Evaluating the Impact of Aging on Path-Delay Self-Test Libraries

Evaluating the Impact of Aging on Path-Delay Self-Test Libraries

Riccardo Cantoro, Sandro Sartoni and Matteo Sonza Reorda
Politecnico di Torino
{riccardo.cantoro, sandro.sartoni, matteo.sonzareorda}@polito.it

Lorena Anghel*, Michele Portolan
Univ Grenoble Alpes, CNRS, Grenoble INP,
TIMA, *CEA-IRIG, SPINTEC Laboratory, France

***Abstract*—Self-Test Libraries (STLs) developed for path-delay faults are crucial to ensure the reliability of modern digital integrated circuits, since they represent a widely adopted solution to detect in-field faults occurring during the operational phase, e.g., due to aging. However, physical parameters may shift over time, leading to changes in device behavior and potential failures with respect to end of manufacturing. This is especially crucial in safety-critical applications such as those adopted in automotive systems. The main objective of this paper is to assess the quality of STLs over time by monitoring how critical paths change due to aging effects and evaluating the fault coverage of STLs. An automatic framework is proposed to age an integrated circuit starting from a limited set of physical data related to the adopted technology. The insights gained from this approach will allow test engineers to harden STLs and ensure that strict reliability requirements are always met.**

***Index Terms*—aging, sbst, path delay faults**

I. Introduction

Modern digital integrated circuits are designed and manufactured leveraging on advanced semiconductor technology and nodes. An empiric law introduced by Gordon Moore describes how, at a first approximation, the number of transistors found on microchips doubles every two years circa. Increasing the number of transistors on an integrated circuit not only greatly affects the physical size and geometry of each transistor, but it also has repercussions on the power supply at which the circuit operates and on its parasitic capacitance and resistance, thus impacting the power consumption and operating frequency of the integrated circuit. While this enabled the production of VLSI circuits that are capable of operating at high frequencies with limited power consumption, it also impacted the reliability of said devices. Physical phenomenon that were negligible on larger technology nodes became more and more relevant with newer technologies, and parameters shift in time influences the behavior of the device as it ages, possibly leading to its failure as the circuit is stressed over time. Given that many fields require safety-critical devices to correctly behave over a large period of time (e.g., the automotive field where replacing a faulty device many times over the operative lifetime of the vehicle is not feasible), it is crucial to tackle the issue of ensuring the safety and reliability of integrated circuits as they age.

Indeed, there are several works in the literature that focus on the study of aging in circuits and how it affects their timing behavior [1]–[6]. Aging and performance degradation of an integrated circuit is the consequence of several physical phenomena that occur concurrently, impacting the device in different ways. As integrated circuits age, the propagation delay of each cell in the design increases, with potentially severe repercussions on the timing behavior of the whole device. Delay fault models are well suited to study such issues, particularly the path delay fault model. Path delay fault testing is typically performed by means of Design-for-Test (DFT) based techniques or functional techniques. A DFT approach leverage the capabilities of the test hardware that is added to the original design under test (DUT), e.g., scan chains, to apply test vectors generated by means of Automatic Test Patter Generators (ATPGs) to the circuit to be tested as well as to monitor its responses, looking for potential errors [7]–[9]. DFT solutions are mature and allow achieving high fault coverage figures at the expense of additional hardware that may cause timing and area overhead, together with test procedures that may (i) have higher power consumption as the circuit internal nodes toggle in a non-functional way, and (ii) require a non-negligible amount of time to execute. Functional solutions, usually in the form of Software-Based Self-Test (SBST) [10], revolve around the development of Self-Test Libraries (STLs) that are launched to test the DUT, usually a processor core or a peripheral [11]–[17]. As the STL is executed, its instructions apply to the circuit the test vectors that are required to test eventual path delay faults within the circuit, and the DUT responses to such vectors are compacted into signatures that are compared against the golden circuit's ones. SBST solutions are cheap, reliable and time-effective, as they do not require additional hardware and are applied at the functional clock frequency, i.e., at the circuit's nominal speed, a crucial feature when tackling delay faults. All these properties make SBST a suitable integration for already existing DFT testing schemes, allowing to ensure the DUT's reliability at all times.

Regardless of the approach, all testing solutions for path delay faults focus on faults that stem out of the set of critical paths that are found in the fresh circuit. However, aging impacts the timing behavior of the device, and paths that may be sub-critical, i.e., paths whose slack is non-negligible but not large enough to be critical paths, at time zero, may become critical in time. This can lead to a degradation in fault coverage figures over time, as well as an overall reduced safety and reliability of the device.

