last-level on-chip cache using spin-torque transfer RAM (STT RAM). IEEE Trans.Very Large Scale Integr. Syst. **19**(3), 483–493 (2011)
28. Zhang, Y., Wang, X., Li, Y., Jones, A.K., Chen, Y.: Asymmetry of MTJ switching and its implication to STT-RAM designs. In: Proceedings of the Design, Automation Test in Europe Conference and Exhibition (DATE), pp. 1313–1318 (2012)
29. Zhang, Y., Zhang, L., Chen, Y.: MLC STT-RAM design considering probabilistic and asymmetric MTJ switching. In: Proceedings on International Symposium on Circuits and Systems (ISCAS), pp. 113–116 (2013)
30. Zhang, Y., Li, Y., Sun, Z., Li, H., Chen, Y., Jones, A.K.: Read performance: The newest barrier in scaled STT-RAM. IEEE Trans. Very Large Scale Integr. Syst. **23**(6), 1170–1174 (2015)

Hardware/Software Codesign for Energy Efficiency and Robustness: From Error-Tolerant Computing to Approximate Computing

Abbas Rahimi and Rajesh K. Gupta

1 Introduction

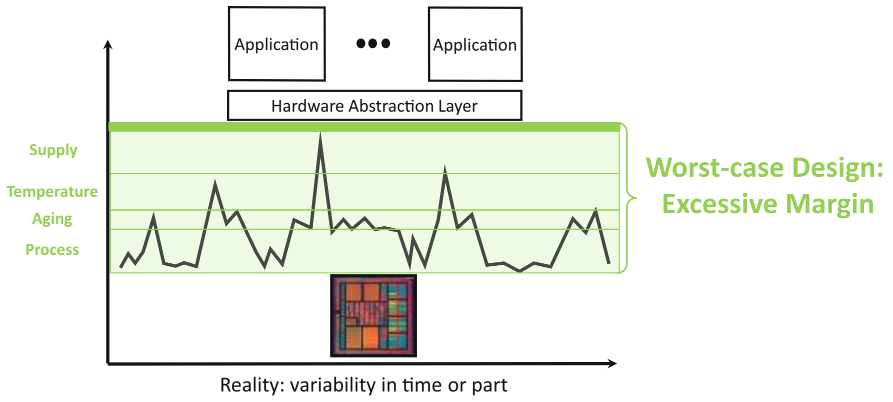
Let us step back and first look at an ideal hardware where the entire software stack can be executed. In reality, however, the hardware underneath of computing is being challenged as CMOS scaling continues to nanometer dimensions [14, 41]. The hardware experiences different sources of variability over time, or across different parts (see Fig. 1a). These variations include: manufacturing process variability that causes static variations in critical dimension, channel length, and threshold voltage of devices [6]; temporal aging/wear out variability that causes slow degradation in devices [22]; and finally, dynamic variability in ambient condition that is caused by fluctuations in operating temperature and supply voltage [8, 25]. The way that designers typically combat with these sources of variability is to consider worst-case design by imposing a large margin in hardware to ensure the correct execution of the software stack. This conservative margin leads to a loss of energy efficiency. Further, the ever-increasing amount of variability [15] limits how far we can drive down the energy per operation (i.e., voltage scaling). This means that we cannot reduce the energy as we used to.

What if we reduce the excessive margin to enable better energy scaling? The direct manifestation of reducing margin is a timing error as shown in Fig. 1b. A timing error means capturing an invalid value to a storage element like a flip-flop or a memory cell, so the result of computation might become wrong. Instead of blindly dealing with variability and its resulting timing errors, we propose to expose them to

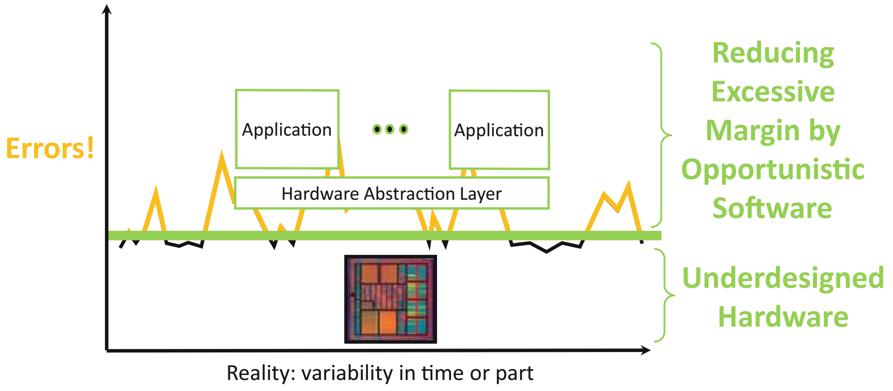
A. Rahimi (✉)
ETH Zurich, Zurich, Switzerland
e-mail: abbas@iis.ee.ethz.ch

R. K. Gupta
University of California San Diego, La Jolla, CA, USA
e-mail: gupta@cs.ucsd.edu

J. Henkel, N. Dutt (eds.), *Dependable Embedded Systems*, Embedded Systems,
https://doi.org/10.1007/978-3-030-52017-5_22


Application
Application
Hardware Abstraction Layer
Supply
Temperature
Aging
Process
Worst-case Design: Excessive Margin
Reality: variability in time or part

(a) Worst-case design by considering excessive margin leads to loss of energy efficiency.


Errors!
Application
Application
Hardware Abstraction Layer
Reducing Excessive Margin by Opportunistic Software
Underdesigned Hardware
Reality: variability in time or part

(b) Reducing excessive margin results in timing errors.

Fig. 1 Overdesigned (i.e., worst-case) hardware vs. underdesigned hardware and their interactions with software stack. (**a**) Worst-case design by considering excessive margin leads to loss of energy efficiency. (**b**) Reducing excessive margin results in timing errors

the higher levels in the stack where their side effects can be mitigated [27]. Essentially, we develop an opportunistic software layer that can operate with reduced margins and sense variability in *underdesigned* hardware—instead of overdesigned hardware with positive margins. The software layer accordingly performs adaptation by means of metadata mechanisms that reflect the state of hardware and variability, then the software can perform introspection and adaptation. The main continuations of this chapter lie on the application, software, and architectural layers as illustrated in Fig. 2.

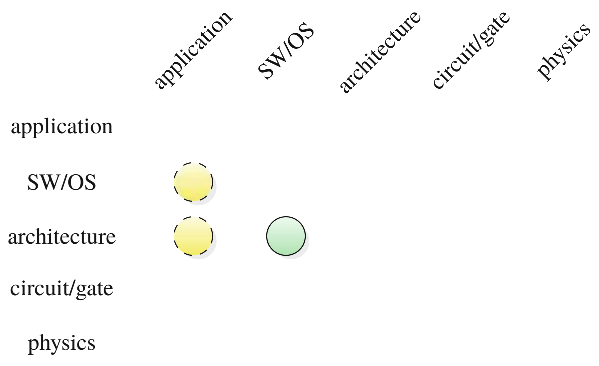


Fig. 2 Main abstraction layers of embedded systems and this chapter's major (green, solid) and minor (yellow, dashed) cross-layer contributions

1.1 Clockwise Y-Chart: From Positive Margin, to Zero Margin, to Negative Margin

In the following, we discuss about the possible approaches to reduce margin and handle timing errors. The tree possible approaches are conceptualized in a Y-chart shown in Fig. 3.

This first approach is to predict and prevent the errors by keeping a positive margin. Hence, we try to reduce the excessive margin but it is still positive to ensure the error-free execution of software. In this direction, our work spans defining and measuring the notion of error tolerance, from instruction set architecture (ISA) to procedures, and to parallel programs. These measures essentially capture the likelihood of errors and associated cost of error correction at different levels. We first characterize the manifestations of variability in ISA [28] that is the finest granularity to represent a processor functionality. Then, we characterize a sequence of instructions where the timing errors can be eliminated [31]. Going higher in the software stack, we schedule different procedure calls in multi-core architecture [29] and finally a large number of kernels on massively parallel cores [32] such that there are no timing errors. At the boundary of hardware/software, we focus on adaptive compilation methods to reduce the side effects of aging and increase lifetime for massively parallel integrated architectures like GPUs [30].

What is the next approach? The next approach is about detecting and correction errors by reducing the margin to zero (i.e., operating at the edge of errors). Basically, we reduce the margin to zero such that the errors can occur. Hence, we first need to detect the errors by means of circuit sensors [7, 11, 46] and then take actions to correct them. Our focus was instead on reducing the cost of error correction in software. In this direction, we focus on variability-aware runtime environment to cover various embedded parallel workloads in OpenMP including tasks [34, 38], parallel sections, and loops [37].

Finally, the third approach is about accepting errors—i.e., approximate computing—by pushing the margin to negative. This means that the errors and approximations are becoming acceptable as long as the outcomes have a well-

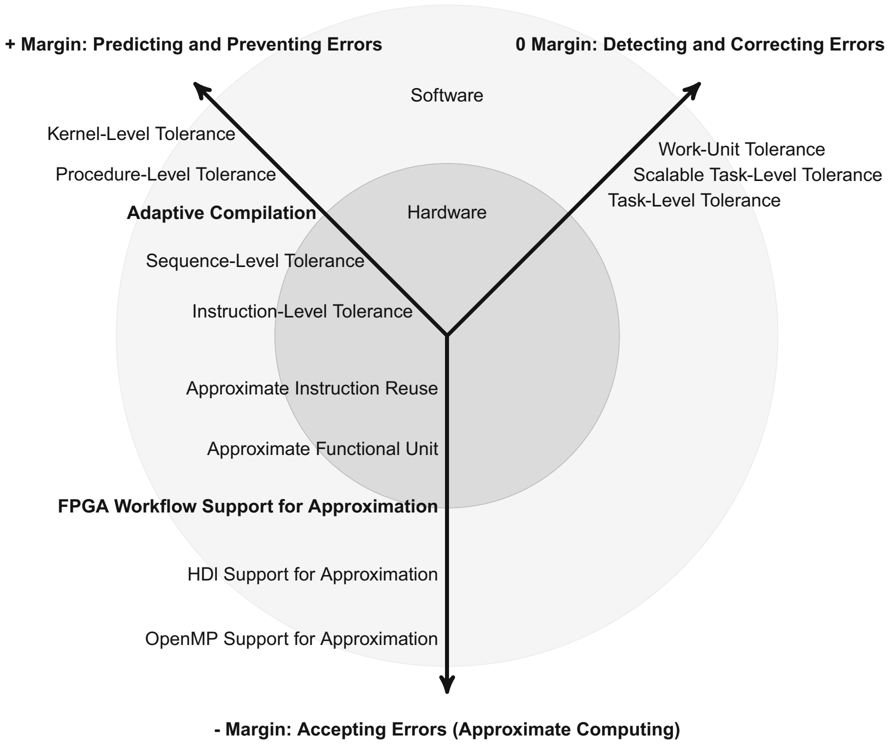

+ Margin: Predicting and Preventing Errors
0 Margin: Detecting and Correcting Errors
Software
Kernel-Level Tolerance
Procedure-Level Tolerance
Adaptive Compilation
Sequence-Level Tolerance
Instruction-Level Tolerance
Hardware
Work-Unit Tolerance
Scalable Task-Level Tolerance
Task-Level Tolerance
Approximate Instruction Reuse
Approximate Functional Unit
FPGA Workflow Support for Approximation
HDl Support for Approximation
OpenMP Support for Approximation
- Margin: Accepting Errors (Approximate Computing)

Fig. 3 Taxonomy of error tolerance in a clockwise Y-Chart: from positive margin, to zero margin, to finally negative margin

defined statistical behavior. In this approach, fatal errors can be avoided at the cost of benign approximation that can in fact allow for improving throughput and energy efficiency. Toward this goal, we enable approximate computing in instructions [33, 36, 39, 40], functional units [16–19], runtime execution environments [35], and ultimately hardware description languages [47], and high-level synthesis [23].

By looking at the Y-chart clockwise, we go from preventing errors, to correcting errors, and finally to accepting errors. This move also changes the margin from positive, to zero, and eventually to negative, leading to higher energy efficiency. In the rest of this chapter, we only focus on two methods describing how cooperative hardware/software techniques can improve energy efficiency and robustness in the presence of variability. These two methods, highlighted with bold in the Y-chart, cover examples from approaches with positive margin (i.e., predicting and preventing errors in Sect. 2) and negative margin (i.e., accepting errors in Sect. 3). Interested readers can refer to [42] for reading more about approaches with zero margin (i.e., detecting and correcting errors).

2 Positive Margin Example: Mitigating Aging in GPUs by Adaptive Compilation

In this section, we demonstrate a prime example of software that can respond to hardware variations due to aging. The goal is to reduce the excessive margin due to device aging in a setting where margin is still positive to guarantee correct execution. The idea is to combine hardware sensing circuits and adaptive software (an aging-aware compiler) to manage the workload stress. One major aging mechanism is negative bias temperature instability (NBTI) that adversely affects the reliability of a processing element by introducing new delay-induced faults. However, the effect of these delay variations is not uniformly spread across processing elements within a chip: some are affected more—hence less reliable—than others. We propose an NBTI-aware compiler-directed very long instruction word (VLIW) assignment that uniformly distributes the stress of instructions among available processing elements, with the aim of minimizing aging without any performance penalty [30]. The compiler matches the measured aging degradation with the distribution of instructions to equalize the expected lifetime of each processing element.

The rest of this chapter is organized as follows: Section 2.1 covers an overview of NBTI-induced performance degradation. Section 2.2 describes a GPU architecture and its workload distribution used in this study. Finally, our adaptive compiler is presented in Sect. 2.3.

2.1 NBTI Degradation

Among various aging mechanisms in hardware, the generation of interface traps under NBTI in PMOS transistors has become a critical issue in determining the lifetime of CMOS devices [10]. NBTI manifests itself as an increase in the PMOS transistor voltage threshold (Vth) that causes delay-induced failures (see Fig. 4 (left)). NBTI is best captured by the reaction–diffusion model [26]. This model describes NBTI in two stress and recovery phases. NBTI occurs due to the generation of the traps at the Si–SiO_2 interface when the PMOS transistor is negatively biased (i.e., during the stress phase). As a result, Vth of the transistor increases which in turn slows down the device. Removing stress from the PMOS transistor can eliminate some of the traps which partially recover the Vth shift. This is also known as the recovery phase. The work in [5] derived a long-term cycle-to-cycle model of NBTI. NBTI effects can be significant: its impact on circuit delay is about 15% on a 65 nm technology node and it gets worse in sub-65 nm nodes [4]. Further, NBTI-induced performance degradation is typically non-uniform which is a major concern for many-core GPUs, e.g., with up to 320 five-way VLIW processing elements [1].

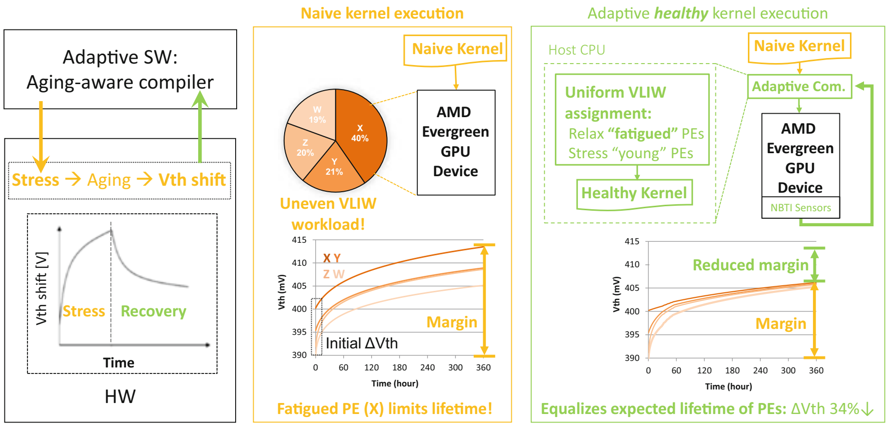


Fig. 4 Adaptive compiler to mitigating aging in GPUs: (left) sensing NBTI degradation; (middle) naive kernel execution and its impact on degradation of processing elements; (right) adaptive compiler and healthy kernel execution

2.2 *GPU Architecture and Workload*

We focus on the evergreen family of AMD GPUs (a.k.a. Radeon HD 5000 series), designed to target not only graphics applications but also general-purpose data-intensive applications. The Radeon HD 5870 GPU compute device consists of 20 compute units (CUs), a global front-end ultra-thread dispatcher, and a crossbar to connect the global memory to the L1-caches. Every CU contains a set of 16 stream cores. Finally, each stream core contains five processing elements (PEs), labeled X, Y, Z, W, and T constituting a VLIW processor to execute machine instructions in a vector-like fashion. The five-way VLIW processor capable of issuing up to five floating point scalar operations from a single VLIW consists primarily of five slots (slot_X, slot_Y, slot_Z, slot_W, slot_T). Each slot is related to its corresponding PE. Four PEs (X, Y, Z, W) can perform up to four single-precision operations separately and perform two double-precision operations together, while the remaining one (T) has a special function unit for transcendental operations. In each clock cycle, VLIW slots supply a bundle of data-independent instructions to be assigned to the related PEs for simultaneous execution. In an n-way VLIW processor, up to n data-independent instructions, available on n slots, can be assigned to the corresponding PEs and be executed simultaneously. Typically, this is not done in practice because the compiler may fail to find sufficient instruction-level parallelism to generate complete VLIW instructions. On average, if m out of n slots are filled during an execution, we call the achieved packing ratio is m/n. The actual performance of a program running on a VLIW processor largely depends on the packing ratio.

2.2.1 GPU Workload Distribution

Here, we analyze the workload distribution on the Radeon HD GPUs at architecture level, where there are many PEs to carry out computations. As it is mentioned earlier, the NBTI-induced degradation strongly depends on the resource utilization, which depends on the execution characteristics of the workload. Thus, it is essential to analyze how often the PEs are exercised during the execution of the workload. To this end, we first monitor the utilization of various CUs (inter-CU) and then the utilization of PEs within a CU (intra-CU).

To examine the inter-CU workload variation, the total number of executed instructions by each CU is collected during a kernel execution. We observe that the CUs execute almost equal number of instructions, and there is a negligible workload variation among them. We have configured six compute devices with different number of CUs, {2, 4,..., 64}, to finely examine the effect of the workload variation on a variety of GPU architectures.[1] For instance, during DCT kernel execution, the workload variation between CUs ranges from 0% to 0.26% depending on the number of physical CUs on the compute device. Execution of a large number of different kernels confirms that the inter-CU workload variation is less than 3%, when running on the device with 20 CUs (i.e., HD 5870). This nearly uniform inter-CU workload distribution is accomplished by load balancing and uniform resource arbitration algorithms of dispatcher in the GPU architecture.

Next, we examine the workload distribution among the PEs. Figure 4 (middle) shows the percentage of the executed instructions by various PEs during execution of kernels. We only consider four PEs (PE_X, PE_Y, PE_Z, PE_W) which are identical in their functions [1]; they differ only in the vector elements to which they write their result at the end of the VLIW. As shown, the instructions are not uniformly distributed among PEs. For instance, the PE_X executes 40% of ALU instructions, while the PE_W executes only 19% of the instructions. This non-uniform workload variation causes non-uniform aging among PEs. In other words, some PEs are exhausted more than other and thus have shorter lifetime as shown in Fig. 4 (middle). Unfortunately, this non-uniformity happens within all CUs since their workload is highly correlated together. Therefore, no PE throughout the entire compute device is immune from this unbalanced utilization.

The root cause of non-uniform aging among PEs is the frequent and non-uniform execution of VLIW slots. In other words, higher utilization of PE_X implies that $slot_X$ of VLIW is occupied more frequently than the other slots. This substantiates that the compiler does not uniformly assign the independent instructions to various VLIW slots, mainly because the compiler only employs optimization for increasing the packing ratio through finding more parallelism to fully pack the VLIW slots. The VLIW processors are designed to give the compiler tight control over program

[1]The latest Radeon HD 5000 series, HD 5970, has 40 CUs featuring 4.3 billion transistors in 40 nm technology.

execution; however, the flexibility afforded by such compilers, for instance, to tune the order of instructions packing, can be used towards reliability improvement.

2.3 Adaptive Aging-Aware Compiler

The key idea of an aging-aware compilation is to assign independent instructions uniformly to all slots: idling a fatigued PE and reassigning its instructions to a young PE through swapping the corresponding slots during the VLIW bundle code generation. This basically exposes the inherent idleness in VLIW slots and guides its distribution that does matter for aging. Thus, the job of dynamic binary optimizer, for k-independent instructions, is to find k-young slots, representing k-young PEs, among all available n slots, and then assign instructions to those slots. Therefore, the generated code is a "healthy" code that balances workload distribution through various slots maximizing the lifetime of all PEs (see Fig. 4 (right)). Here, we briefly describe how these statistics can be obtained from silicon, and how the compiler can predict and thus control the non-uniform aging. The adaptation flow includes four steps: (1) aging sensor readout; (2) kernel disassembly, static code analysis, and calibration of predictions; (3) uniform slot assignment; (4) healthy code generation. We explain them in the following.

The compiler first needs to access the current aging data (ΔVth) of PEs to be able to adapt the code accordingly. The ΔVth is caused by the temporal degradation due to NBTI and/or the intrinsic process variation, thus PEs even during the early life of a chip might have different aging. Employing the compact per-PE NBTI sensors [44, 45] which provide ΔVth measurement with 3σ accuracy of 1.23 mV for a wide range of temperature enables large scale data-collection across all PEs. The performance degradation of every PE can be reliably reported by a per-PE NBTI sensor, thanks to the small overhead of these sensors. The sensors support digital frequency outputs that are accessed through memory-mapped I/O by the compiler in arbitrary epochs of the post-silicon measurement. After sensor readouts, the compiler estimates the degradation of PEs using the NBTI models. In addition to the current aging data, the compiler needs to have an estimate regarding the impact of future workload stress on the various PEs. Hence, a just-in-time disassembler disassembles a naive kernel binary to a device-dependent assembly code in which the assignment of instructions to the various slots (corresponding PEs) are explicitly defined and are thus observable by the compiler. Then, a static code analysis technique is applied that estimates the percentage of instructions that will be carried out on every PE in a static sense. It extracts the future stress profile, and thus the utilization of various PEs using the device-dependent assembly code. If the predicted stress of a PE is overestimated or underestimated, mainly due to the static analysis of the branch conditions of the kernel's assembly code, a linear calibration module fits the predicted stress to the observed stress, in the next adaptation period.

Thus far, we have described how the compiler evaluates the current performance degradation (aging) of every PE and their future performance degradation due to the naive kernel execution. Then, the compiler uses this information to perform code transformations with the goal of improving reliability, without any penalty in the throughput of code execution (maintaining the same parallelism). To minimize stress, the compiler sorts the predicted performance degradation of the slots increasingly and the aging of the slots decreasingly and then applies a permutation to assign fewer/more instructions to higher/lower stressed slots. This algorithm is applied for every adaptation period. As a result of the slot reallocation, the minimum/maximum number of instructions is assigned to the highest/lowest stressed slot for the future kernel execution. This reduces ΔVth shifts by 34%, thus uniforming the lifetime of PEs and allowing for reducing the positive margin as shown in Fig. 4 (right).

Execution of all examined kernels shows that the average packing ratio is 0.3 which means there is a large fraction of empty slots in which PEs can be relaxed during kernels execution. This low packing ratio is mainly due to the limitation of instruction-level parallelism. The proposed adaptive compilation approach superposes on top of all optimization performed by a naive compiler and does not incur any performance penalty since it only reallocates the VLIW slots (slips the scheduled instructions from one slot to another) within the same scheduling and order determined by the naive compiler. In other words, our compiler guarantees the iso-throughput execution of the healthy kernel. It also runs fully in parallel with GPU on a host CPU, thus there will be no penalty for GPU kernel execution if dynamic compilation of one kernel can be overlapped with the execution of another kernel. You can refer to [49] for further details.

3 Negative Margin Example: Enabling Approximate Computing in FPGA Workflow

Modern applications including graphics, multimedia, web search, and data analytics exhibit significant degrees of tolerance to imprecise computation. This amenability to approximation provides an opportunity to reduce the excessive margin, namely to negative, by accepting errors that trade the quality of the results for higher efficiency. Approximate computing is a promising avenue that leverages such tolerance of applications to errors [12, 13, 20, 21, 24, 47]. However, there is a lack of techniques that exploits this opportunity in FPGAs.

In [23], we aim to bridge the gap between approximation and the FPGA acceleration through an automated design workflow. Exploiting this opportunity is particularly important for FPGA accelerators that are inherently subject to many resource constraints. To better utilize the FPGA resources, we devise an automated design workflow for FPGAs [23] that leverages imprecise computation to increase data-level parallelism and achieve higher computational throughput. The core of our workflow is a source-to-source compiler that takes in an input kernel and applies a novel optimization technique that selectively reduces the precision of the kernel data and operations. By selectively reducing the precision of the data and operation (analogous to setting margin to negative), the required area to synthesize the kernel on the FPGA decreases allowing to integrate a larger number of operations and *parallel* kernels in the fixed area of the FPGA (i.e., improving energy efficiency per unit of area). The larger number of integrated kernels provides more hardware context to better exploit data-level parallelism in the target applications. To effectively explore the possible design space of approximate kernels, we exploit a genetic algorithm to find a subset of safe-to-approximate operations and data elements and then tune their precision levels until the desired output quality is achieved. Our method exploits a fully software technique and does not require any changes to the underlying FPGA hardware. We evaluate it on a diverse set of data-intensive OpenCL benchmarks from the AMD accelerated parallel processing (APP) SDK v2.9 [3]. We later describe OpenCL execution model and its mapping on FPGA in Section 3.1. The synthesis result on a Stratix V Altera FPGA shows that our approximation workflow yields 1.4×–3.0× higher throughput with less than 1% quality loss (see Sect. 3.2).

3.1 OpenCL Execution Model and Mapping on FPGAs

Altera and Xilinx recently offer high-level acceleration frameworks for OpenCL [2, 43], hence we target acceleration of data-intensive computational OpenCL applications. The challenge is however devising a workflow that can be plugged into the existing toolsets and can automatically identify the opportunities for approximation while keeping the quality loss reasonably low. OpenCL is a platform-independent framework for writing programs that execute across a heterogeneous system consisting of multiple compute devices including CPUs or accelerators such as GPUs, DSPs, and FPGAs. OpenCL uses a subset of ISO C99 with added extensions for supporting data and task-based parallel programming models. The programming model in OpenCL comprises of one or more device kernel codes in tandem with the host code. The host code typically runs on a CPU and launches kernels on other compute devices like the GPUs, DSPs, and/or FPGAs through API calls. The instance of an OpenCL kernel is called a work-item. These kernels execute on compute devices that are a set of compute units (CUs), each comprising of multiple PEs having ALUs. The work-items execute on a single PE and exercise the ALU.

The Altera OpenCL SDK [2] allows programmers to use high-level OpenCL kernels, written for GPUs, to generate an FPGA design with higher performance per Watt [9]. In this work, an OpenCL kernel is first compiled and then synthesized as a special dedicated hardware for mapping on an FPGA. FPGAs can further improve the performance benefits by creating multiple copies of the kernel pipelines (synthesized version of an OpenCL kernel). For instance, this replication process can make n copies of the kernel pipeline. As the kernel pipelines can be executed independently from one another, the performance would scale linearly with the number of copies created owing to the data-level parallelism model supported by OpenCL.

In the following, we describe how our method can reduce the amount of resources for a kernel pipeline to save area and exploit remaining area resources to boost performance by replication. Our method systematically reduces the precision of data and operations in OpenCL kernels to shrink the resources used per kernel pipeline by transforming complex kernels to simple kernels that produce *approximate* results.

3.2 Source-to-Source Compiler

We provide a source-to-source compiler to generate approximate kernels from OpenCL kernels with exact specification as shown in Fig. 5. This transformation automatically detects and simplifies parts of the kernel code that can be executed with reduced precision while preserving the desired quality-of-result. To achieve this goal, our compiler takes in as inputs, an *exact* OpenCL kernel, a set of input test cases, and a metric for measuring the quality-of-result target. The compiler investigates the exact kernel code and detects data elements, i.e., OpenCL kernel variables, that provide possible opportunities for increased performance in exchange for accuracy. It then automatically generates a population of approximate kernels by

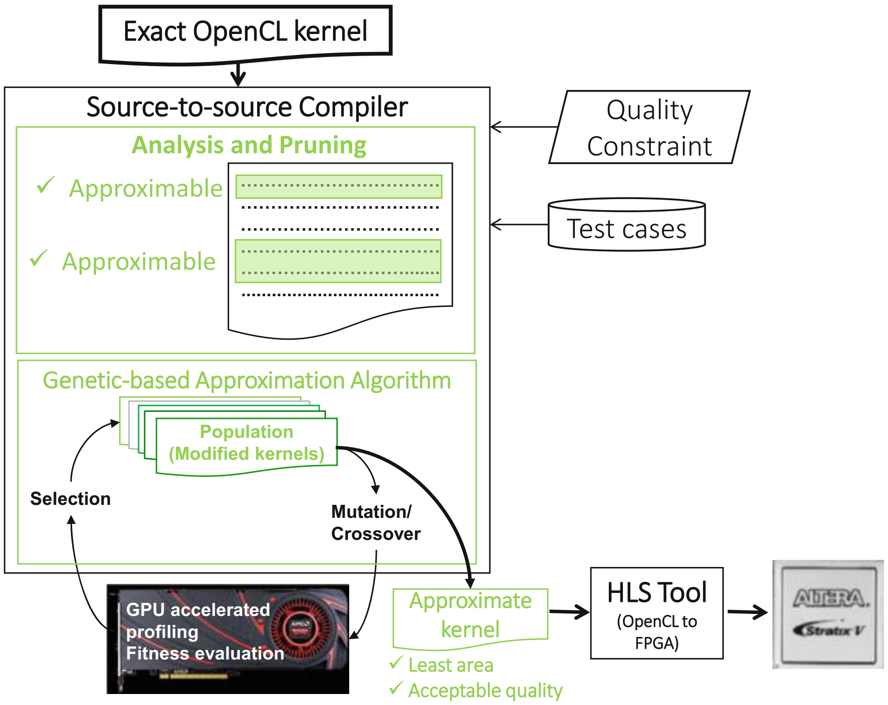


Fig. 5 FPGA approximation design workflow

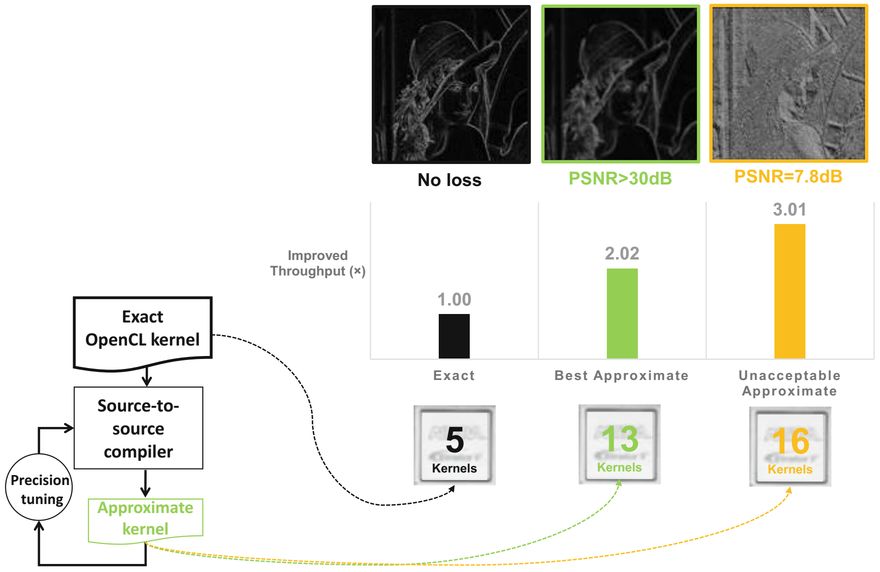


Fig. 6 Example of mapping exact and approximate kernels of Sobel filter on FPGA

means of a genetic algorithm. It can select the approximate kernels that produce acceptable results with the help of GPU profiling. These approximate kernels provide improved performance benefits by reducing the area when implemented on the FPGAs. The compiler finally outputs an optimized approximate kernel with the least area whose output quality satisfies the quality-of-result target.

The compiler uses the precision of the operations and data to tune performance as a trade-off against precision. The transformation investigates a set of kernels where in each version, some of these potential variables are replaced with a less accurate variable. To avoid a huge design space exploration, we devise an algorithm that first detects those variables that are amenable to approximation and then applies a genetic algorithm to approximate the kernel. We discuss the details of our algorithm in [23].

Figure 6 shows an example of Sobel filter kernel that is optimized by our compiler. Naive mapping of exact Sobel kernel allows us to map five instances of the kernel on the FPGA. However, by using the approximate version of kernel, we can map 13 instances of kernel on the same FPGA roughly improving throughput by 2×, thanks to the data-level parallel execution of the kernel, while meeting the quality constraint. We set the quality loss target to a maximum of 0.7% for image processing applications (which is equivalent to PSNR of a minimum 30 dB) and 1% for other applications which is conservatively aligned with other work on quality trade-offs [20, 21, 24, 48]. Benchmarking five kernels from OpenCL AMD APP SDK v2.9 shows that our compiler integrates a larger number of parallel kernels on the same FPGA fabric that leads to 1.4×–3.0× higher throughput on a modern Altera FPGA with less than 1% loss of quality. This is a prime example of accepting *disciplined* errors, in the context of approximate computing, for improved throughput. The approach can be generalized to any controllable error caused by various sources. Further details are provided in [23].

4 Conclusion

Microelectronic variability is a phenomenon at the intersection of microelectronic scaling, semiconductor manufacturing, and how electronic systems are designed and deployed. To address this variability, designers resort to margins. We show how such excessive margins can be reduced, and their effects can be mitigated, by a synergy between hardware and software leading to efficient and robust microelectronic circuit and systems.

We first explore approaches to reduce the margin and enable better than worst-case design while avoiding the errors. We demonstrate its effectiveness on GPUs where the effect of variations is not uniformly spread across over thousands processing elements. Hence, we devise an adaptive compiler that equalizes the expected lifetime of each processing element by regenerating an aging-aware healthy kernel. Such new kernel guides its workload distribution that does matter for the aging, hence effectively responding to the specific health state of GPUs.

Next, we focus on approaches that significantly reduce the margins by accepting errors and exploiting approximation opportunities in computation. We explore purely software transformation methods to unleash untapped capabilities of the contemporary fabrics for exploiting approximate computing. Exploiting this opportunity is particularly important for FPGA accelerators that are inherently subject to many resource constraints. To better utilize the FPGA resources, we develop an automated design workflow for FPGA accelerators that leverages approximate computation to increase data-level parallelism and achieve higher computational throughput.

References

1. Advanced Micro Devices, Inc: AMD Evergreen Family Instruction Set Architecture
2. Altera sdk for opencl: http://www.altera.com/products/software/opencl/opencl-index.html
3. Amd app sdk v2.9: http://developer.amd.com/tools-and-sdks/opencl-zone/amd-accelerated-parallel-processing-app-sdk/
4. Bernstein, K., Frank, D., Gattiker, A., Haensch, W., Ji, B., Nassif, S., Nowak, E., Pearson, D., Rohrer, N.: High-performance cmos variability in the 65-nm regime and beyond. IBM J. Res. Dev. **50**(4.5), 433–449 (2006). https://doi.org/10.1147/rd.504.0433
5. Bhardwaj, S., Wang, W., Vattikonda, R., Cao, Y., Vrudhula, S.: Predictive modeling of the nbti effect for reliable design. In: Custom Integrated Circuits Conference, 2006, CICC '06, pp. 189–192. IEEE (2006). https://doi.org/10.1109/CICC.2006.320885

6. Bowman, K., Duvall, S., Meindl, J.: Impact of die-to-die and within-die parameter fluctuations on the maximum clock frequency distribution. In: Solid-State Circuits Conference, 2001. Digest of Technical Papers, ISSCC, 2001, pp. 278–279. IEEE International (2001). https://doi.org/10.1109/ISSCC.2001.912637
7. Bowman, K., Tschanz, J., Kim, N.S., Lee, J., Wilkerson, C., Lu, S., Karnik, T., De, V.: Energy-efficient and metastability-immune resilient circuits for dynamic variation tolerance. IEEE J. Solid State Circuits **44**(1), 49–63 (2009). https://doi.org/10.1109/JSSC.2008.2007148
8. Bowman, K., Tokunaga, C., Tschanz, J., Raychowdhury, A., Khellah, M., Geuskens, B., Lu, S.L., Aseron, P., Karnik, T., De, V.: Dynamic variation monitor for measuring the impact of voltage droops on microprocessor clock frequency. In: Custom Integrated Circuits Conference (CICC), 2010, pp. 1–4. IEEE (2010). https://doi.org/10.1109/CICC.2010.5617415
9. Chen, D., Singh, D.: Invited paper: Using opencl to evaluate the efficiency of cpus, gpus and fpgas for information filtering. In: 22nd International Conference on Field Programmable Logic and Applications (FPL), 2012, pp. 5–12. https://doi.org/10.1109/FPL.2012.6339171
10. Chen, G., Li, M.F., Ang, C., Zheng, J., Kwong, D.L.: Dynamic nbti of p-mos transistors and its impact on mosfet scaling. IEEE Electron Device Lett. **23**(12), 734–736 (2002). https://doi.org/10.1109/LED.2002.805750
11. Drake, A., Senger, R., Deogun, H., Carpenter, G., Ghiasi, S., Nguyen, T., James, N., Floyd, M., Pokala, V.: A distributed critical-path timing monitor for a 65nm high-performance microprocessor. In: Solid-State Circuits Conference, 2007, ISSCC 2007. Digest of Technical Papers, pp. 398–399. IEEE International (2007). https://doi.org/10.1109/ISSCC.2007.373462
12. Esmaeilzadeh, H., Sampson, A., Ceze, L., Burger, D.: Architecture support for disciplined approximate programming. In: Proceedings of the Seventeenth International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS XVII, pp. 301–312. ACM, New York, NY, USA (2012). https://doi.org/10.1145/2150976.2151008. http://doi.acm.org/10.1145/2150976.2151008
13. Esmaeilzadeh, H., Sampson, A., Ceze, L., Burger, D.: Neural acceleration for general-purpose approximate programs. In: Proceedings of the 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture, MICRO-45, pp. 449–460. IEEE Computer Society, Washington, DC, USA (2012). https://doi.org/10.1109/MICRO.2012.48
14. Ghosh, S., Roy, K.: Parameter variation tolerance and error resiliency: New design paradigm for the nanoscale era. Proc. IEEE **98**(10), 1718–1751 (2010). https://doi.org/10.1109/JPROC.2010.2057230
15. Gupta, P., Agarwal, Y., Dolecek, L., Dutt, N., Gupta, R.K., Kumar, R., Mitra, S., Nicolau, A., Rosing, T.S., Srivastava, M.B., Swanson, S., Sylvester, D.: Underdesigned and opportunistic computing in presence of hardware variability. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **32**(1), 8–23 (2013). https://doi.org/10.1109/TCAD.2012.2223467
16. Jiao, X., Rahimi, A., Narayanaswamy, B., Fatemi, H., de Gyvez, J.P., Gupta, R.K.: Supervised learning based model for predicting variability-induced timing errors. In: 2015 IEEE 13th International New Circuits and Systems Conference (NEWCAS), pp. 1–4 (2015). https://doi.org/10.1109/NEWCAS.2015.7182029
17. Jiao, X., Jiang, Y., Rahimi, A., Gupta, R.K.: Wild: A workload-based learning model to predict dynamic delay of functional units. In: 2016 IEEE 34th International Conference on Computer Design (ICCD), pp. 185–192 (2016). https://doi.org/10.1109/ICCD.2016.7753279
18. Jiao, X., Jiang, Y., Rahimi, A., Gupta, R.K.: Slot: A supervised learning model to predict dynamic timing errors of functional units. In: Design, Automation Test in Europe Conference Exhibition (DATE), 2012 (2017)
19. Jiao, X., Rahimi, A., Jiang, Y., Wang, J., Fatemi, H., de Gyvez, J.P., Gupta, R.K.: Clim: A cross-level workload-aware timing error prediction model for functional units. IEEE Trans. Comput. **67**(6), 771–783 (2018). https://doi.org/10.1109/TC.2017.2783333
20. Kahng, A., Kang, S.: Accuracy-configurable adder for approximate arithmetic designs. In: Design Automation Conference (DAC), 2012 49th ACM/EDAC/IEEE, pp. 820–825 (2012)