979-8-3503-1500-4/23/$31.00 ©2023 IEEE

This work presents an evaluation flow that allows to easily assess the impact of aging by generating lists of aged critical paths, requiring a limited set of input data, starting from an aging model defined in previous works in literature. Thanks to an automatic aging tool, it is possible to estimate the effectiveness of testing solutions such as based on STLs, over time, also providing test engineers a metric on how STLs for path delay faults should be developed so that satisfactory fault coverage figures are ensured throughout the operative lifetime of the device. The novelty of this work can be summarized into three main points:

1) an automatic tool that is capable of aging an integrated circuit with a limited amount of input data, which can be used in conjunction with any commercial synthesis and static timing analysis tool, thus making it easily reusable;
2) a flow that allows to generate a list of critical paths given an aged circuit and to compare the list of critical paths of the aged circuit with respect to those found in the fresh circuit;
3) a set of strategies that ensure that a high path delay fault coverage is achieved throughout the operative lifetime of the device under test.

This approach is validated on an open-source RISC-V core, synthesized with a proprietary FDSOI 28nm library. Experimental results provide insights on how critical paths change over time, and fault coverage figures accordingly.

The article is organized as follows: Section II provides extensive details on the physical phenomena that cause aging, as well as mathematical models for delay degradation. Section III presents the automatic aging tool that allows to obtain a list of aged critical paths, describing in details each step of the aging process. Section IV provides details on the DUT used to validate the proposed approach, while Section V presents data on the aged critical paths and how they affect the path delay fault coverage. Finally, Section VI draws the conclusions.

II. Background

Several phenomena concur to aging a device, with *Bias Temperature Instability* (BTI), *Hot Carrier Injection* (HCI), and *Time-Dependent Dielectric Breakdown* (TDDB) being among the main causes for degradation over time in modern integrated circuits. *BTI* is a destructive phenomenon that can be further divided into Negative Bias Temperature Instability (NBTI) [18] and Positive Bias Temperature Instability (PBTI). The former affects PMOS transistors mostly, and it induces a more significant degradation with respect to the latter, which affects NMOS transistors mostly. *HCI* occurs whenever an electron or a hole gains sufficient kinetic energy to overcome the potential barrier, thus penetrating the dielectric oxide layer that is found below the gate plane in a MOSFET [19]–[21]. As transistor size scales, the voltage at which they operate does not scale accordingly, which causes large electrical fields inside the device. The larger such fields, the higher the probability a hot carrier, may it be electron or hole, is injected in dielectric films, thus degrading the device over prolonged periods and impacting its physical properties. Finally, *TDDB* affects the dielectric film of transistors, and it is described as a change in properties of the dielectric due to the presence of electric fields, shifting from a material with insulating properties to one with more conductive features [22]. This affects leakage currents when the device is supposed to be off and threshold voltage too, and it is a major reliability issue in MOSFETS [23].

In order to tackle these issues, researchers have worked on producing a mathematical model that could predict the shift in propagation delay of any given cell in a circuit. Such model can be obtained by integrating formulas described in literature and actual data gathered on logic gates implemented on silicon wafers. A model for delay estimation on MOSFET devices, also known as the alpha-power law, was first introduced and described in [1]. Such a model states that the propagation delay is proportional to the output capacitance the cell is subjected to, times the voltage supply at which it operates over its drain current as follows:

$$d_{cell} \propto C_{out} \frac{V}{I_d} \tag{1}$$

Starting from that model, further works have refined the alpha-power law leading to an equation for evaluating the propagation delay of an inverter cell with respect to time, supply voltage, and operating temperature [5], [6], [24]:

$$d_{inv}(V,T,t) = p_\beta + \frac{p_{\mu^{-1}}(T) \cdot V}{(V - (p_{V_{th}}(T) + \Delta p_{V_{th}}(V,T,t)))^{p_\alpha}} \tag{2}$$

where p_β and p_α are constants, while $p_{\mu^{-1}}(T)$ is related to the transistors mobility and $p_{V_{th}}(T)$, $\Delta p_{V_{th}}(V,T,t)$ are factors related to the threshold voltage, the latter describing a shift in threshold voltage depending, among the other parameters, to time.