21. Kulkarni, P., Gupta, P., Ercegovac, M.: Trading accuracy for power with an underdesigned multiplier architecture. In: 2011 24th International Conference on VLSI Design (VLSI Design), pp. 346–351 (2011). https://doi.org/10.1109/VLSID.2011.51
22. Li, X., Qin, J., Bernstein, J.: Compact modeling of mosfet wearout mechanisms for circuit-reliability simulation. IEEE Trans. Device Mater. Reliab. **8**(1), 98–121 (2008). https://doi.org/10.1109/TDMR.2008.915629
23. Lotfi, A., Rahimi, A., Yazdanbakhsh, A., Esmaeilzadeh, H., Gupta, R.K.: Grater: An approximation workflow for exploiting data-level parallelism in fpga acceleration. In: 2016 Design, Automation Test in Europe Conference Exhibition (DATE), pp. 1279–1284 (2016)
24. Moreau, T., Wyse, M., Nelson, J., Sampson, A., Esmaeilzadeh, H., Ceze, L., Oskin, M.: Snnap: Approximate computing on programmable socs via neural acceleration. In: 2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA), pp. 603–614 (2015). https://doi.org/10.1109/HPCA.2015.7056066
25. Murali, S., Mutapcic, A., Atienza, D., Gupta, R., Boyd, S., Benini, L., De Micheli, G.: Temperature control of high-performance multi-core platforms using convex optimization. In: Proceedings of the Conference on Design, Automation and Test in Europe, DATE '08, pp. 110–115. ACM, New York, NY, USA (2008). https://doi.org/10.1145/1403375.1403405. http://doi.acm.org/10.1145/1403375.1403405
26. Ogawa, S., Shiono, N.: Generalized diffusion-reaction model for the low-field charge-buildup instability at the si-sio2 interface. Phys. Rev. **51**(7), 4218–4230 (1995)
27. Rahimi, A.: From variability-tolerance to approximate computing in parallel computing architectures. Ph.D. thesis, University of California San Diego, https://escholarship.org/uc/item/1c68g008 (2015)
28. Rahimi, A., Benini, L., Gupta, R.K.: Analysis of instruction-level vulnerability to dynamic voltage and temperature variations. In: Design, Automation Test in Europe Conference Exhibition (DATE), 2012, pp. 1102–1105 (2012). https://doi.org/10.1109/DATE.2012.6176659
29. Rahimi, A., Benini, L., Gupta, R.K.: Procedure hopping: A low overhead solution to mitigate variability in shared-l1 processor clusters. In: Proceedings of the 2012 ACM/IEEE International Symposium on Low Power Electronics and Design, ISLPED '12, pp. 415–420. ACM, New York, NY, USA (2012). https://doi.org/10.1145/2333660.2333754. http://doi.acm.org/10.1145/2333660.2333754
30. Rahimi, A., Benini, L., Gupta, R.K.: Aging-aware compiler-directed vliw assignment for gpgpu architectures. In: Proceedings of the 50th Annual Design Automation Conference, DAC '13, pp. 16:1–16:6. ACM, New York, NY, USA (2013). https://doi.org/10.1145/2463209.2488754. http://doi.acm.org/10.1145/2463209.2488754
31. Rahimi, A., Benini, L., Gupta, R.K.: Application-adaptive guardbanding to mitigate static and dynamic variability. IEEE Trans. Comput. (2013). https://doi.org/10.1109/TC.2013.72
32. Rahimi, A., Benini, L., Gupta, R.K.: Hierarchically focused guardbanding: An adaptive approach to mitigate pvt variations and aging. In: Design, Automation Test in Europe Conference Exhibition (DATE), 2013, pp. 1695–1700 (2013). https://doi.org/10.7873/DATE.2013.342
33. Rahimi, A., Benini, L., Gupta, R.K.: Spatial memoization: Concurrent instruction reuse to correct timing errors in simd architectures. IEEE Trans. Circuits Syst. II Express Briefs **60**(12), 847–851 (2013). https://doi.org/10.1109/TCSII.2013.2281934
34. Rahimi, A., Marongiu, A., Burgio, P., Gupta, R.K., Benini, L.: Variation-tolerant openmp tasking on tightly-coupled processor clusters. In: Design, Automation Test in Europe Conference Exhibition (DATE), 2013, pp. 541–546 (2013). https://doi.org/10.7873/DATE.2013.121
35. Rahimi, A., Marongiu, A., Gupta, R.K., Benini, L.: A variability-aware openmp environment for efficient execution of accuracy-configurable computation on shared-fpu processor clusters. In: 2013 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS), pp. 1–10 (2013). https://doi.org/10.1109/CODES-ISSS.2013.6659022

36. Rahimi, A., Benini, L., Gupta, R.K.: Temporal memoization for energy-efficient timing error recovery in gpgpus. In: Design, Automation and Test in Europe Conference and Exhibition (DATE), 2014, pp. 1–6 (2014). https://doi.org/10.7873/DATE2014.113
37. Rahimi, A., Cesarini, D., Marongiu, A., Gupta, R.K., Benini, L.: Improving resilience to timing errors by exposing variability effects to software in tightly-coupled processor clusters. IEEE J. Emerging Sel. Top. Circuits Syst. **4**(2), 216–229 (2014). https://doi.org/10.1109/JETCAS.2014.2315883
38. Rahimi, A., Cesarini, D., Marongiu, A., Gupta, R.K., Benini, L.: Task scheduling strategies to mitigate hardware variability in embedded shared memory clusters. In: Proceedings of the 52Nd Annual Design Automation Conference, DAC '15, pp. 152:1–152:6. ACM, New York, NY, USA (2015). https://doi.org/10.1145/2744769.2744915. http://doi.acm.org/10.1145/2744769.2744915
39. Rahimi, A., Ghofrani, A., Cheng, K.T., Benini, L., Gupta, R.K.: Approximate associative memristive memory for energy-efficient gpus. In: Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition, DATE '15, pp. 1497–1502 (2015). http://dl.acm.org/citation.cfm?id=2757012.2757158
40. Rahimi, A., Benini, L., Gupta, R.K.: Circa-gpus: Increasing instruction reuse through inexact computing in gp-gpus. IEEE Des. Test **33**(6), 85–92 (2016). https://doi.org/10.1109/MDAT.2015.2497334
41. Rahimi, A., Benini, L., Gupta, R.K.: Variability mitigation in nanometer cmos integrated systems: A survey of techniques from circuits to software. Proc. IEEE **104**(7), 1410–1448 (2016). https://doi.org/10.1109/JPROC.2016.2518864
42. Rahimi, A., Benini, L., Gupta, R.K.: From Variability Tolerance to Approximate Computing in Parallel Integrated Architectures and Accelerators. Springer International Publishing (2017)
43. SDAccel: http://www.xilinx.com/products/design-tools/sdx/sdaccel.html (2015)
44. Singh, P., Karl, E., Sylvester, D., Blaauw, D.: Dynamic nbti management using a 45 nm multi-degradation sensor. IEEE Trans. Circuits Syst. I Regular Papers **58**(9), 2026–2037 (2011). https://doi.org/10.1109/TCSI.2011.2163894
45. Singh, P., Karl, E., Blaauw, D., Sylvester, D.: Compact degradation sensors for monitoring nbti and oxide degradation. IEEE Trans. Very Large Scale Integr. VLSI Syst. **20**(9), 1645–1655 (2012). https://doi.org/10.1109/TVLSI.2011.2161784
46. Tschanz, J., Bowman, K., Walstra, S., Agostinelli, M., Karnik, T., De, V.: Tunable replica circuits and adaptive voltage-frequency techniques for dynamic voltage, temperature, and aging variation tolerance. In: 2009 Symposium on VLSI Circuits, pp. 112–113 (2009)
47. Yazdanbakhsh, A., Mahajan, D., Thwaites, B., Park, J., Nagendrakumar, A., Sethuraman, S., Ramkrishnan, K., Ravindran, N., Jariwala, R., Rahimi, A., Esmaeilzadeh, H., Bazargan, K.: Axilog: Language support for approximate hardware design. In: 2015 Design, Automation Test in Europe Conference Exhibition (DATE), pp. 812–817 (2015). https://doi.org/10.7873/DATE.2015.0513
48. Yazdanbakhsh, A., Park, J., Sharma, H., Lotfi-Kamran, P., Esmaeilzadeh, H.: Neural acceleration for gpu throughput processors. In: Proceedings of the 48th International Symposium on Microarchitecture, MICRO-48, pp. 482–493. ACM, New York, NY, USA (2015). https://doi.org/10.1145/2830772.2830810. http://doi.acm.org/10.1145/2830772.2830810
49. Yuan, F., Xu, Q.: Intimefix: A low-cost and scalable technique for in-situ timing error masking in logic circuits. In: Design Automation Conference (DAC), 2013 50th ACM / EDAC / IEEE, pp. 1–6 (2013)

Reliable CPS Design for Unreliable Hardware Platforms

Wanli Chang, Swaminathan Narayanaswamy, Alma Pröbstl, and Samarjit Chakraborty

1 Introduction

Battery-operated cyber-physical systems (CPS) increasingly exist in households, factories, and the public area. For instance, zero local emission, independence from fossil fuels, and potential improvement of energy conversion efficiency have made electric vehicles (EVs) an alternative of conventional vehicles with internal combustion engines (ICEs). Design of the underlying embedded control loops such as electric motor control, braking control, stabilization, and battery management plays a crucial role in EVs and other types of battery-operated devices. Conventionally, these control loops are evaluated by a number of quality-of-control (QoC) indices. One common QoC metric is settling time. In order to ensure performance and reliability, the design also needs to take into account a number of issues on the hardware implementation platforms, such as battery behaviour and semiconductor aging. Battery is the key component influencing the device performance, when being the main power source. As the integrated circuit fabrication technology has progressed, processors become more and more susceptible to aging. In order to ensure correct functioning, the processor operating frequency has to be reduced, which could potentially worsen QoC and compromise reliability. The focus of this

W. Chang (✉)
University of York, York, YO10 5DD, UK
e-mail: wanli.chang@york.ac.uk

S. Narayanaswamy · A. Pröbstl
TU Munich, München, Germany
e-mail: swaminathan.narayanaswamy@tum.de; alma.proebst@tum.de

S. Chakraborty
UNC Chapel Hill, Chapel Hill, NC, United States
e-mail: samarjit@cs.unc.edu

J. Henkel, N. Dutt (eds.), *Dependable Embedded Systems*, Embedded Systems,
https://doi.org/10.1007/978-3-030-52017-5_23

chapter is on a design framework towards reliability of CPS considering unreliable hardware platforms.

A battery pack with large capacity is needed to offer longer usage. However, with larger capacity, the battery weight also increases leading to higher energy consumption. Moreover, the capacity is often restricted by the space that can be allocated to the battery pack. One potential solution to the above problem is to design the controller in such a way that the energy consumption of the control task can be minimized.

All off-the-shelf battery packs are labelled with a nominal capacity. However, due to the rate capacity effect, the full charge capacity (FCC) of a battery pack, which is defined to be the amount of electric charges that can be delivered from the battery after it is fully charged, actually varies with different discharging current profiles. Generally speaking, larger discharging current tends to reduce the FCC. For most common lithium-ion batteries in the market, the capacity could potentially get significantly compromised if the rate capacity effect is not properly considered in the control systems design. In this chapter, we discuss an optimization framework considering QoC as one design objective and battery usage as the other. We quantify the battery usage by the total duration that the battery can be used to continuously run the control task after one full charge. In order to maximize the battery usage, the energy consumption of the control task should be small and the battery FCC should be increased by generating a battery-friendly discharging current profile. The battery aging effect can also be incorporated. That is, the battery behaviour in the long run is another optimization dimension.

The other important design aspect is processor aging. As a processor ages, the switching time of its transistors increases, resulting in longer path delays. On-chip monitors could be used to measure the delay of the critical path. It always has to be guaranteed that the signal transmission can be completed along any path within one clock cycle. Therefore, the processor operating frequency is reduced based on the new critical path delay. On the other hand, a shorter sampling period can potentially provide a better QoC. Therefore, with a smaller processor operating frequency, the sampling period increases and QoC gets deteriorated, which is dangerous and thus highly unwanted for safety-critical applications such as electric motor control in EVs. To deal with the above situation, we can re-optimize the controller with the longer sampling period, which results from processor aging.

The entire design flow towards CPS reliability considering unreliable hardware platforms is divided into two phases. In Phase I, before the processor ages, an optimization framework is used with QoC and battery behaviour considered as design objectives. With heuristic methods implemented, this battery-aware controller design gives a Pareto front of well-distributed and non-dominated solutions. The trade-off between these objectives is explored. In Phase II, after the processor ages, QoC is found to get degraded if the controller design does not change. The same optimization framework is used with slight modification. The processor aging effect is mitigated in the way that there is a minimal change of QoC with all safety requirements satisfied.

The remainder of this chapter is organized as follows: Sect. 2 gives an overview of the background on embedded control systems design, battery rate capacity effect and aging, as well as processor aging. In Sect. 3, we present the reliable CPS design framework, and finally, Sect. 4 concludes the chapter.

2 Background

2.1 Control Systems

In this subsection, we first describe the feedback control application considered in this chapter and several background concepts. Then, we present the system modelling of the electric motor control application in EVs.

2.1.1 Basic Concepts

Plant Dynamics A control scheme is responsible for controlling a plant or dynamical system. In this chapter, we consider linear time-invariant (LTI) single-input single-output (SISO) systems,

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{A}\mathbf{x}(t) + \mathbf{B}u(t), \\ y(t) &= \mathbf{C}\mathbf{x}(t), \end{aligned} \tag{1}$$

where $\mathbf{x}(t) \in \mathbb{R}^m$ is the state vector, $\dot{\mathbf{x}}(t)$ is the derivative of $\mathbf{x}(t)$ with respect to time, $y(t)$ is the output, and $u(t)$ is the control input applied to the system. The number of dimensions for the system is m. Constant matrices $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ are of appropriate dimensions with respect to m. In a state-feedback control algorithm, the control input $u(t)$ is computed utilizing the plant states $\mathbf{x}(t)$ as feedback signals. The computed $u(t)$ is then applied to the plant, which is expected to achieve the desired behaviour.

Discretized Dynamics In most applications, the controller is implemented in a digital fashion on a computer. This implies that the plant states must be sampled when measured by the sensors. Assuming the sampling period to be a constant, the continuous-time system in (1) can be transformed into the following discrete-time system:

$$\begin{aligned} \mathbf{x}[k+1] &= \mathbf{A}_d\mathbf{x}[k] + \mathbf{B}_d u[k], \\ y[k] &= \mathbf{C}_d\mathbf{x}[k], \end{aligned} \tag{2}$$

where sampling instants are $\{t_k \mid k \in \mathbb{N}\}$, the sampling period is $t_{k+1} - t_k = h$, and

$$\mathbf{A}_d = e^{\mathbf{A}h},\ \mathbf{B}_d = \int_0^h \left(e^{\mathbf{A}t}dt\right)\mathbf{B},\ \mathbf{C}_d = \mathbf{C}. \tag{3}$$

It is noted that $\mathbf{x}[k]$ and $y[k]$ are the values of $\mathbf{x}(t)$ and $y(t)$ at $t = t_k$. The initial condition is denoted as $\mathbf{x}[0]$. The control input $u[k]$ is applied to the plant from t_k to t_{k+1}.

Feedback Controller One common goal of a control task is to make $y[k] \to r$ as soon as possible, where r is the reference for $y[k]$ to track. Towards this and other application-specific objectives, we design $u[k]$ utilizing the states $\mathbf{x}[k]$. This is then a state-feedback controller with a general structure as follows:

$$u[k] = \mathbf{K}\mathbf{x}[k] + Fr, \tag{4}$$

where $\mathbf{K}$ is the feedback gain vector and F is the feedforward gain.

Closed-Loop System With the feedback controller as shown in (4), the system closed-loop dynamics from (2) becomes

$$\mathbf{x}[k+1] = (\mathbf{A}_d + \mathbf{B}_d\mathbf{K})\mathbf{x}[k] + \mathbf{B}_d Fr = \mathbf{A}_{cl}\mathbf{x}[k] + \mathbf{B}_d Fr, \tag{5}$$

where $\mathbf{A}_{cl}$ is the closed-loop system matrix. Different locations of the closed-loop system poles, i.e., eigenvalues of $\mathbf{A}_{cl}$, result in different system behaviours. In the pole placement, poles are placed (eigenvalues are set) to fulfil various high-level goals, such as optimization of QoC and other application-specific objectives, and constraints satisfaction. In order to ensure system stabilization, all the poles must be less than unity. In this chapter, we restrict the poles in the real non-negative plane—which is common in most of the real-life design problems.

Once the poles are decided, the feedback gain vector $\mathbf{K}$ can be computed with Ackermann's formula. The static feedforward gain F used to make $y[k]$ track the reference r can be computed by

$$F = \frac{1}{(\mathbf{C}_d(\mathbf{I} - \mathbf{A}_{cl})^{-1}\mathbf{B}_d)}, \tag{6}$$

where $\mathbf{I}$ is an n-dimensional identity matrix [2].

QoC We use settling time as the metric to quantify the QoC. The time it takes for the system output $y[k]$ to reach and stay in a closed region around the reference value r (e.g., $0.98r$–$1.02r$) is the settling time and denoted as t_s. Shorter t_s indicates better QoC. When the controller poles are given, and the feedback and feedforward gains are computed accordingly, the output behaviour can be simulated, and the settling time can be derived.

Constraints There are often hard physical constraints that have to be respected in the embedded control systems, as part of the safety requirements [6, 8]. For instance, the input signal $u[k]$ could be constrained by an upper limit $U_{\max}$ and a lower limit $U_{\min}$. Similarly, the plant states could be constrained by a region. With the given controller poles and the corresponding gains, both the plant states and the control input throughout the entire control task can be simulated. Therefore, the constraints satisfaction can be evaluated.

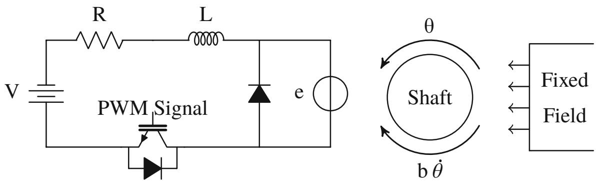


Fig. 1 A diagram of a DC motor with the armature circuit powered up by a battery pack

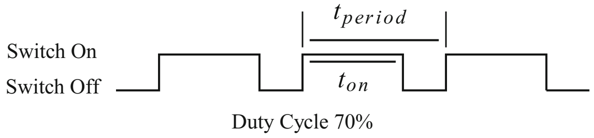


Fig. 2 Pulse-width modulation control signals in the armature circuit to adjust the DC voltage applied to the rotor

2.1.2 Electric Motor Control

Electric motor control is a key function in EVs. As shown in Fig. 1, we consider a DC motor running in the *speed control* mode. The controller is supposed to operate the motor at various speeds according to the driver input and environmental conditions. The DC voltage provided by the battery pack is V. The resistance and inductance in the armature circuit are R and L, respectively. The back electromotive force (EMF) from the motor is e. The insulated gate bipolar transistor (IGBT) works as a switch controlled by pulse-width modulation (PWM) signals at the gate. When the switch is on, V is applied to the armature circuit. When the switch is off, the diode flows out the remaining current in the motor and thus the applied voltage is equivalent to zero. Periodic PWM signals are shown in Fig. 2, where the duty cycle c is calculated as

$$c = \frac{t_{on}}{t_{period}}, \tag{7}$$

and the effective voltage applied in the armature circuit is

$$V_{eff} = cV. \tag{8}$$

We can clearly see that V_{eff} is adjustable between 0 and V by controlling the PWM signals.

In general, the torque T generated by a DC motor is proportional to the armature current i and the strength of the magnetic field. We assume the magnetic field to be constant and thus the torque is calculated as

$$T = K_t i, \tag{9}$$

where K_t is the motor torque constant. We denote the angular position of the motor to be θ. The angular velocity and acceleration are then $\dot{\theta}$ and $\ddot{\theta}$, respectively. The back EMF is proportional to the angular velocity of the shaft by a constant factor K_e as follows:

$$e = K_e\dot{\theta}. \tag{10}$$

A viscous friction model is assumed and the friction torque is proportional to the shaft angular velocity $\dot{\theta}$ by a factor of b. Now we can derive the following governing equations based on Newton's second law and Kirchhoff's law:

$$\begin{aligned} J\ddot{\theta} + b\dot{\theta} &= K_t i, \\ L\frac{di}{dt} + Ri &= V_{eff} - K_e\dot{\theta}, \end{aligned} \tag{11}$$

where J is the moment of inertia of the motor. It is noted that in the steady state (i.e., $\ddot{\theta} = 0$),

$$\dot{\theta} = \frac{K_t i}{b}. \tag{12}$$

The state-space system modelling as in (1) becomes

$$\begin{aligned} \frac{d}{dt}\begin{bmatrix} \dot{\theta} \\ i \end{bmatrix} &= \begin{bmatrix} -\frac{b}{J} & \frac{K_t}{J} \\ -\frac{K_e}{L} & -\frac{R}{L} \end{bmatrix}\begin{bmatrix} \dot{\theta} \\ i \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{L} \end{bmatrix} V_{eff}, \\ y &= \begin{bmatrix} 1 & 0 \end{bmatrix}\begin{bmatrix} \dot{\theta} \\ i \end{bmatrix}. \end{aligned} \tag{13}$$

The states are the angular velocity of the motor $\dot{\theta}$, constrained in $[0,\ \dot{\theta}_{\max}]$, and the armature current i, constrained in $[0,\ i_{\max}]$. The control input is the effective voltage V_{eff}, constrained in $[0,\ V]$ as discussed above. The system output is $\dot{\theta}$. The control goal is to make $\dot{\theta}$ track r.

2.2 Battery

Batteries are increasingly used as power source for many applications nowadays ranging from low-power applications such as portable electronics, wearable devices to high-power applications such as EVs and stationary electrical energy storage (EES) systems for smart grid applications [3]. Lithium-ion battery chemistry has been dominating the market for most low-power and high-power applications mainly due to their high energy and power densities compared to other rechargeable battery chemistry. While the terminal voltage and nominal capacity of a single lithium-ion cell are limited for achieving high operating voltages and high capacities required for EVs, multiple individual lithium-ion cells are combined in series or parallel to form a high-power battery pack.

Major concerns affecting the widespread adoption of EVs include range anxiety and battery degradation that will result in an early replacement of their power source. For instance, battery packs in EVs have to be replaced when their state-of-health (SoH), a ratio of capacity at present to the capacity when the battery was new, falls below 70%. In addition to the long-term aging, battery packs are also subject to capacity degradation within individual charging–discharging cycles. This is mainly due to the *rate capacity* effect, which states that discharging a battery with a higher current will reduce the overall capacity of the pack that can be used in this cycle. Therefore, while designing control applications that use battery as a power source, the capacity degradation at single charging–discharging cycles and long-term battery aging have to be considered for maximizing the battery usage and its lifetime.

2.2.1 Battery Basics

Batteries are electrochemical storage devices, meaning their chemical reaction is coupled with an electron transfer. They perform a reversible chemical reaction, which allows them to store electrical energy (*charging*) and release the stored electrical energy by performing the opposite reaction (*discharging*). The basic unit of a rechargeable battery is an electrochemical cell, which consists of a positive electrode *cathode*, a negative electrode *anode*, and an electrolyte to favour the movement of the charge carriers between the two electrodes inside the cell as shown in Fig. 3.

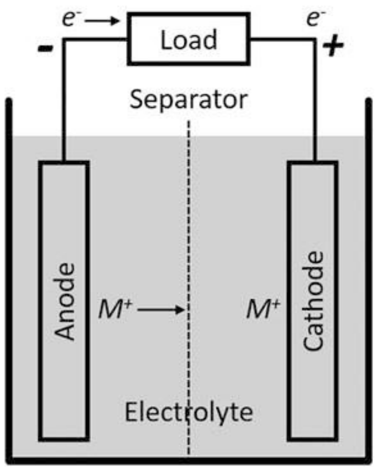


Fig. 3 Electrochemical cell. Shuttle ions (M^+) are oxidized at the anode and move towards the cathode releasing an electron (e^-) to the outer circuit powering the load [13]

During the discharging process, shuttle ions (M^+) are oxidized at the anode side and release electrons (e^-), which travel through the outer circuit to power the load. The oxidized shuttle ions move through the electrolyte to the cathode inside the cell and are reduced by the incoming electrons from the outer circuit. This process is represented by the following equations:

$$Anode\colon M \rightarrow M^+ + e^- \qquad (Oxidation) \tag{14}$$

$$Cathode\colon M^+ + e^- \rightarrow M \qquad (Reduction). \tag{15}$$

The opposite reaction takes place during charging, facilitating storage of electrical energy in the form of chemical reactions.

In the ideal case, one would assume that while discharging the voltage of the electrochemical cell as seen by the load stays constant throughout the discharging process and suddenly drops to zero when the battery capacity is empty. Moreover, the capacity of the battery stays constant irrespective of the amplitude of the discharge current. However, in reality, the battery exhibits several non-linear effects and as a result the battery voltage instead of remaining constant slowly decreases with time while discharging. Furthermore, the usable capacity of a battery significantly depends on the rate of the load current. Discharging a battery with a higher current will result in a reduced effective capacity obtained from the cell.

2.2.2 Rate Capacity Effect

The FCC of a battery pack is reduced when a battery is discharged with a higher discharge current [5]. This can be seen from Fig. 4 where discharging a cell with a higher current reaches the lower threshold voltage faster than discharging with a lower current. This effect is termed as *rate capacity* or *rate* effect. The fundamental concept behind the rate capacity effect can be explained in terms of overpotential as in [13]. Whenever a current is drawn from a battery, the voltage of that battery will drop depending upon the magnitude of the discharging current. For a battery to obtain maximum energy output, the cell voltage V_T should follow the discharge profile of the equilibrium voltage V_0, which is defined as the cell voltage at the chemical equilibrium at a given state of charge and temperature. However, the cell voltage deviates with the discharging current and this deviation is termed as overpotential η, which can be expressed as

$$\eta = V_0 - V_T. \tag{16}$$

This overpotential is mainly divided into three parts as ohmic, activation, and concentration overpotentials. At higher states of charge, the cell voltage is predominantly dominated by ohmic overpotentials, which behaves like a resistive drop to the cell voltage and as the cell discharges to a lower state of charge, the activation and

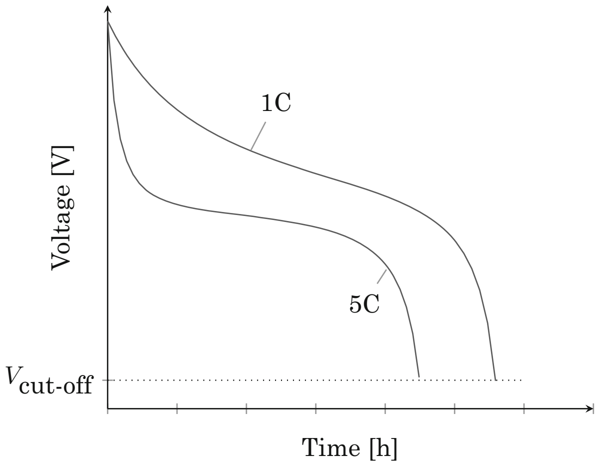


Fig. 4 Higher discharge current results in a reduced usable capacity of the cell due to rate capacity effect. The discharge rate is given by C rating, where 1C means the cell is discharged completely within 1 h [9]

concentration overpotentials dominate the ohmic drop. This reduction in cell voltage and the capacity due to the overpotentials of the cell is termed as rate capacity effect.

In battery terminology, the *C-rate* is often used to define the charge or discharge current of a battery. 1C corresponds to the current necessary to charge or discharge the battery completely in 1 h, whereas a 2C discharge will deplete the battery in half hour. The rate capacity effect is modelled by using Peukert's law [9] as

$$L = \frac{a}{I^b}, \tag{17}$$

where L is the battery lifetime, I is the discharge current, a and b are constants obtained from experiments. In ideal case a would be the battery capacity and b would be equal to 1, whereas in reality a is close to the battery's capacity and b is greater than one. While this model holds good for predicting battery capacity for constant continuous load, it does not work well with variable or interrupted loads. An extended version of Peukert's law was proposed in [15] as

$$L_t = \frac{a}{\left(\frac{\sum_{k=1}^{n} I_k\left(t'_{k+1}-t'_k\right)}{L_t}\right)^b}, \tag{18}$$

where $t'_1 = 0$ is the starting time stamp and $L_t = t'_{n+1}$ is the total duration that the battery can be used and divided into n slots.

2.2.3 Battery Aging

In addition to the single cycle capacity loss due to rate capacity effect, which can be rectified by reducing the discharging current at subsequent cycles, battery aging is a long-term process where the battery cell cannot hold the same amount of charge as it was new. Battery aging can be classified into calendar aging and cycling aging, where the former refers to the loss of capacity due to storing at high states of charge and high temperatures and the latter refers to the loss of lithium-ions due to the charge/discharge process. The main factors for battery cycling aging are depth of discharge (DoD), average state of charge, state-of-charge swing, temperature, and the rate of the discharge current [18] as

$$Q_{loss} = f\left(t, T, DoD, Rate\right), \tag{19}$$

where t is the cycling time. Without the DoD, which does not significantly affect the cycling capability of lithium-ion cells, the capacity loss can be modelled with the following equation:

$$Q_{loss} = B.exp\left(\frac{-E_a}{RT}\right)(A_h)^z, \tag{20}$$

where R is the gas constant, T is the temperature, A_h is the ampere-hour throughput of the cell, z is the power law factor, and B is a constant obtained by experimental data. With multiple experimental analysis for different discharge rates performed in [18], the value of z was approximated to 0.55 and the constant B was calculated for each C-rate. Figure 5 shows that with a higher discharge current the battery capacity drops significantly and will reach their end-of-life faster than discharging at a lower discharge current.

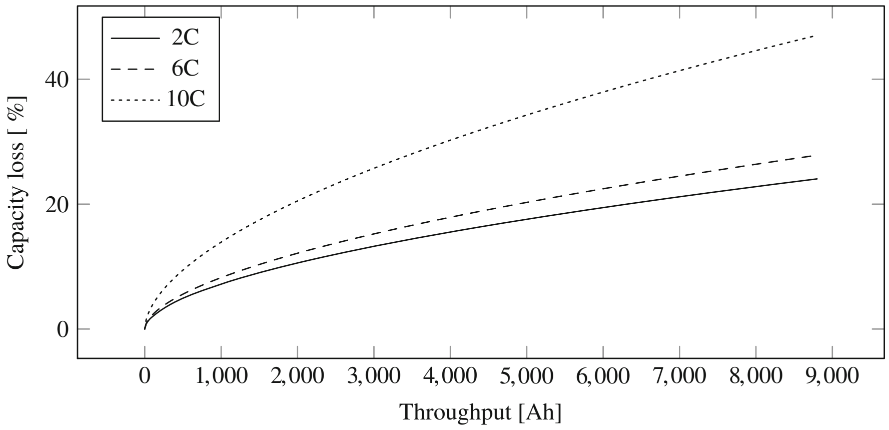


Fig. 5 Capacity loss due to higher discharge rates [5]

The capacity loss with discharging can be approximately modelled by the following equation as proposed in [18]:

$$Q_{loss} = B.exp\left[\frac{-31700 + 370.3 \cdot C_{rate}}{RT}\right](A_h)^{0.55}. \tag{21}$$

2.3 Processor Aging

Processors are known to age over time and stress resulting in reduced operating speed. This is problematic in the sense that lower processor speed negatively impacts the performance of applications running on it.

2.3.1 Aging Mechanisms

The main transistor aging mechanisms are hot carrier injection and negative bias temperature instability [1, 17]. Hot carrier injection results in changes in the threshold voltage of the semiconductor. Similarly in the case of negative bias temperature instability, the threshold voltage of MOSFETs is increased. These aging-induced voltage changes result in longer transistor switching time. And as a consequence, the operation of the transistor becomes less reliable, which is of course highly undesirable. Such increased switching time lowers the performance of a processor and of applications running on the processor. Applications then potentially violate performance requirements and produce faulty calculation results, which in most cases is not acceptable.

2.3.2 Countermeasures

As a countermeasure to increased path delays, chips typically would run at very conservative clock rates, also called guard bands or safety margins. Such guard bands include enough margin to achieve the same clock rate throughout the whole intended lifetime of a processor. Intuitively, we see that this pessimistic approach results in a huge waste of resources or energy as the processor could generally achieve much higher speeds [11]. The problem even becomes more severe as we see a trend to decreasing transistor sizes which increases operational variations.

However an increase in the supply voltage could compensate for aging circuits [16]. By that, the delays could be kept constant and the operating frequency could stay the same throughout the intended lifetime of the processor. An adaptive control circuit accommodates for the currently required voltage settings. The downside of this approach is quadratically increased dynamic power consumption of the processor [14] and additional constraints such as maximum input current, cooling requirements, and temperature-dependent reliability problems [17].

Another measure to be taken is to decrease the operating frequency of the processor to compensate for critical path delays [1]. Other than increasing the supply voltage, decreasing the operating frequency actually lowers the power consumption. However, the processor becomes slower, which results in degraded control performance [4] and schedulability issues [12]. Nevertheless, dynamic operating frequency adjustments are a promising approach due to not negatively impacting the overall energy consumption while maintaining high usage of resources.

If aging could be measured on-chip in real-time, the operating voltage and frequency could be adjusted to always provide the maximum speed possible. This however means that control applications—that were designed for a higher speed, which becomes infeasible at some point in time—need to be readjusted to the changes. Such on-chip aging monitors have been developed for the delay of critical paths. Paths that potentially become critical in the future need to be identified first and then their degradation needs to be watched [19]. The path timing monitors typically work on replications of the paths that have statically been identified as the critical ones to not interfere with real functions. The information gathered from the replicated paths is then used to decide if the operating frequency needs to be changed and the according new frequency can be determined from the monitored delays. Processors that implement such critical path monitors are also called autonomic frequency scaling processors.

From the application designers' perspective, the processors lose speed over time and this needs to be considered when designing applications that will run on those processors. Lower processor operating frequency results in longer worst-case execution time of programs. However, in control applications that require high sampling frequencies this worst-case execution time may become the bottleneck for reliable control output. As already outlined above, safety margins between worst-case execution time and sampling period are possible but costly as the processor would run much below its capabilities throughout most of its lifetime. Hence, making full use of the available resources, here the processor speed, is of high interest in cost-sensitive domains.

2.3.3 Aging Estimation

A simple model of critical path delays uses temperature, supply voltage, and stress time, i.e., time during which the processor is active [12]. Let us use this model to consider the use case of processor aging in an electrical taxi. Assuming that the electrical taxi is in use for two-thirds of a day with drivers taking shifts, i.e., for 16 h, we can now estimate the decrease of the processor speed used in the car. We find that after 2 years of taxi use, the on-time of the processor is approximately 1.33 years. As a consequence, the processor speed has degraded in the worst case by roughly 7%. The degradation then continues. After four and then 10 years, the corresponding duration of the processor being switched on amounts to 2.66 and 6.66 years, respectively. These on-times relate to critical path delays of roughly 9% and 12%. As a vehicle usually incorporates many real-time and safety-critical tasks on multiple processors and at the same time the automotive domain can be considered to be very cost-sensitive, such delays need to be considered in the design stage.

2.3.4 Related Work from the Software Perspective

Multiple works have proposed techniques to design software for aged processors or to reduce the aging process. Processors that have slowed down due to aging have higher execution delays of tasks, which is particularly problematic in the context of hard real-time systems and safety-critical applications [12]. As a result, the schedulability analysis for such safety-critical systems needs to take the estimated worst-case execution delays into consideration and the traditional problem formulation needs to be extended by system lifetime constraints. All scheduled tasks with worst-case delays need to meet their respective deadlines at all times even with severe aging-induced slow-down of the processor speed.

Mitigation of aging can be done in multi-core systems. Such systems often use redundant multi-threading to reduce soft errors. The aging variation among cores is due to varying workloads. Such unbalanced aging states are highly undesired as the system lifetime is constrained by the weakest component, i.e., the slowest core. As a remedy, the mapping of tasks should consider the current aging status of the respective cores. The proposed system [10] maps tasks in a way that aging variations are mitigated and aging of already slower cores is reduced.

3 Reliable CPS Design Framework

We formulate the reliable CPS design on unreliable hardware platforms to be an optimization problem with two objectives—t_s to quantify QoC and L_t to quantify the battery usage. We aim to minimize t_s and maximize L_t. Usually an optimization technique takes objectives either to minimize or maximize but not both. Therefore, we minimize $f_1 = t_s$ and $f_2 = -L_t$. It is noted that L_t is only related to the single-cycle behaviour with the battery rate capacity effect. Other objectives with respect to battery aging can also be defined, such as the total duration that a battery can run the control task after some time like 1 year, i.e., the L_t in 1 year, or when the capacity drops below the threshold like 70%.

The constraints are on the plant states and control input. Additional constraints on the objectives can be imposed depending on the requirements. For example, t_s can be set as shorter than or equal to 20 s. The decision variables are the poles that are less than unity on the real non-negative plane. Clearly, the decision space is continuous. Given a set of decision variables, the objectives and constraints can be evaluated as explained in Sect. 2.

There are generally two goals to pursue in solving bi-objective or multi-objective optimization problems. First, the final solution set (i.e., the obtained Pareto front) only consists of non-dominated points. By convention, Point A is said to dominate

Point B, if Point A is better than or equal to Point B in all objectives and better than Point B in at least one objective. Second, the final solution set has a good distribution in terms of objective values. This gives designers better options under different circumstances.

It is challenging to solve the formulated non-convex optimization problem with a continuous design space. Stochastic population-based heuristics such as the non-dominated sorting genetic algorithm (NSGA) can be used. In NSGA, an initial population is first generated and serves as parents. Offspring are then produced with crossover and mutation. The crossover function tries to keep the good genes of parents, which in this context means that the offspring are close to parents in the decision space.[1] The mutation function aims to better explore the decision space. Elitism is implemented for environmental selection, so that the next generation is selected among both the parents and offspring. This not only speeds up convergence but also ensures that good solutions will not be lost once they are found. There are two termination conditions whether the population has converged and whether the maximum allowed number of generations has been reached.

In selection, all the parents and offspring are sorted and ranked by domination. For each point, the number of points that dominate it (i.e., dominating points) is its rank. The new generation is filled in a way that points with lower ranks have priorities. This sorting feature values dominance more than the differences in individual objectives.

Among all the non-dominated points obtained by the above NSGA-based optimization, some may be very close to others in both the objectives. Therefore, it is not necessary to keep all of them. We need to choose a few points to form a well-distributed final solution set. First of all, we define the crowding distance below. As illustrated in Fig. 6, assuming that there are two objectives $\{f_1, f_2\}$ and n solution points $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$[2] ordered by the value of either objective, for each point $\mathbf{x}_i$, $i \in \{1, 2, \ldots, n\}$, that is not at the end of this point sequence, the crowding distance of $\mathbf{x}_i$ in terms of the objective f_k, $k \in \{1, 2\}$ can be calculated as

$$q_i^k = |f_k(\mathbf{x}_{i+1}) - f_k(\mathbf{x}_{i-1})|, \tag{22}$$

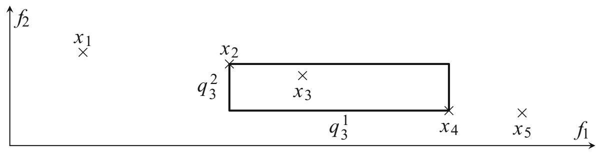


Fig. 6 Illustration of the crowding distance calculation

[1]This may not be the case in general. Offspring can be quite different from the parents.

[2]These are just general notations to explain the method. In this chapter, the decision variables are the poles as discussed earlier.