Eq. (2), although effective, has some important limitations. First, it only describes the delay of an inverter gate, while any integrated circuit worth discussing has several different gates within its design. Second, it does not factor in the effect of *switching activity*, i.e., how much the cell toggles throughout the device's operative lifetime. Aging is the result of several physical phenomena, and while not every phenomenon strictly depends on the switching activity, e.g., NBTI, others show a direct dependence on such parameter, e.g., HCI, hence why it is important to include it into the aging model. For this reason, the works in [6], [24] also provide a way to generalize Eq. (2) by adding two more terms: the *logical effort* [25] — a parameter that allows to calculate the delay of an arbitrary cell based on that cell and the inverter's physical parameters — and a term based on switching activity, first introduced in [26]. This leads to the final delay formula:

$$d_{gate}(V,T,t,SA) = d_{inv}(V,T,t) \cdot d(SA) \cdot LE_{gate} \tag{3}$$

where $d_{inv}(V,T,t)$ is Eq. (2), $d(SA)$ is the switching activity contribution and LE_{gate} is the logical effort of a generic gate.

Through Eq. (3), it is finally possible to estimate how the delay of a gate degrades in time with respect to voltage, temperature, time and switching activity, closely emulating what happens in an actual circuit implemented on silicon. Although effective, this approach needs large quantities of data that can only be gathered by means of experiments on devices implemented on silicon. To deal with this issue, the work in [6] proposes a multiple linear regression algorithm through which it is possible to create a continuous function [27], i.e., the delay of a gate cell, in the form of:

$$y = \alpha + \beta_1 \cdot x_1 + \beta_2 \cdot x_2 + ... + \beta_n \cdot x_n \quad (4)$$

where y is the delay of a given gate cell, α is the y-intercept value, (x_1, x_2, ..., x_n) is the set of gate features (voltage, time, switching activity, temperature), and (β_1, β_2, ..., β_n) the coefficients of gate parameters. The idea behind this approach is that, once the parameters are tuned so that the error sum of squared errors (SSE) between observed and predicted results is minimized [28], such a model can be used to generate reliable delay values for aged cells, thus aging the whole circuit.

III. Proposed Evaluation Framework

The automatic aging framework proposed in this article aims at providing a unified and automatic method that is capable of generating aged delays for each cell in a circuit starting from data generated by commercial tools currently used in research and industry. Part of the framework implements the studies on aging evaluation conducted in Grenoble INP [5], [6], [24]. At its core, the proposed aging framework implements the set of equations and linear regression model presented in [6], and provides a wrapper that acts as an interface capable of parsing input information and generating output information seamlessly. However, the proposed aging framework can be easily adapted so that it employs other aging models, e.g., for industrial applications. As a result, we implemented an automatic aging tool, whose structure and the preliminary steps that are required for it to function correctly are summarized in Fig. 1.

The very first step that is required for this tool to work is performing a synthesis of the device under test with a technology library for which physical parameters are known or available. This is an important aspect, as the whole mathematical model on top of which this approach is built requires that these parameters are known for calculating the propagation delay for each cell. No constraint is placed on the synthesis process, thus leaving the test engineer full freedom in customizing such step as required. The synthesis step is required to obtain information on the propagation delay of each cell of design at time zero, also referred to as *fresh delay information*, which will constitute the starting point for calculating the aged delay. Usually, such an information is automatically generated at the end of the synthesis process in the form of a Standard Delay Format (SDF) file, which reports information on the cell name — i.e., the name of the cell found in the synthesized circuit — the cell type — i.e., whether it is an AND, INV, OR gate — and slow, typical and fast propagation delays from each