Algorithm 1 Removal of less representative solution points according to the crowding distance ranking

```
Input: $S = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}, n_d, \{\rho_1, \rho_2\}, \{f_1, f_2\}$
Output: $S_d = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{n_d}\}$
26 for $j \in \{1, 2, \ldots, n - n_d\}$ do
27     for $i \in \{1, 2, \ldots, n - j + 1\}$ do
28         calculate $q_i^1$ and $q_i^2$ for the element $\mathbf{x}_i$ as in (22)
29     for $k \in \{1, 2\}$ do
30         sort S based on $q_i^k$ from maximum to minimum  for $i \in \{1, 2, \ldots, n - j + 1\}$ do
31             assign the position index (1 for maximum to $n - j + 1$ for minimum) of $\mathbf{x}_i$ in S to
               $r_i^k$
32     for $i \in \{1, 2, \ldots, n - j + 1\}$ do
33         calculate $r_i^0$ as in (23)
34     Remove the element with the maximum $r_i^0$ from $S$
35 $S_d = S$
```

where $\mathbf{x}_{i+1}$ and $\mathbf{x}_{i-1}$ are the two closest points to $\mathbf{x}_i$ on each side, respectively. Since we deal with a set of Pareto points that are non-dominated, $\mathbf{x}_{i+1}$ and $\mathbf{x}_{i-1}$ are closest to $\mathbf{x}_i$ in terms of both objectives. Both the end points of the point sequence are assumed to have infinite crowding distance calculation.

The algorithm removing the less representative points to achieve a good distribution is shown in Algorithm 1. The desired number of Pareto points is denoted as n_d. First, for each point, we calculate the two crowding distances corresponding to the two objectives. (Lines 2–4) Two ranks r^1 and r^2 are assigned to it based on the comparison in crowding distances with other points. (Lines 5–10) If the point $\mathbf{x}_i$ has the maximum crowding distance in terms of f_1 among all the n points, then $r_i^1 = 1$. If $\mathbf{x}_i$ has the minimum crowding distance, then $r_i^1 = n$. The overall rank of $\mathbf{x}_i$ is

$$r_i^0 = \rho_1 r_i^1 + \rho_2 r_i^2, \tag{23}$$

where ρ_1 and ρ_2 are importance factors of the two objectives, respectively. (Lines 11–13) These values depend on the application and

$$\rho_1 + \rho_2 = 1. \tag{24}$$

For example, if in an application, only the distribution in terms of f_1 is important, we may set ρ_1 to be 1 and ρ_2 to be 0. In this case, r_i^0 is equal to r_i^1 and all the points are ranked according to their crowding distances in terms of f_1. After each point $\mathbf{x}_i$ has an overall rank r_i^0, the point with the largest r_i^0 is removed from the solution set. (Line 14) The entire process starting from crowding distance calculation is iterated until the desired number of points n_d is reached. Both the end points of the point sequence are always kept in the set (due to the infinite crowding distances)

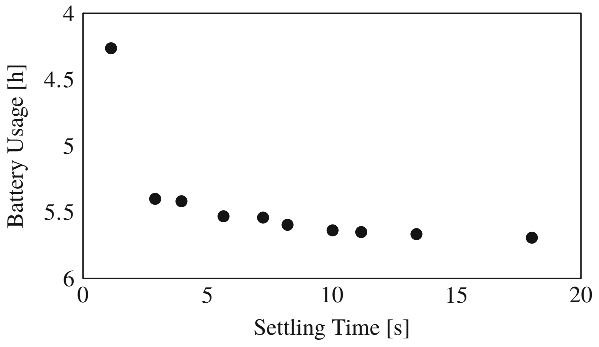


Fig. 7 An example trade-off between QoC and battery usage

Table 1 The ten design options: the original values and the aged values

Design option	t_s	L_t	Aged t_s		Aged L_t	
1	1.1257 s	4.2643 h	1.2369 s	9.88%	4.4601 h	4.59%
2	2.8972 s	5.4000 h	3.1859 s	9.96%	5.2717 h	−2.38%
3	3.9540 s	5.4182 h	4.3483 s	9.97%	5.2742 h	−2.66%
4	5.6430 s	5.5315 h	6.2062 s	9.98%	5.3111 h	−3.98%
5	7.2314 s	5.5412 h	7.9534 s	9.99%	5.5423 h	0.02%
6	8.2142 s	5.5959 h	9.0345 s	9.99%	5.5141 h	−1.46%
7	10.0238 s	5.6377 h	11.0251 s	9.99%	5.5639 h	−1.31%
8	11.1746 s	5.6506 h	12.2910 s	9.99%	5.5627 h	−1.56%
9	13.3920 s	5.6663 h	14.7302 s	9.99%	5.4613 h	−3.62%
10	18.0271 s	5.6930 h	19.8287 s	9.99%	5.5239 h	−2.97%

The percentage with the aged values is computed based on the original values

to maintain the coverage of the solution set. It is noted that Algorithm 1 takes two objectives into account and can be trivially extended for more objectives.

An example trade-off between QoC and battery usage with the electric motor control presented earlier in this chapter is illustrated in Fig. 7. As the processor ages, there is a decrease in the processor operating frequency and an increase in the sampling period. Taking the number 10% as an example, the change of both objectives is reported in Table 1. In 8 out of the 10 design options shown in Fig. 7, the aged points are dominated by the original points. That is, the settling time is increased (with a positive percentage) and the battery usage is decreased (with a negative percentage). The average deterioration in the control performance is 9.97%. The average deterioration in the battery usage is 1.53%. It is noted that for design option 2, the constraints on the plant states and control input, as discussed earlier in this chapter, are not satisfied anymore.

The processor aging effect can be mitigated by re-optimizing the controller poles with the prolonged sampling period, using the design framework earlier in this chapter. After obtaining the Pareto front, there can be different ways to reach the final solution set. For instance, Algorithm 1 can be deployed again. Alternatively, for each design option, we can keep the point that is closest to the original point

Table 2 The ten design options: the recovered values with re-optimization

Design option	Recovered t_s		Recovered L_t	
1	1.1475 s	−7.23%	4.2345 h	−5.06%
2	2.8389 s	−10.89%	5.3361 h	1.22%
3	3.9264 s	−9.70%	5.4116 h	2.60%
4	5.9551 s	−4.05%	5.5294 h	4.11%
5	7.5233 s	−5.41%	5.5676 h	0.46%
6	8.1094 s	−10.24%	5.5754 h	1.11%
7	10.8225 s	−1.84%	5.6383 h	1.34%
8	10.8225 s	−11.95%	5.6383 h	1.36%
9	13.7615 s	−6.58%	5.6595 h	3.63%
10	13.7615 s	−30.60%	5.6595 h	2.45%

The percentage is computed based on the aged values

in the settling time. The latter is executed in this case. The recovered results after re-optimization are shown in Table 2. In 9 out of the 10 design options, the recovered points dominate the aged points. That is, the settling time is decreased (with a negative percentage) and the battery usage is increased (with a positive percentage). The average improvement in the control performance is 9.85%. The average improvement in the battery usage is 1.32%. It should be noted that the design options 7 and 8 have the same recovered point. So do the design options 9 and 10. For all the design options, the constraints on the plant states and control input are guaranteed to be satisfied.

4 Concluding Remarks

In this chapter, we have discussed a design optimization framework for CPS. We consider unreliable hardware platforms with respect to processor aging and battery aging and rate capacity effect. The trade-off between the QoC and battery usage is explored. Furthermore, when the processor ages, both the QoC and battery usage get deteriorated, and safety requirements may be violated. The processor aging effect can be mitigated by re-optimizing the controller with the prolonged sampling period, using the same design framework. The change of QoC is minimal and the safety requirements are guaranteed to be met—leading to reliable CPS design. Besides the processor and battery, there are other hardware components that can be unreliable and should be investigated, e.g., the memory and communication systems [7].

References

1. Bowman, K., Tschanz, J., Wilkerson, C., Lu, S.L., Karnik, T., De, V., Borkar, S.: Circuit techniques for dynamic variation tolerance. In: 2009 46th ACM/IEEE Design Automation Conference, pp. 4–7. IEEE, New York (2009)
2. Chang, W., Chakraborty, S.: Resource-aware automotive control systems design: a cyber-physical systems approach. Found. Trends Electron. Des. Autom. **10**(4), 249–369 (2016)
3. Chang, W., Lukasiewycz, M., Steinhorst, S., Chakraborty, S.: Dimensioning and configuration of ees systems for electric vehicles with boundary-conditioned adaptive scalarization. In: International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS) (2013)
4. Chang, W., Pröbstl, A., Goswami, D., Zamani, M., Chakraborty, S.: Battery-and aging-aware embedded control systems for electric vehicles. In: 2014 IEEE Real-Time Systems Symposium, pp. 238–248. IEEE, New York (2014)
5. Chang, W., Proebstl, A., Goswami, D., Zamani, M., Chakraborty, S.: Reliable CPS design for mitigating semiconductor and battery aging in electric vehicles. In: 2015 IEEE 3rd International Conference on Cyber-Physical Systems, Networks, and Applications, pp. 37–42 (2015)
6. Chang, W., Roy, D., Zhang, L., Chakraborty, S.: Model-based design of resource-efficient automotive control software. In: IEEE/ACM International Conference on Computer-Aided Design (ICCAD) (2016)
7. Chang, W., Goswami, D., Chakraborty, S., Ju, L., Xue, C., Andalam, S.: Memory-aware embedded control systems design. IEEE Trans. Comput. Aided Design Integr. Circuits Syst. 36(4), 586–599 (2017)
8. Chang, W., Goswami, D., Chakraborty, S., Hamann, A.: OS-aware automotive controller design using non-uniform sampling. ACM Trans. Cyber-Phys. Syst. **2**(4), 26 (2018)
9. Jongerden, M., Haverkort, B.: Battery Modeling. No. TR-CTIT-08-01 in CTIT Technical Report Series, Design and Analysis of Communication Systems (DACS) (2008)
10. Knebel, F., Rehman, S., Shafique, M., Henkel, J.: Ageopt-rmt: compiler-driven variation-aware aging optimization for redundant multithreading. In: 2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC), pp. 1–6 (2016). doi:10.1145/2897937.2897980
11. Lefurgy, C.R., Drake, A.J., Floyd, M.S., Allen-Ware, M.S., Brock, B., Tierno, J.A., Carter, J.B.: Active management of timing guardband to save energy in power7. In: Proceedings of the 2011 44th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), pp. 1–11. IEEE, New York (2011)
12. Masrur, A., Kindt, P., Becker, M., Chakraborty, S., Kleeberger, V., Barke, M., Schlichtmann, U.: Schedulability analysis for processors with aging-aware autonomic frequency scaling. In: Proceedings of the 2012 IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, pp. 11–20. IEEE, New York (2012)
13. Narayanaswamy, S., Schlueter, S., Steinhorst, S., Lukasiewycz, M., Chakraborty, S., Hoster, H.E.: On battery recovery effect in wireless sensor nodes. ACM Trans. Des. Autom. Electron. Syst. **21**(4), 60:1–60:28 (2016)
14. Park, J., Abraham, J.A.: A fast, accurate and simple critical path monitor for improving energy-delay product in dvs systems. In: Proceedings of the 17th IEEE/ACM international symposium on Low-power electronics and design, pp. 391–396. IEEE, New York (2011)
15. Rakhmatov, D.N., Vrudhula, S.B.K.: An analytical high-level battery model for use in energy management of portable electronic systems. In: IEEE/ACM International Conference on Computer Aided Design (ICCAD 2001). IEEE/ACM Digest of Technical Papers (Cat. No.01CH37281), pp. 488–493 (2001)
16. Stojanovic, V., Markovic, D., Nikolic, B., Horowitz, M.A., Brodersen, R.W.: Energy-delay tradeoffs in combinational logic using gate sizing and supply voltage optimization. In: Proceedings of the 28th European Solid-State Circuits Conference, pp. 211–214. IEEE, New York (2002)

17. Tschanz, J., Kim, N.S., Dighe, S., Howard, J., Ruhl, G., Vangal, S., Narendra, S., Hoskote, Y., Wilson, H., Lam, C., et al.: Adaptive frequency and biasing techniques for tolerance to dynamic temperature-voltage variations and aging. In: 2007 IEEE International Solid-State Circuits Conference. Digest of Technical Papers, pp. 292–604. IEEE, New York
18. Wang, J., Liu, P., Hicks-Garner, J., Sherman, E., Soukiazian, S., Verbrugge, M., Tataria, H., Musser, J., Finamore, P.: Cycle-life model for graphite-lifepo4 cells. J. Power Sour. **196**(8), 3942–3948 (2011)
19. Wang, S., Chen, J., Tehranipoor, M.: Representative critical reliability paths for low-cost and accurate on-chip aging evaluation. In: Proceedings of the International Conference on Computer-Aided Design (ICCAD '12), pp. 736–741. ACM, New York (2012). doi:10.1145/2429384.2429543. http://doi.acm.org/10.1145/2429384.2429543

Power-Aware Fault-Tolerance for Embedded Systems

Mohammad Salehi, Florian Kriebel, Semeen Rehman, and Muhammad Shafique

1 Introduction

High core integration in multi-/many-core chips can facilitate reliability management through exploiting different hardening modes considering variants of redundant multithreading (RMT) [17]. However, in such large-scale chips the maximum number of cores that can simultaneously operate is constrained by the thermal design power (TDP, i.e., the maximum amount of power a chip is expected to dissipate and the nominal value for the cooling system to be designed around). Under a given TDP budget either less cores can be powered-on at the full performance level (the power-gated cores are referred to as dark silicon) or relatively more cores can be powered-on at a lower performance level (referred to as "dim" or "gray" silicon) [16]. In case TDP is exceeded, the elevated on-chip temperatures beyond the cooling capacity aggravate reliability threats like temperature-dependent transient faults and aging [1, 5, 6], unless the chip is throttled down which may lead to performance degradation. Reliability management under TDP constraints becomes even more challenging in the presence of manufacturing process variations that result in chip-to-chip or core-to-core variations in the maximum operating frequency and leakage power. This chapter presents a system-level power–reliability management technique for dark silicon multi-/many-core processors that jointly accounts for transient faults, process variations, and the TDP constraint.

M. Salehi (✉)
Guilan University, Rasht, Iran
e-mail: mohammad.salehi@guilan.ac.ir

F. Kriebel · S. Rehman · M. Shafique
Technische Universität Wien (TU Wien), Wien, Austria
e-mail: florian.kriebel@tuwien.ac.at; semeen.rehman@tuwien.ac.at; muhammad.shafique@tuwien.ac.at

J. Henkel, N. Dutt (eds.), *Dependable Embedded Systems*, Embedded Systems,
https://doi.org/10.1007/978-3-030-52017-5_24

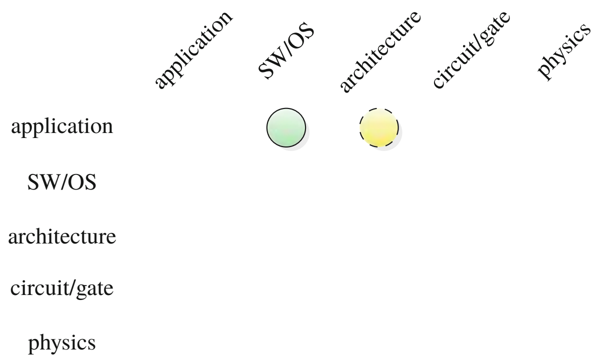


Fig. 1 Main abstraction layers of embedded systems and this chapter's major (green, solid) and minor (yellow, dashed) cross-layer contributions

In this chapter, at first, the system modeling including the power consumption and reliability models, as well as the reliability techniques are presented. Then, the power, reliability, and performance tradeoffs at the software and hardware levels as well as for different hardening modes are studied. After that, the power–reliability management technique is presented. It jointly considers multiple hardening modes at the software and hardware levels, each offering distinct power, reliability, and performance properties. At the software level, it leverages multiple reliable code versions that vary in terms of their reliability, performance, and power properties. At the hardware level it exploits different protection features and different RMT modes, subjected to the manufacturing process variations and different operating conditions (like changing the voltage–frequency levels). Finally, a framework for the system-level optimization is introduced. It considers different power–reliability–performance management problems for many-core processors depending upon the target system and user constraints (i.e., power, reliability, and performance constraints).

The main contributions of this chapter in the scope of this book lie on the application, SW/OS, and architectural layers as illustrated in Fig. 1.

2 System Models

2.1 Power Consumption Model

Power consumption in digital systems consists of static power (e.g., due to sub-threshold leakage) and dynamic power (mainly dissipated due to the circuit switching activities). The power consumption, when the system is operating under a supply voltage and a corresponding maximum allowable frequency (we call it a voltage and frequency (V-f) level), can be written as Eq. 1. For the systems that support the dynamic voltage and frequency scaling (DVFS), different V-f levels are specified at the design time, while at run time, a V-f level is selected considering the system workload [10]. The static power P_{Static} increases exponentially when the threshold voltage (V_{th}) decreases and is proportional to the supply voltage (V). The dynamic power $P_{Dynamic}$ is proportional to the circuit switching activity (α), load capacitance (C_L), operating frequency (f), and the square of the supply voltage (V) [11, 12, 14, 15].

$$P(V, f) = P_{Static} + P_{Dynamic} = I_0 e^{\frac{-V_{th}}{\eta V_T}} V + \alpha C_L V^2 f \tag{1}$$

2.2 Fault and Reliability Models

In this chapter, transient faults which appear randomly in the underlying hardware and then disappear after certain time are considered. Examples of transient faults are single- and multiple-bit upsets due to energetic radiation particle strikes, circuit metastability, signals cross-talk, voltage noises due to electromagnetic interference (EMI), etc. [1, 5]. Transient faults in the hardware level may manifest themselves as bit-flips in the memory or combinational logic, i.e., the so-called soft errors. These errors may propagate to the software level (e.g., as silent data corruption, crash, halt, and wrong register values) and may finally result in a software failure [5].

These transient faults occur randomly and are typically modeled as a Poisson process with rate λ. The fault rate increases exponentially with a decrease in supply voltage V, as Eq. 2 [11, 12, 15].

$$\lambda(V) = \lambda_0 10^{\frac{V_{max} - V}{\Delta}} \tag{2}$$

In Eq. 2, λ_0 is the raw fault rate at the maximum voltage V_{max} (i.e., the minimum value for fault rate λ) and the parameter Δ determines the amount of increase in fault rate with one step decrease in voltage.

The *software's vulnerability* to soft errors due to hardware-level transient faults at the instruction-level can be quantified by the function vulnerability index (FVI) model [8, 9]. This model projects the error probability for an application software considering vulnerabilities of different instructions (modeled using instruction vulnerability index (IVI)) when executing through different hardware units (e.g., different pipeline stages) in a core. The IVI refers to the probability of an instruction's result being erroneous. It accounts for *temporal vulnerabilities* of different instructions (i.e., different instructions have different execution latency, instruction dependency, and intervals of the operand values) as well as *spatial vulnerabilities* (i.e., different hardware components occupy different chip area and perform different operations) [8, 9]. Knowing the hardware-level fault rate (λ) and the software vulnerability to soft errors (FVI), the software failure rate can be projected as $\lambda(V) \cdot FVI$. Accordingly, the functional reliability FR of an application execution that is defined as the probability of failure-free execution of the application can be written as [11, 12, 15]:

$$FR(FVI, c, V, f) = e^{-\lambda(V)\cdot FVI\cdot \frac{c}{f}} \tag{3}$$

In Eq. 3, $\frac{c}{f}$ is the application execution time under the operating frequency f and c is the number of clock cycles that are required by the core to finish the application execution.

Besides the *functional reliability* of an application, in many systems (e.g., real-time embedded systems), it is also required that the application execution has to finish before a deadline, referred to as *timing reliability* (i.e., probability of meeting deadlines). To jointly consider the functional reliability FR and timing reliability TR, the functional–timing reliability model of Eq. 4 can be employed. In this model, the parameter $0 \leq \beta \leq 1$ specifies the priority for functional and timing reliability. For example, for the systems with tight timing constraints, lower values for β are considered, and for the systems with severe constraints of timing and reliability (e.g., hard real-time systems), $\beta = 0.5$ can be considered to represent the same priority for functional and timing reliability.

$$R = \beta FR + (1-\beta)TR \tag{4}$$

The reliability for a single application execution given by Eq. 4 may not satisfy the reliability constraint of the target system. In the following section, we study reliability techniques and hardening modes that can be used for soft error mitigation and reliability improvement in many-core processors.

2.3 Reliability Techniques

One prominent technique for tolerating transient faults in many-core processors is *process level redundancy (PLR)*, where multiple identical copies of an application task are executed on different cores and the application finishes successfully if at least one of the task executions finishes successfully (i.e., the application execution fails only if all the task executions fail). Therefore, the *total application reliability* is defined as the probability of at least one application task being executed successfully. Suppose that n identical copies of an application task are executed on n different cores and possible faults can be detected with the probability of μ_{FD}. The total reliability of the application can be calculated as:

$$R_{total}(R_1,\ R_2,\ \ldots,\ R_n) = \mu_{FD}\left(1-\prod_{i=1}^{n}(1-R_i)\right) \tag{5}$$

In Eq. 5, R_i is the reliability of the i-th task copy execution (given by Eq. 4). Here, it is assumed that (1) there is no spatial correlation between fault occurrences in different cores and (2) parallel task executions on different cores are not dependent from the viewpoint of fault propagation, i.e., a fault occurrence in a core does not affect the operation of the other cores. This assumption is also considered for the other reliability techniques in this chapter in which we have parallel task executions on different cores.

2.3.1 Software Error Detection with Re-execution (SEDR)

In this technique, each application task is executed on a single core and a software error detection mechanism (e.g., software-based control flow checking and acceptance tests) is used for error detection. Here, if an error occurs during the task execution, the task is re-executed once again on the same core for error recovery.

Therefore, the reliability of this technique can be calculated by Eq. 6, where μ_{SFD} is the error detection coverage of the software error detection mechanism (i.e., the probability of detecting existing errors).

$$R_{SEDR}(R) = \mu_{SFD}(R + (1 - R)R) = \mu_{SFD}(2R - R^2) \tag{6}$$

Here, it is assumed that there is no temporal correlation between the fault occurrences in consecutive executions of the same task, i.e., a fault occurrence during an execution of a task does not affect the next execution of the same task. This assumption is also considered for the other reliability techniques in this chapter in which we have consecutive task executions on the same core.

2.3.2 Dual Modular Redundancy (DMR) with Re-execution (DMRR)

Software-based error detection in the SEDR technique may not provide a high error detection coverage and also it may not be useful for some applications, e.g., it may entail incurring extra delay that may not be acceptable for hard real-time systems. One practical and powerful error detection mechanism is the comparison of the output result. In this mechanism, two identical copies of each application task are executed on different cores in parallel and the output results of the task are compared for error detection (i.e., DMR is applied at the individual core level). If the comparison task finds that the results are in agreement, the result is assumed to be correct. The implicit assumption here is that it is highly unlikely that both task executions experience the identical errors and they produce identical erroneous results. If the results are different, an error has occurred during the task execution, and the task is re-executed on another core for error recovery. Let R_{cmp} be the reliability of the result comparison process. Assuming that the two cores are identical, each with a reliability R, the reliability of the DMRR technique can be calculated by Eq. 7.

$$R_{DMRR}(R, R_{cmp}) = R_{cmp}\left(R^2 + 2(1 - R)R^2\right) = R_{cmp}(3R^2 - 2R^3) \tag{7}$$

2.3.3 Triple Modular Redundancy (TMR)

N-Modular redundancy (NMR) that is an $M - of - N$ system with N (an odd number) and $M = (N + 1)/2$ can be applied at the individual core level, where N copies of each task are executed on N different cores in parallel and the results of at least M of them are required to be identical for proper operation. Thus, the task execution fails when the majority voting task finds that fewer than M results are identical [13]. This is similar to the redundant multithreading (RMT) approach if considering architecture-level redundancy management or process level redundancy (PLR) approach if considering operating system-level redundancy management. Here, it is considered that TMR ($N = 3$) is applied at the individual core level, i.e., three copies of each task are executed in parallel on three different cores, and majority voting is performed on the results for error masking. Let R_{vot} be the reliability of the majority voting task. The reliability of TMR can be calculated by Eq. 8.

$$R_{TMR}(R, R_{vot}) = R_{vot}\left(R^3 + 3R^2(1 - R)\right) = R_{vot}\left(3R^2 - 2R^3\right) \tag{8}$$

3 Power–Reliability–Performance Tradeoffs

3.1 *Tradeoffs at the Hardware Level*

Due to technology process variations, the maximum operating frequency and the leakage power consumption vary for different cores in a single chip [3]. Figure 2a illustrates that the core-to-core frequency and leakage power variations in an Intel's 80-core test chip are up to 38 and 47%, respectively [7]. Therefore, regardless of which application is executed, different processing cores present different performance and power consumption.

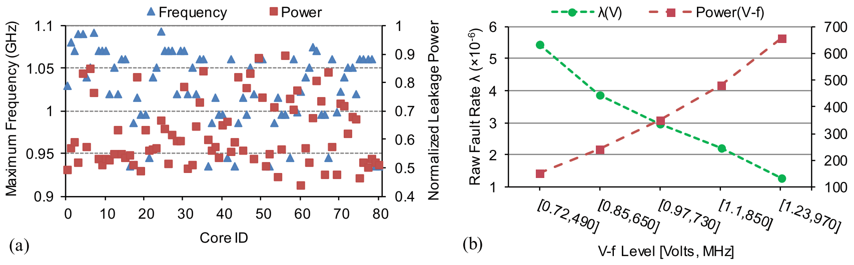


Fig. 2 (**a**) Core-to-core variations in maximum operating frequency and leakage power and (**b**) hardware-level fault rate and power variations at different V-f levels. Adapted from [15]

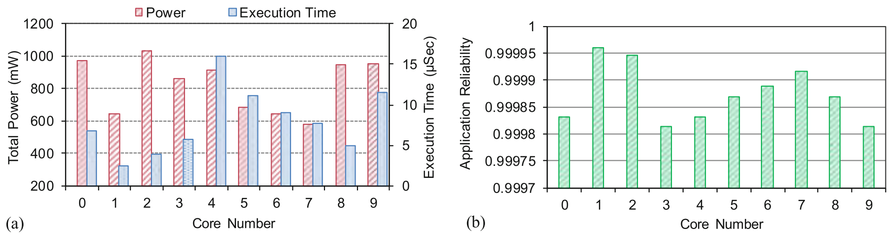


Fig. 3 Core-to-core variations in power, execution time, and reliability when executing the same application. Adapted from [15]

One effective way to reduce power consumption is to decrease the operating V-f level through the DVFS technique. However, based on Eq. 2, decreasing the V-f level increases the hardware-level fault rate. Figure 2b shows that how the total power consumption (Eq. 1) and the hardware-level fault rate (given by Eq. 2) for a given core vary at different V-f levels. For the processor cores in our experiments, it is considered that the V-f level can have five different values as shown in Fig. 2b, i.e., the minimum V-f level is [0.72 V, 490 MHz] and the maximum V-f level is [1.23 V, 970 GHz], see details in Sect. 5.

Due to the core-to-core variations in the frequency and leakage power, when a given application is executed on different cores but under the same supply voltage, it presents different power consumption, performance (execution time), and reliability. Figure 3 shows core power consumption, application reliability, and execution time for the discrete cosine transform (DCT) application when it is executed on different cores but under the same supply voltage. Figure 3a illustrates that due to core-to-core variations in operating frequency, executing a given application on different cores presents different power consumption and performance properties.

According to Eq. 3, the application reliability depends upon the hardware-level fault rate, software vulnerability, and application execution time. Figure 3b illustrates that when a given application is executed on different cores, it provides different reliability levels. This is because, in this case, software vulnerability (FVI) and hardware-level fault rate (λ) remain the same in Eq. 3 (the same application is executed under the same voltage level) but due to core-to-core variations in operating frequency, the application execution time ($\frac{c}{f}$) varies when executed on different cores.

The analyses in Figs. 2 and 3 illustrate that the diversities in power and performance of different cores in a chip when executing the same application along with exploiting different V-f levels can be utilized for efficient reliability management at hardware level.

3.2 Tradeoffs at the Software Level

Since different applications execute different instructions on different operand values, they present different power, performance, and reliability properties even when executed on the same core and under the same V-f level. Figure 4 shows the power consumption, execution time, software vulnerability, and reliability for different applications when executed on the same core and under the same V-f level. Different applications exhibit different circuit switching activity and also require different clock cycles to complete, and hence, they exhibit distinct power consumption and execution time properties even when executed on the same core; see Fig. 4a.

Also, since different instructions present different vulnerabilities to soft errors (e.g., single event upsets), as shown in Fig. 4b, different applications exhibit distinct software vulnerabilities. Figure 4b shows that different applications, even when executed under the same V-f level (i.e., under the same hardware-level fault rate) and on the same core, exhibit different system-wide reliability. This is because, based on Eq. 3, the application reliability also depends upon its software vulnerability and execution time.

The above analysis shows that different applications exhibit different power, performance, and reliability levels when executed on different cores, thus enabling power–reliability–performance tradeoffs at software level.

3.3 Tradeoffs for Hardening Modes

3.3.1 Tradeoffs for Reliability Techniques

Reliability techniques usually employ *different types of redundancy* (e.g., hardware, software, and time redundancy) and *different redundancy levels* (e.g., dual or triple modular redundancy). Therefore, they offer different reliability, performance, and power properties. Also, two different reliability techniques may provide the same error tolerance capability but at different performance and power cost. For example,

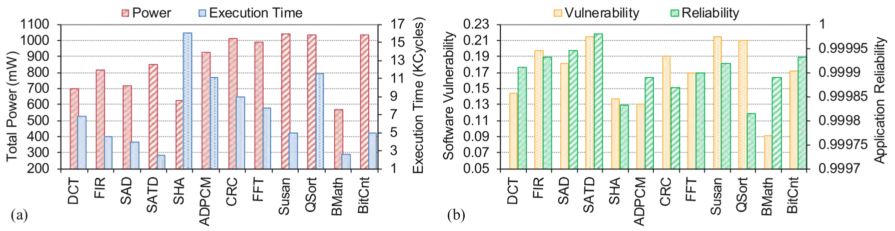


Fig. 4 Application-to-application variations in power, execution time, software vulnerability, and application reliability when executed by the same core. Adapted from [15]

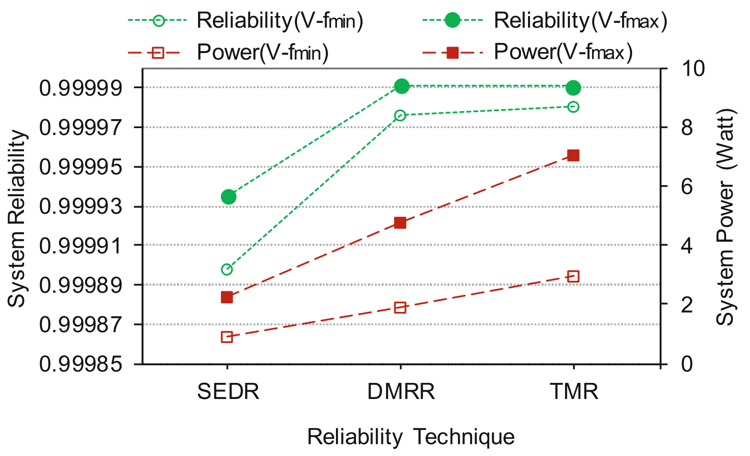


Fig. 5 System-wide reliability and power for different reliability techniques when performed at the minimum and maximum V-f levels (V-f_{min} and V-f_{max}, respectively). Adapted from [12]

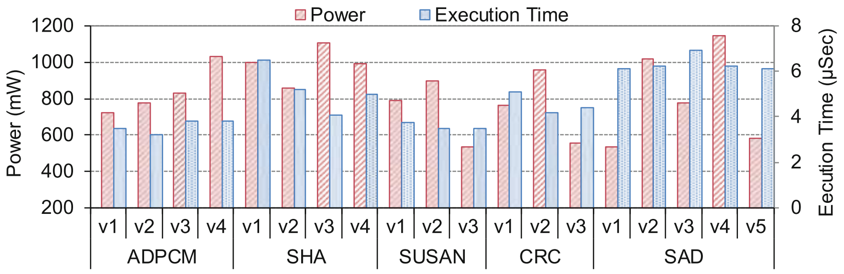
both the DMRR and TMR techniques can tolerate one single task failure; however, when an error occurs, DMRR requires more time to re-execute the task for error recovery, which incurs a performance overhead. Nevertheless, DMRR may consume less power and energy when compared to TMR. This is because when no error occurs, which could be a case for most of the time, DMRR does not require re-executing the task, while TMR always executes the third copy.

Figure 5 shows system reliability and power consumption when the reliability techniques in Sect. 2.3 are employed. To illustrate the effects of scaling the operating V-f level on the system reliability and power consumption, the reliability techniques are executed under the minimum and maximum V-f levels (i.e., V-f_{min} and V-f_{max}). Also, in this figure, reliability techniques with different redundancy levels are considered (i.e., SEDR with a low redundancy level and TMR with a high redundancy level). Figure 5 illustrates that increasing the redundancy level (from SEDR to TMR) and V-f level (from V-f_{min} to V-f_{max}) improves reliability but at the cost of increased power consumption.

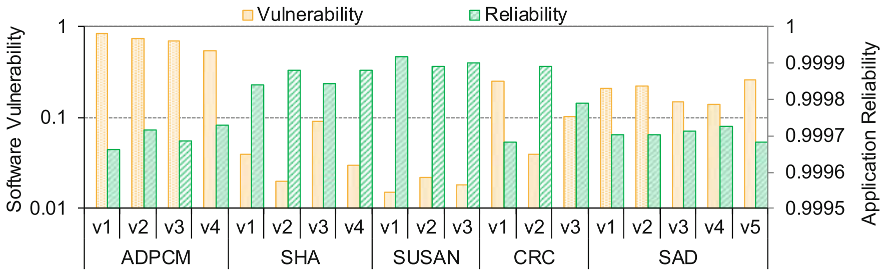
The experiment in Fig. 5 shows that different reliability techniques when operating in different V-f levels exhibit distinct power, performance, and reliability properties, enabling power–reliability–performance tradeoffs that can be employed for power–reliability management.

3.3.2 Tradeoffs for Software Hardening

To further expand the power–reliability–performance optimization space, a reliability-aware compiler can be used to generate multiple reliable compiled code versions for a given application task through reliability-driven software



(a)



(b)

Fig. 6 Different compiled code versions of each application have different: (**a**) Power and performance (execution time); and (**b**) Software vulnerability (in log scale) and application reliability. Adapted from [15]

code transformations (see more details in [8, 9]). Different code versions of the same task present dissimilar power, reliability, and performance properties while implementing the same functionality. For instance, Fig. 6 shows power, execution time (in terms of clock cycles), software vulnerability, and overall reliability (given by Eq. 3) of different compiled code versions for five applications.

The reliability and execution time of an application task also vary with the operating V-f level of the underlying core. Figure 7 shows the reliability and execution time of three code versions for the *ADPCM* application under different V-f levels. Figure 7a illustrates that how different code versions of the *ADPCM* application when executed under different V-f levels can be used to achieve a given reliability requirement for the application (R_{req} in this figure). For instance, to meet the reliability requirement $R_{req} \geq 0.999$, shown by the dotted horizontal line in Fig. 7a, the operating V-f level for the code versions $cv1$ and $cv2$ should be at least [0.97 V, 730 MHz], whereas the V-f level for the code version $cv3$ can be [0.85 V, 650 MHz]. Assume that the application execution has a deadline constraint to finish within 5ms, as shown by the dotted horizontal line in Fig. 7b. In this case, the operating V-f level for the code version $cv2$ should be at least of [0.85 V, 650 MHz], for $cv1$ should be at least [0.97 V, 730 MHz], and for $cv3$ should be at least [1.1 V,

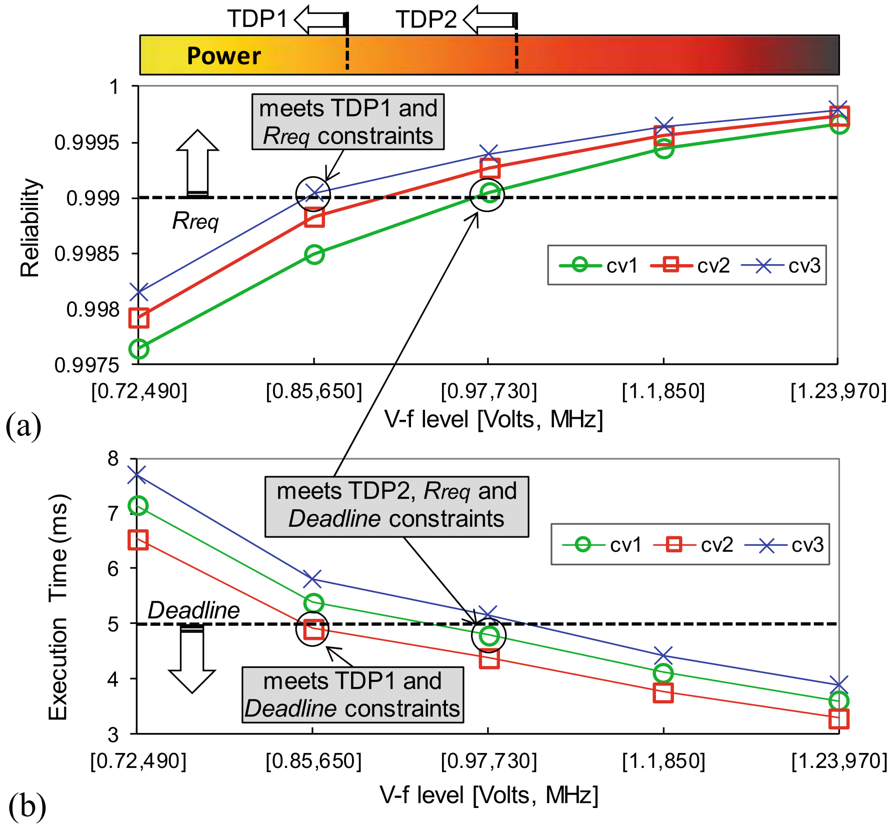


Fig. 7 Reliability and execution time of three compiled code versions for the *ADPCM* application under different voltage–frequency levels. Adapted from [11]

850 MHz]. Now assume that the underlying core has a TDP constraint that requires its operating V-f level should be at most [0.85 V, 650 MHz] (the TDP1 constraint in Fig. 7). Under the TDP1 constraint, to meet the reliability constraint we should select the code version $cv3$; see Fig. 7a. However, under the given TDP1 constraint, if the deadline constraint has to be met, we would select the code version $cv2$; see Fig. 7b. As another example assume that the core has the TDP2 constraint (i.e., the operating V-f level for the core should be at most [0.97 V, 730 MHz]). In this case, the code version $cv1$ is the best choice since it can satisfy both the reliability constraints of $R_{req} \geq 0.999$ and the deadline constraints within 5 ms while meeting the power constraint TDP2.

4 Power–Reliability–Performance Management

From Sects. 2 and 3, the following key observations can be derived that lay the foundation of designing an efficient system for power–reliability–performance Management.

1. Executing tasks at a higher V-f level provides lower execution time and fault rate, resulting in higher system-wide reliability. However, the task power consumption at a high V-f level may be beyond the chip power constraint.
2. An effective way to decrease the power consumption is to lower the operating V-f level, e.g., through DVFS. However, lowering the V-f level leads to an increased execution time of the task that may result in a performance degradation and a missed deadline.
3. Different compiled code versions for an application task exhibit different vulnerability and execution time properties when executed on the same core.
4. Different compiled code versions for each application task when executed by different reliability techniques on different cores with frequency variations and supporting different V-f levels present distinct power, reliability, and performance properties.

In short, the variations in vulnerability and execution time of different compiled versions for each task along with the variations in reliability, power, and performance when using different reliability techniques and V-f levels can be exploited for power–reliability–performance optimization.

The previous works, dynamic redundancy and voltage scaling (DRVS) [12] and dark silicon reliability management (dsReliM) [11], consider the above-mentioned variations at hardware and software levels for run-time power–reliability management. DRVS exploits run-time reliability technique (task-level redundancy) with V-f selection for each application task to minimize system power consumption under reliability and timing (deadline) constraints. dsReliM leverages multiple pre-compiled code versions for each application task with V-f selection at run time to maximize reliability under timing and power constraints. However, such techniques that solely use task-level redundancy or pre-compiled code versions may impose the following restrictions on power–reliability management. Although task-level redundancy can substantially increase reliability, it may increase chip power consumption beyond its power constraint. Also, this technique can only be used if sufficient cores are available for task-level redundancy. In this case, it may be useful to leverage reliable compiled codes to improve reliability, since no extra cores are required to execute different code versions for each application task. Although compile-time software hardening can decrease power consumption, it may increase the execution time of the tasks beyond their timing constraint. Therefore, power–reliability management requires joint considerations of reliability, performance, and power properties of hardening techniques at both software and hardware levels, which is the primary consideration of this chapter.

4.1 Problem Definition

System reliability and total power consumption, V-f level and code version assignments, and task-to-core mapping are represented by different matrices with $n \times m \times c \times v$ elements. Here, n is the number of ready tasks, m is the number of available code versions for each task, c is the number of free cores, and v is the number of available V-f levels for each core. The matrices are:

- $R \in \mathbb{R}^{n \times m \times c \times v}$: A matrix to represent the system reliability. In this matrix, each element $R_{i,j,k,l}$ represents the reliability of the task i when the code version j of the task is executed by the core k under the V-f level l.
- $P \in \mathbb{R}^{n \times m \times c \times v}$: A matrix to represent the system total power consumption. In this matrix, each element $P_{i,j,k,l}$ represents the power consumption for the task i when the code version j of the task is executed by the core k under the V-f level l.
- $X \in \{0,\ 1\}^{n \times m \times c \times v}$: A matrix to represent the code version and V-f level assignments and task-to-core mapping. Code version j for the task i is mapped to the core k and is executed under the V-f level l if and only if $X_{i,j,k,l} = 1$.

Considering power, reliability, and performance as a design object or a design constraint, the potential goals of a power-aware reliable system design can be:

1. Maximize system reliability while keeping total power consumption under a given power constraint (e.g., TDP) and meeting tasks timing requirements (e.g., tasks deadlines) OR
2. Minimize power consumption while satisfying the system reliability and timing requirements.

The power–reliability–performance management problems can be formulated as a constrained 0-1 integer linear program (ILP). In the following, the problem is formulated where reliability is the design objective, while power and performance are the design constraints. That is,

- **Optimization Goal:** Maximizing the system reliability that is defined by the correct execution of all the application tasks.

$$\text{maximize} \prod_{i,j,k,l} X_{i,j,k,l} R_{i,j,k,l} \tag{9}$$

This is a 0–1 assignment problem, and hence, we have

$$X_{i,j,k,l} \in \{0,\ 1\} \tag{10}$$

- **Chip Power Constraint:** Total power consumption of the chip, i.e., the sum of power consumption of all cores should be less than the chip power constraint (i.e., chip-level TDP).

$$\sum_{i,j,k,l} X_{i,j,k,l} P_{i,j,k,l} \leq P_{TDP,chip} \tag{11}$$

- **Cores Power Constraint:** Power consumption of each core should be less than the core power constraint (i.e., core-level TDP).

$$X_{i,j,k,l} P_{i,j,k,l} \leq P_{TDP,k} \tag{12}$$

- **Tasks Timing Constraint:** The execution time $\frac{w_{i,j}}{f_{k,l}}$ of a task i when the code version j of the task (with $w_{i,j}$ clock cycles) is executed on the core k at the V-f level l should satisfy the task timing constraint (defined by the deadline d_i).

$$X_{i,j,k,l} \frac{w_{i,j}}{f_{k,l}} \leq d_i \tag{13}$$

- **Code Version Constraint:** The code version does not change during a task execution, i.e., for each execution of a task only one code version can be used.

$$\forall\, i, k, l \sum_{j} X_{i,j,k,l} = 1 \tag{14}$$

- **V-f Levels Assignment Constraint:** The V-f level does not change during a task execution, i.e., during a task execution the underlying core can only perform under a single V-f level.

$$\forall\, i, j, k \sum_{l} X_{i,j,k,l} = 1 \tag{15}$$

4.2 Proposed Solution

The *power-aware fault-tolerance (PAFT)* technique in this chapter jointly accounts for soft errors, process variations, user defined reliability constraint, and processor power constraint (i.e., TDP). At design time, considering the inherent software-level variations in the execution time of the applications, power, and vulnerability, the PAFT technique selects suitable code versions from multiple compiled codes for each application task (Sect. 4.2.1). At run time, considering the hardware-level variations in performance, fault rate, and power, the PAFT approach selects the hardware/software hardening mode (i.e., reliability technique and code version for each task) and performs task mapping and V-f level allocation (Sect. 4.2.2).

4.2.1 Design-Time Code Selection

As discussed in Sect. 3.3, different compiled code versions for an application task and also different reliability techniques (with different redundancy levels) exhibit different reliability, performance, and power properties. Therefore, for each application task, a tradeoff can be made between two cases:

1. Exploiting a code version with higher reliability and a reliability technique with lower redundancy level (e.g., SEDR) to achieve both high reliability and low power consumption.
2. Exploiting a code version with higher performance and a reliability technique with a higher redundancy level (e.g., TMR) to achieve both high performance and high reliability.

To enable the above tradeoff at run time, we leverage the design-time generated multiple code versions for each task using a reliability-aware compiler (see details in [8, 9]). Then, two types of code versions are chosen as follows (as shown in Fig. 8):

1. **Reliability-Driven Code Selection:** At run time, the reliability of executing a code version of a task on a single core (in the SEDR mode) may be high enough to satisfy the system reliability requirement. In this case, for the task, there is no need to employ a reliability technique with a higher redundancy level. However, the execution time of all the code versions with high reliability, even when executed on a high-performance core and at the maximum V-f level may not be low enough to meet the task deadline constraint. Also, the power consumption of a code version with high reliability may be higher than the processor power budget. Therefore, at design time, for each application task, a set of code versions with high reliability, low execution time, and low power consumption is selected. To do this, first we find the reliability-wise best code versions. Then, from the highly reliable code versions, the performance-wise best code versions (i.e., the code versions with the lowest execution time) are selected. Finally, from the selected code versions, the code versions that provide the lowest power consumption are chosen.
2. **Performance-Driven Code Selection:** The reliability of executing a single task on a single core (in SEDR mode) may not be high enough to satisfy the task reliability requirement. In this case, the task can be executed under a reliability technique with a higher redundancy level (e.g., in the DMRR or TMR mode) to improve its reliability. Here, a code version with a high performance is executed under a high redundancy level to make a balance between timing and functional reliability. Therefore, the performance-wise best code versions with high reliability and low power consumption are chosen.

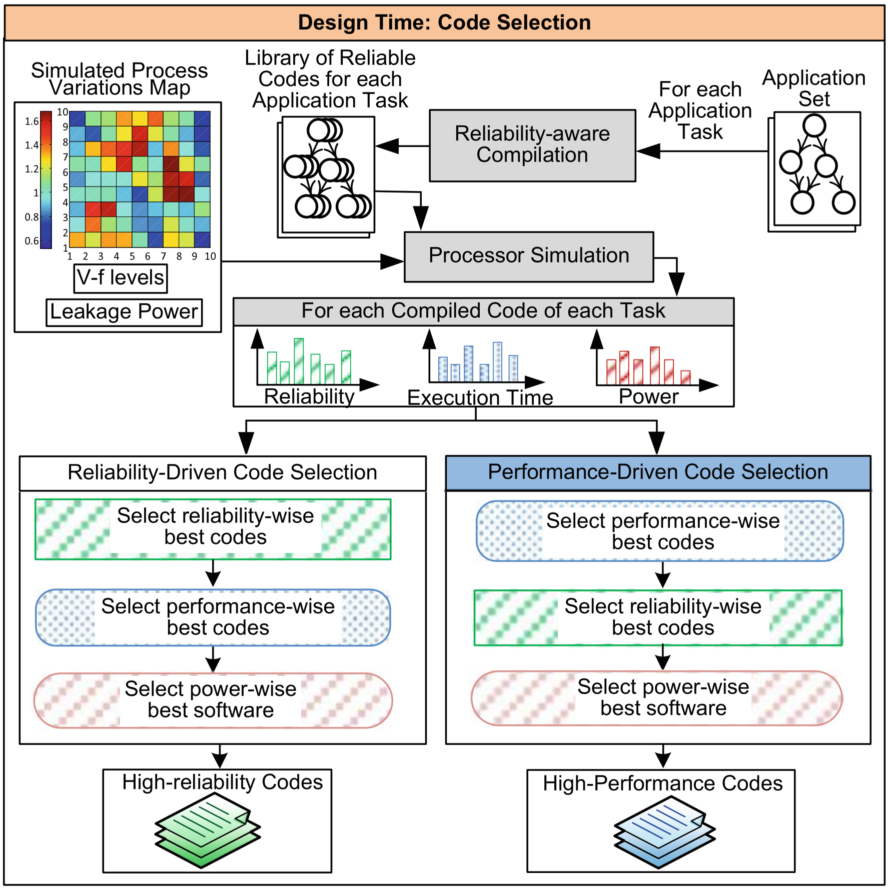

Design Time: Code Selection
Simulated Process Variations Map
V-f levels
Leakage Power
Library of Reliable Codes for each Application Task
Reliability-aware Compilation
For each Application Task
Application Set
Processor Simulation
For each Compiled Code of each Task
Reliability
Execution Time
Power
Reliability-Driven Code Selection
Select reliability-wise best codes
Select performance-wise best codes
Select power-wise best software
Performance-Driven Code Selection
Select performance-wise best codes
Select reliability-wise best codes
Select power-wise best software
High-reliability Codes
High-Performance Codes

Fig. 8 Overview of the design-time part of the power-aware fault-tolerance (PAFT) technique

4.2.2 Run-Time Hardening Mode and V-f Level Selection and Task-to-Core Mapping

The run-time part of the PAFT technique is shown in Fig. 9. It chooses the hardening mode (reliable code version and reliability technique), operating V-f levels and set of cores to implement the reliability techniques, such that the design objectives are achieved while satisfying the design constraints (e.g., maximized system reliability under timing and power constraints). The problem can be effectively solved by the use of existing ILP solvers (formulated in Sect. 4.1). Since 0-1 ILP problems belong to the class of NP-complete problems, ILP solvers generally exploit branch-and-bound mechanisms to find the optimal solution which leads to an exponential increase in their run-time complexity. Therefore, ILP solvers cannot be used in online scenarios where the parameters that are required for decision-making are

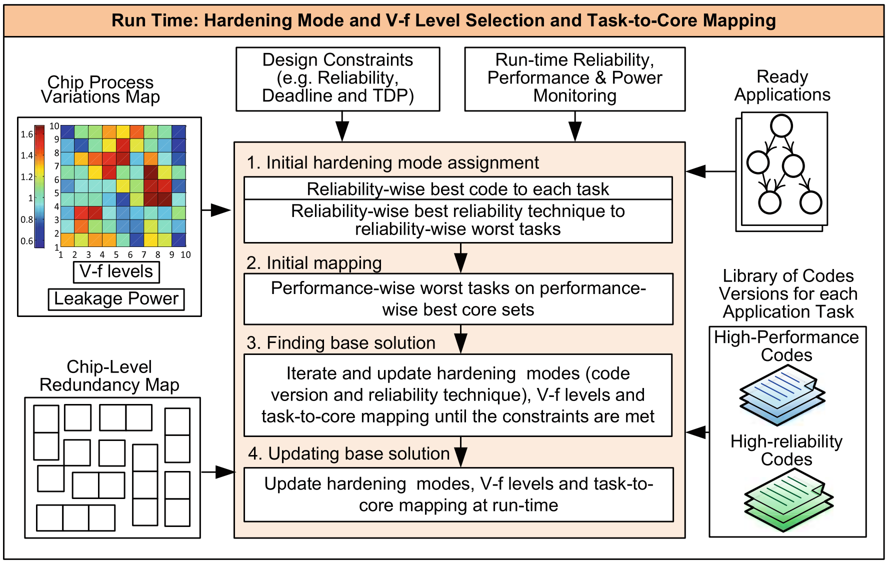

Run Time: Hardening Mode and V-f Level Selection and Task-to-Core Mapping
Chip Process Variations Map
V-f levels
Leakage Power
Chip-Level Redundancy Map
Design Constraints (e.g. Reliability, Deadline and TDP)
Run-time Reliability, Performance & Power Monitoring
Ready Applications
1. Initial hardening mode assignment
Reliability-wise best code to each task
Reliability-wise best reliability technique to reliability-wise worst tasks
2. Initial mapping
Performance-wise worst tasks on performance-wise best core sets
3. Finding base solution
Iterate and update hardening modes (code version and reliability technique), V-f levels and task-to-core mapping until the constraints are met
4. Updating base solution
Update hardening modes, V-f levels and task-to-core mapping at run-time
Library of Codes Versions for each Application Task
High-Performance Codes
High-reliability Codes

Fig. 9 Overview of the run-time part of the power-aware fault-tolerance (PAFT) technique

determined at run time (e.g., ready tasks and free cores). In the case of this problem, the complexity of ILP solvers increases at run time with the number of ready tasks, code versions for each task, reliability techniques, free cores, and V-f levels. Therefore, for this problem, a heuristic is developed, which at first aggressively chooses the hardening mode, operating V-f levels and set of cores in such a way that the highest possible reliability and performance are obtained. Afterwards, it iterates and updates the hardening mode, operating V-f levels and task-to-core mapping until the design constraints (e.g., reliability, deadline, and power constraints) are satisfied. To do this, the run-time part of the PAFT technique gives the chip processor variation map, chip-level redundancy map, design constraints, and library of selected code versions for each application task as input and performs the following four key steps to each ready application:

1. **Initial Hardening Mode Assignment:** First, a reliability-wise best code version is assigned to each task to achieve the highest possible functional reliability for each task. Then, beginning from the task with the lowest reliability, the reliability-wise best technique is assigned to the reliability-wise worst tasks. Therefore, the reliability of the tasks with the lowest reliability is improved, resulting in an improvement in the overall system reliability (the overall system reliability is less than or equal to the reliability of the task with the lowest reliability).
2. **Initial Mapping:** In this step, starting from the task with the highest execution time (lowest performance), the performance-wise worst tasks are mapped on the performance-wise best cores (cores with the highest operating frequency).

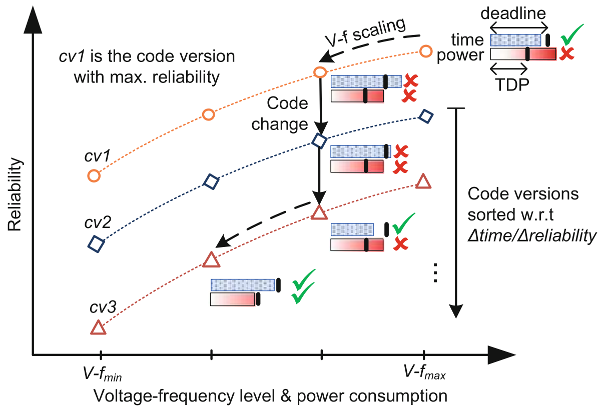
3. **Finding the Base Solution:** Until now, the highest possible reliability and performance have been achieved for the tasks which may be higher than the system performance and reliability requirements. Also, this may lead to an increased chip power consumption beyond its power constraint. In this step, starting from the task with the highest power consumption, the redundancy and V-f level assigned to the task are increased to reduce power consumption until the point that the chip power constraint is satisfied. Here, reducing the operating V-f level may lead to a task deadline miss. In this case, the task code is replaced with a code version with less execution time.
4. **Updating the Base Solution:** At run time, missed deadlines can be considered for performance monitoring and the number of encountered errors can be considered for making the reliability decision. Also, the power information can be acquired from a proxy power monitor. By the use of the run-time reliability, performance, and power monitoring information the system considers the following cases for power–reliability–performance management:

   (a) When the execution of a task finishes, the free cores are employed to improve the system reliability and performance through updating reliability techniques and task-to-core mapping.
   (b) When an error occurs, after tolerating the error by the use of a recovery mechanism, the redundancy and V-f levels assigned to the tasks are increased to compensate the reliability degradation and also to provide high reliability against possible consequent errors.
   (c) When chip power consumption approaches its power constraint, redundancy and V-f levels are decreased to reduce power consumption.

Since estimating the tasks reliability, power, and performance properties is time-consuming, the decisions in steps (a)–(c) are made at run time based on the reliability, power, and performance values that are obtained through the design-time measurements and simulations.

Figure 10 shows how the run-time part of the system works on a ready task. In this figure, for simplicity of the explanations, it is assumed that the reliability technique (e.g., SEDR, DMR, and TMR) and the underlying cores for the task execution are already determined and now the code version and V-f level should be chosen under timing (deadline) and power (TDP) constraints. Suppose that, for the task, the design-time code selection part has selected different pre-compiled code versions ($cv1$, $cv2$, $cv3$, …) with different reliability and execution time properties. At run time, to achieve maximum reliability without considering the deadline and power constraints, the code version with the highest reliability (i.e., $cv1$ in Fig. 10) is selected to be executed at the maximum V-f level (V-f_{max}). In Fig. 10, without loss of generality, it is assumed that the execution time of $cv1$ at V-f_{max} is less than its deadline but its power consumption at V-f_{max} exceeds the chip power (TDP) constraint. In this case, the V-f level is scaled down until the power consumption decreases below the TDP constraint. Suppose the case where reducing the V-f level to a lower level increases the task execution time beyond the task deadline (the task timing constraint is missed). In this case, the code is changed to a version with less

Fig. 10 Code version and V-f level assignment for reliability management under timing (deadline) and power (TDP) constraints


deadline
time
power
TDP
V-f scaling
cv1 is the code version with max. reliability
Code change
cv1
cv2
cv3
Reliability
Code versions sorted w.r.t Δtime/Δreliability
V-f_min
V-f_max
Voltage-frequency level & power consumption

execution time (the code version that can meet the deadline). Here, among the code versions that can meet the deadline, the one is selected that provides the maximum execution time reduction and the minimum reliability loss (i.e., the code version with the maximum $\Delta time/\Delta reliability$). However, if there is no code version that can meet the deadline, to achieve the minimum performance degradation, the one with the minimum execution time is selected. After selecting the suitable code version, the V-f level is scaled down and if needed, the code version is updated until the TDP constraint is met (as shown in Fig. 10).

5 Experimental Setup and Results

Figure 11 shows the experimental setup and evaluation framework. Experiments were conducted by the use of a system-level many-core simulator developed in the C/C++ language. Accurate power and performance (execution time) parameters of applications and underlying hardware were provided for the simulator through processor synthesis, logic simulation, and power estimation. To do this, the Synopsys Design Compiler and a TSMC 45 nm technology file were used to synthesize a VHDL implementation of a LEON3 processor core [2]. Different benchmark applications of *MiBench* [4] (listed in Fig. 4) were used, and multiple compiled code versions for each application task were generated by the use of a reliability-aware compiler of [8, 9]. ModelSim was used for logic simulation to acquire execution time (clock cycles) and activity factors for each compiled code versions of each application. Power estimation was done using the Synopsys Power Compiler with the process synthesis and logic simulation outputs.

As another input for the simulator, the process variation maps were generated through SPICE simulations. The frequency and leakage power variations were modeled through simulating a 13-stage ring oscillator containing $FO4$ inverters based on two-input NAND gates (like in [11, 12, 15]). To implement DVFS for

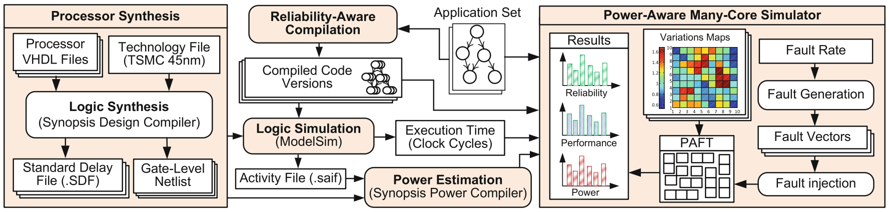

Processor Synthesis
Processor VHDL Files
Technology File (TSMC 45nm)
Logic Synthesis (Synopsis Design Compiler)
Standard Delay File (.SDF)
Gate-Level Netlist
Reliability-Aware Compilation
Application Set
Compiled Code Versions
Logic Simulation (ModelSim)
Execution Time (Clock Cycles)
Activity File (.saif)
Power Estimation (Synopsis Power Compiler)
Power-Aware Many-Core Simulator
Results
Reliability
Performance
Power
Variations Maps
PAFT
Fault Rate
Fault Generation
Fault Vectors
Fault injection

Fig. 11 The experimental setup and simulation flow

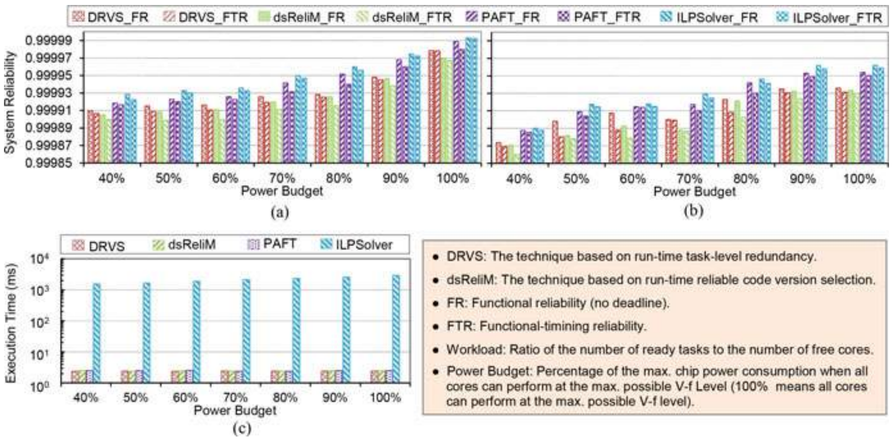
the processor cores, it is considered that the voltage level can change from 0.72 to 1.23 V, with 0.13 V steps, and the corresponding frequency to the minimum and maximum voltage levels is 490 MHz and 970 GHz, respectively.

For reliability evaluations, multiple fault vectors were generated by a Poisson process where the transient fault rate at different V-f levels was modeled based on Eq. 2 with $\lambda_0 = 10^{-4}$ and $\Delta = 1V$. Also, for the system, two types of reliability requirements were considered: (1) *functional reliability (FR)* where only correct output of the tasks is required and the tasks have no deadline and (2) *functional–timing reliability (FTR)* where both correct output of the tasks and meeting tasks deadlines are required. For FTR, for each task, we considered a deadline between its execution time and 1.5× its execution time. Considering the stochastic behavior of transient faults, multiple combinations of benchmark applications were executed for 100,000 times (as a Monte Carlo simulation) and reported the average results.

To model chip power budget, different TDP constraints were considered for each chip between 40 and 100% of its maximum power consumption when all cores perform at their maximum V-f level. This determines a wide range of TDP from a high TDP constraint where up to 40% of the cores can perform at their highest V-f level (i.e., at least 60% dark silicon) to no TDP constraint where all of the cores can perform at their highest V-f level (i.e., 0% dark silicon). Also, two types of system workload were considered in the experiments: (1) *high workload*, when the number of ready tasks is more than 50% of available cores and (2) *low workload*, when the number of ready tasks is less than 50% of the number of available cores.

To evaluate the accuracy and run-time efficiency of the *power-aware fault-tolerance (PAFT)* technique in finding solutions at run time, it was compared with the following techniques:

- dsReliM [11]: which uses compile-time software hardening (different reliable code versions) with run-time code version and V-f level selection.
- DRVS [12]: which exploits run-time task-level redundancy through the SEDR, DMRR, and TMR modes (explained in Sect. 3.3) with V-f level selection.
- ILP Solver: which exploits both compile-time software hardening and run-time task-level redundancy with code version and V-f selection. It searches for the



- DRVS: The technique based on run-time task-level redundancy.
- dsReliM: The technique based on run-time reliable code version selection.
- FR: Functional reliability (no deadline).
- FTR: Functional-timing reliability.
- Workload: Ratio of the number of ready tasks to the number of free cores.
- Power Budget: Percentage of the max. chip power consumption when all cores can perform at the max. possible V-f Level (100% means all cores can perform at the max. possible V-f level).

Fig. 12 Reliability and execution time for different power–reliability management techniques. (**a**) Reliability under low workload. (**b**) Reliability under high workload. (**c**) Execution time. Adapted from [15]

optimum solution through ILP solving. For this system, Gurobi[1] was used as a well-known ILP solving tool.

The results of the reliability and execution time efficiency evaluations are shown in Fig. 12. From this figure, the following observations can be made:

- All four techniques achieve higher reliability when there is no timing constraint for the tasks (denoted by FR in Fig. 12) compared to the case when the tasks have a timing constraint (denoted by FTR in Fig. 12). This is because when tasks have no deadline, a higher task-level redundancy (more task copies) can be considered for the execution of the tasks, resulting in a higher reliability. In addition, highly reliable code versions even with a high execution time can be executed. However, when there are timing constraints only the code versions that can satisfy the timing constraints can be selected. For the same reasons, similar results are obtained for higher system workloads (see Fig. 12b).
- All four techniques provide higher reliability when the chip power budget increases from 40 to 100% of the maximum chip power consumption. This is because more cores can be powered-on and higher redundancy levels can be leveraged for more task executions. In addition, highly reliable code versions even with higher power consumption can be executed.
- From the viewpoint of accuracy, reliability levels provided by PAFT deviate far less than one order of magnitude from the optimum reliability provided by ILP Solver, while the execution time of PAFT is up to 1680× less than the execution

[1] http://www.gurobi.com/.

time of ILP Solver for an 8×8 cores chip (Fig. 12c shows the average execution time for chips with 4×4, 6×6, and 8×8 cores).

- As Fig. 12c shows, the execution time for PAFT is up to 3% higher than the execution time of DRVS and dsReliM, while it achieves at least one order of magnitude more reliability. This is because PAFT leverages both task-level redundancy and code version and V-f selection to discover better tradeoffs between reliability, power, and performance.

6 Conclusion

This chapter presents a power-aware fault-tolerance technique (PAFT) that jointly accounts for transient faults, process variations, and the TDP constraint in multi-/many-core chips. It synergistically exploits different reliability techniques, software hardening modes, and V-f levels at run time for power–reliability management. The problem was modeled as a constrained 0–1 integer linear program (ILP), and a computationally lightweight yet efficient heuristic-based technique for solving the problem was proposed. Results have shown that compared to an ILP solver tool, PAFT deviates far less than one order of magnitude in terms of reliability efficiency while seeding up the reliability management decision time by a factor of up to 1680. PAFT also provides at least one order of magnitude reliability improvement under different TDP constraints when compared to the systems that use either hardware reliability techniques or software hardening modes while increasing the execution time less than 3%.

Acknowledgments This work was supported in parts by the German Research Foundation (DFG) as part of the priority program “Dependable Embedded Systems” (SPP 1500—spp1500.itec.kit.edu).

References

1. Brooks, D.M., Dick, R.P., Joseph, R., Shang, L.: Power, thermal, and reliability modeling in nanometer-scale microprocessors. IEEE Micro **27**(3), 49–62 (2007). https://doi.org/10.1109/MM.2007.58
2. Gaisler, C.: LEON3 processor. http://www.gaisler.com/index.php/products/processors/leon3. Accessed 07 June 2019
3. Gupta, P., Agarwal, Y., Dolecek, L., Dutt, N.D., Gupta, R.K., Kumar, R., Mitra, S., Nicolau, A., Rosing, T.S., Srivastava, M.B., Swanson, S., Sylvester, D.: Underdesigned and opportunistic computing in presence of hardware variability. IEEE Trans. CAD Integr. Circuits Syst. **32**(1), 8–23 (2013). https://doi.org/10.1109/TCAD.2012.2223467
4. Guthaus, M.R., Ringenberg, J.S., Ernst, D., Austin, T.M., Mudge, T., Brown, R.B.: MiBench: a free, commercially representative embedded benchmark suite. In: Proceedings of the Workload Characterization, IEEE International Workshop, WWC’01, pp. 3–14. IEEE Computer Society, Washington (2001). https://doi.org/10.1109/WWC.2001.15

5. Henkel, J., Bauer, L., Becker, J., Bringmann, O., Brinkschulte, U., Chakraborty, S., Engel, M., Ernst, R., Härtig, H., Hedrich, L., Herkersdorf, A., Kapitza, R., Lohmann, D., Marwedel, P., Platzner, M., Rosenstiel, W., Schlichtmann, U., Spinczyk, O., Tahoori, M.B., Teich, J., Wehn, N., Wunderlich, H.: Design and architectures for dependable embedded systems. In: Proceedings of the 9th International Conference on Hardware/Software Codesign and System Synthesis, CODES+ISSS 2011, Part of ESWeek '11 Seventh Embedded Systems Week, Taipei, 9–14 October, 2011, pp. 69–78 (2011). https://doi.org/10.1145/2039370.2039384
6. Henkel, J., Bauer, L., Zhang, H., Rehman, S., Shafique, M.: Multi-layer dependability: from microarchitecture to application level. In: The 51st Annual Design Automation Conference 2014, DAC'14, San Francisco, June 1–5, 2014, pp. 47:1–47:6 (2014). https://doi.org/10.1145/2593069.2596683
7. Rangan, K.K., Powell, M.D., Wei, G., Brooks, D.M.: Achieving uniform performance and maximizing throughput in the presence of heterogeneity. In: 17th International Conference on High-Performance Computer Architecture (HPCA-17 2011), February 12–16 2011, San Antonio, pp. 3–14 (2011). https://doi.org/10.1109/HPCA.2011.5749712
8. Rehman, S., Shafique, M., Kriebel, F., Henkel, J.: Reliable software for unreliable hardware: embedded code generation aiming at reliability. In: Proceedings of the 9th International Conference on Hardware/Software Codesign and System Synthesis, CODES+ISSS 2011, part of ESWeek'11 Seventh Embedded Systems Week, Taipei, 9–14 October, 2011, pp. 237–246 (2011). https://doi.org/10.1145/2039370.2039408
9. Rehman, S., Kriebel, F., Shafique, M., Henkel, J.: Reliability-driven software transformations for unreliable hardware. IEEE Trans. CAD Integr. Circuits Syst. **33**(11), 1597–1610 (2014). https://doi.org/10.1109/TCAD.2014.2341894
10. Salehi, M., Ejlali, A.: A hardware platform for evaluating low-energy multiprocessor embedded systems based on COTS devices. IEEE Trans. Ind. Electron. **62**(2), 1262–1269 (2015). https://doi.org/10.1109/TIE.2014.2352215
11. Salehi, M., Shafique, M., Kriebel, F., Rehman, S., Tavana, M.K., Ejlali, A., Henkel, J.: dsRelim: power-constrained reliability management in dark-silicon many-core chips under process variations. In: 2015 International Conference on Hardware/Software Codesign and System Synthesis, CODES+ISSS 2015, Amsterdam, October 4–9, 2015, pp. 75–82 (2015). https://doi.org/10.1109/CODESISSS.2015.7331370
12. Salehi, M., Tavana, M.K., Rehman, S., Kriebel, F., Shafique, M., Ejlali, A., Henkel, J.: DRVS: power-efficient reliability management through dynamic redundancy and voltage scaling under variations. In: IEEE/ACM International Symposium on Low Power Electronics and Design, ISLPED 2015, Rome, July 22–24, 2015, pp. 225–230 (2015). https://doi.org/10.1109/ISLPED.2015.7273518
13. Salehi, M., Ejlali, A., Al-Hashimi, B.M.: Two-phase low-energy n-modular redundancy for hard real-time multi-core systems. IEEE Trans. Parallel Distrib. Syst. **27**(5), 1497–1510 (2016). https://doi.org/10.1109/TPDS.2015.2444402
14. Salehi, M., Tavana, M.K., Rehman, S., Shafique, M., Ejlali, A., Henkel, J.: Two-state checkpointing for energy-efficient fault tolerance in hard real-time systems. IEEE Trans. VLSI Syst. **24**(7), 2426–2437 (2016). https://doi.org/10.1109/TVLSI.2015.2512839
15. Salehi, M., Ejlali, A., Shafique, M.: Run-time adaptive power-aware reliability management for manycores. IEEE Des. Test **35**(5), 36–44 (2018). https://doi.org/10.1109/MDAT.2017.2775738
16. Shafique, M., Garg, S., Henkel, J., Marculescu, D.: The EDA challenges in the dark silicon era: temperature, reliability, and variability perspectives. In: The 51st Annual Design Automation Conference 2014, DAC'14, San Francisco, June 1–5, 2014, pp. 185:1–185:6 (2014). https://doi.org/10.1145/2593069.2593229
17. Shye, A., Moseley, T., Reddi, V.J., Blomstedt, J., Connors, D.A.: Using process-level redundancy to exploit multiple cores for transient fault tolerance. In: Proceedings of the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2007, 25–28 June 2007, Edinburgh, pp. 297–306 (2007). https://doi.org/10.1109/DSN.2007.98

Our Perspectives

Jian-Jia Chen and Joerg Henkel

Research and development in the last decades have led to a silicon process that has been expected to become inherently undependable in the near future when migrating towards new technologies. The special priority program (SPP) 1500 funded by the German Research Foundation (Deutsche Forschungsgemeinschaft, DFG) in 2010–2016 and the Variability Expedition funded by the National Science Foundation (NSF) in 2010–2015 made a joint effort to explore design challenges of *Power Consumption*, *Reliability*, *Interference*, and *Manufacturability* under such a design requirement.

The exploration started with a vision to go beyond simply developing fault-tolerant systems that monitor the device at run-time and react to error detection. Instead, the design should consider error as a design constraint and develop methodologies to achieve *resilience* at the presence of errors. Under such a design principle, error is inevitable and the error rate should be a tradeoff against performance.

This book summarizes the achievements of the SPP 1500 partners, the Variability Expedition partners, and their collaborators. After telling the successful stories in the previous chapters, this chapter provides a summary of our perspectives of the exploration and a short outlook of future.

One important perspective to achieve resilience at the presence of faults is to *quantitatively* define resilience and errors and use the resilience in a *cross-layer* manner. Specifically, the RAP model summarized in chapter "RAP Model–Enabling Cross-Layer Analysis and Optimization for System-on-Chip Resilience" provides a milestone to help annotate how variability related to physical faults can be expressed

J.-J. Chen
TU Dortmund, Dortmund, Germany
e-mail: jian-jia.chen@tu-dortmund.de

J. Henkel (✉)
Karlsruhe Institute of Technology, Karlsruhe, Germany
e-mail: henkel@kit.edu

J. Henkel, N. Dutt (eds.), *Dependable Embedded Systems*, Embedded Systems,
https://doi.org/10.1007/978-3-030-52017-5_25

at higher abstraction levels. RAP is a result of several working group meetings and collaborative efforts among SPP 1500 partners. It has been also used as a demonstrator in several projects. We believe that RAP is an initial step towards good abstractions that can be used to model the faulty hardware and its impact on the software. It may be possible that the probabilistic information encoded in RAP is not precise enough for further optimizations. We envision that a more flexible and more precise model to correctly quantify the resilience will be needed in the future. It may be a set of models that can be configured depending on the required accuracy level.

One possible way to analyze system-level resilience in a modularized manner is to enable compositional reliability analysis. The support of composition and decomposition is important for modularized analysis and can be used to model uncertainties in both functional and non-functional properties. The modulability provided in chapter "EM Lifetime Constrained Optimization for Multi-Segment Power Grid Networks" extends the existing compositional performance analysis (CPA) (or real-time calculus) to handle reliability. We believe that there is a great potential to utilize the concept in the system design. However, to achieve composition and decomposition, rules to bound the approximation errors of composition and/or decomposition would be needed. The automatic design of efficient and effective rules is essential for compositional reliability analysis.

In many of the results of the SPP 1500 and Variability Expedition partners, cross-layer and interactive optimization has been explored. Unlike the classic multi-layered approach, in which each layer *passively* takes the input from the higher/lower layers, the cross-layer approach applies *active* optimization routines across multiple layers. Since the system-level resilience cannot be optimized unless all the layers are optimized, such a cross-layer approach has been used as interfaces between different layers. An overview of the (coarse-grained) layers and their interactions can be found in Fig. 10 in chapter "RAP Model–Enabling Cross-Layer Analysis and Optimization for System-on-Chip Resilience".

To validate the research results, *fault injection* through instrumentation, emulation, and simulation has been developed and used. Fault injection is an important routine that should be deployed before fault detection. The computational effort of fault injection can become a bottleneck. Proper models and tools for fault injections are important contributions of the research partners. For example, the FPGA fault injection tool in chapter "Dependability Aspects in Configurable Embedded Operating Systems" can be used to emulate the entire SoC with specific faults. Different fault injection scenarios can be found in chapter "Lightweight Software-Defined Error Correction for Memories". Despite its importance, to the best of our knowledge, there is no integrated tool that can be used for benchmarking the quality and (intended) consequence of fault injection. Although there have been several attempts to provide an integrated tool from the partners in SPP1500 for different types of fault injection, the diverse scenarios in the cross-layer settings made the integration very difficult. We envision that fault injection tools that can be configured and applied for different layers and scenarios can be developed in the near future so that cross-layer design and optimization can be further modularized and deployed.

When the design considers error as a design constraint, the system has to be *adaptive* to react (or even be proactive) according to the faults and errors to achieve the targeted resilience. Adaptive methods in physical, micro-architecture, architecture (ISA), compiler, and operating systems are explored and discussed. It has been demonstrated in several research results that adaptivity should be applied across layers. For example, the error semantics in chapter "Soft Error Handling for Embedded Systems using Compiler-OS Interaction" in the software development process provides the information in the compilation needed for the operating systems to be adaptive according to faults. Moreover, the annotation of multiple execution versions in chapter "Cross-Layer Dependability: From Architecture to Software and Operating System" provides a means to the run-time system to execute different versions according to the reliability condition. Furthermore, the dependability aspects can be further configured in operating systems as demonstrated in chapter "ASTEROID and the Replica-Aware Co-scheduling for Mixed-Criticality". We strongly believe that adaptivity is a key insight. However, the reported achievement is based on ad hoc treatments for well-defined scenarios. It will be very practical and impactful to explore automatic adaptivity so that suggestions of proper means can be provided to the designers for achieving high resilience.

The adaptive handling of errors and faults naturally makes the timing behavior dynamic over time. When there is no fault, an embedded system functions correctly with respect to the specified timing. However, when there are faults, the embedded system may not function correctly anymore since some jobs may be aborted or may miss their deadlines. Therefore, it is of importance to explore both functional and timing correctness. If all jobs have to meet their deadlines, the hardware may have to be over-dimensioned. If some jobs can be allowed to miss their deadlines when faults are present, the system designer just has to ensure that all the desired timing behavior can be verified offline. Such dynamic timing requirements can be modeled as mixed criticality. When the system does not suffer from any fault, it is at the low-criticality mode. When the system suffers from some faults, it is promoted to the high-critical mode. Such a treatment has been presented in chapter "Dependability Aspects in Configurable Embedded Operating Systems". An alternative is to explore the probability (or miss rate) of deadline misses, presented in chapter "Cross-Layer Dependability: From Architecture to Software and Operating System". Although the above treatments are successful, they are not originally from the resilience perspectives. It remains open whether the timing requirements to achieve system resilience should be treated as the first-class design objective. More specifically, although dynamic timing behavior and requirements are considered, they are not directly related to resilience. Moreover, the tradeoffs of the timing requirement and system resilience in the presence of faults are still in the infant stage and require more research efforts to reach a conclusion.

We believe that the successful stories in the previous chapters and the perspectives presented in this chapter provide cornerstones for the design of dependable systems on unreliable hardware. Based on the foundation established by the partners, designs which consider faults/errors as a design constraint will be continued in different directions, including physical, micro-architecture, architecture (ISA), compiler, and operating systems layers, and, *most importantly*, in a **cross-layer** manner.

Index

J. Henkel, N. Dutt (eds.), *Dependable Embedded Systems*, Embedded Systems,
https://doi.org/10.1007/978-3-030-52017-5