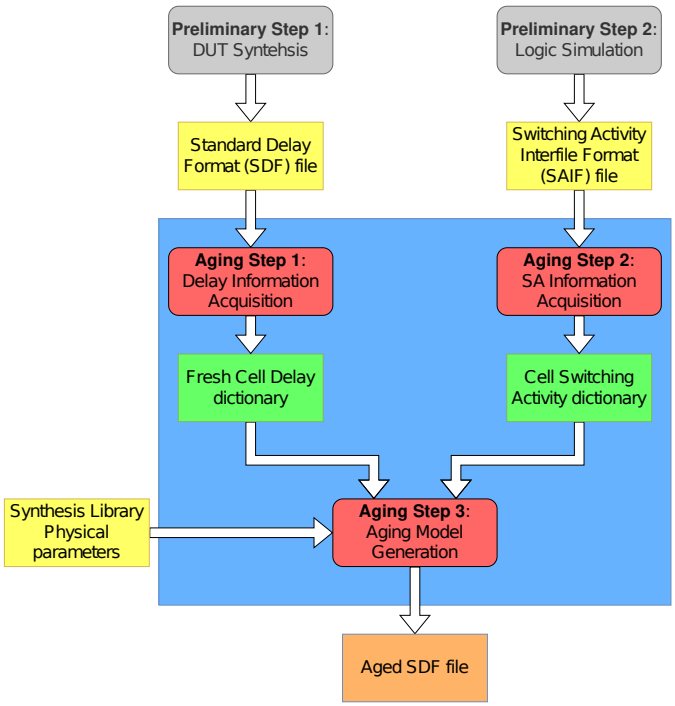

Fig. 1. Automatic Aging Tool flow diagram

input to each output, as well as the logic values of the other inputs. The second preliminary step is the execution of a logic simulation where the device under test performs its tasks and activities, as if it was embedded in the environment where it is supposed to operate throughout its operative lifetime. The reason for recreating this scenario lies in the fact that aging depends, among other factors, on how much the circuit is stressed, i.e., on how much its internal nodes toggle, which can be easily tracked by evaluating the switching activity on each cell. Two same circuits, if applied in different scenarios, may show different aging patterns over time, hence why this step is required. While the logic simulation unfolds, a Value Change Dump (VCD) file is recorded, storing information on the value stored by every cell at each time instant. Such file is then converted into a Switching Activity Interfile Format (SAIF) file, which reports for each cell how many toggles have occurred.

Once the preliminary steps are cleared, the tool can proceed with the aging process. Initially, the tool parses the information stored in the SDF and SAIF files so that it can use that data as a starting point to calculate the aging delay. First, the SDF file is read, transposing the data stored into the SDF into a fresh cell delay dictionary where each instance name is associated to the cell type and the delay data, a sub-dictionary that maps conditions on input ports to two triples of slow, typical and fast delays for rising and falling transitions from an input to the output. In a similar fashion, data from the SAIF file too is parsed into a cell switching activity dictionary, where for each cell in the design the correspondent switching activity is recorded. As the switching activity value ranges in the interval $[0, 1]$, rather than using the absolute value reported in the SAIF

file, for each cell the tool calculates the normalized switching activity value with respect to a reference signal, e.g., the clock signal.

Next, the tool proceeds to generate an aging model for each cell in the design based on the mathematical equations introduced in Section II and the switching activity information stored in the related dictionary. Such models are then used to calculate the relative increment in delay (RID) for each cell, defined as:

$$RID = 1 + (AD - FD)/FD \tag{5}$$

where AD is the aged delay, and FD is the fresh delay obtained by applying Eq. (3). The RID is multiplied to the fresh delay reported in the SDF file so that the final aged propagation delay for each cell is obtained. Finally, the tool writes an aged version of the SDF file so that it can be used by timing analysis tools to generate a list of aged critical paths. The reason for introducing and using the relative increment in delay rather than the aged delay generated through the mathematical formulas itself lies in the fact that in this way it is possible to achieve more accurate results in predicting the aged delay. Even though the mathematical model is accurate, errors are introduced by formulas as they are obtained by fitting processes over experimental data. Such errors, however, are mitigated by using the relative values rather than the absolute ones.

Finally, it is worth highlighting that, even though the equations used in the construction of the aging model for each cell are generic and thus can be used with any technology, the values of the physical parameters that are used in such equations strictly depend on the technology on top of which a library of cell gates is defined. This implies that each library produced by different manufacturers, or even two libraries defined with different technologies belonging to the same manufacturer, will have its own set of parameters, thus leading to different aging models and results.

IV. Case Study

In order to analyze how aging affects a processor core, the proposed evaluation flow has been validated using the PULPino SoC [29], which includes an in-order, pipelined 32-bit RISC-V processor, i.e., ri5cy. The ri5cy core was synthesized with a proprietary FDSOI 28nm library provided by STMicroelectronics, the same library that has also been used in [6].

The initial set of critical paths was extracted first, obtaining a total of 6,683 critical paths with a slack ranging from 0 to 2.5 ns, while the clock signal period is 5 ns. We chose to group critical paths in three main functional groups, namely, paths belonging to an adder found within the divider unit of the ALU, those belonging to an adder belonging to the load-store unit, and those belonging to an adder that calculates the address that should be taken after a jump instruction. Table I shows how many paths belong to each group. Following the approach described in [15], it was possible to generate an STL capable of testing all path delay faults stemming from these paths. Such STL constitutes the starting point for evaluating how much the fault coverage degrades as new critical paths are introduced with time.

TABLE I
Paths distribution per modules in the fresh circuit

Module	*#Paths*
ALU_Div Adder	51
LoadStore Adder	2,004
Jump_Addr Adder	4,628
Total	6,683

The proposed aging framework requires to provide a SAIF file that contains switching activity data for each cell in the processor. For this reason, two programs were chosen so that switching activity data could be extracted from performing a logic simulation, giving an insight on how much aging depends on the switching activity parameter. The two programs are *basicmath_small* and *qsort* from the *automotive* section of the MiBench-Embedded benchmark suite [30]. While *basicmath_small* can be run as is, *qsort* required a slight change in the source C code to remove all instances related to `FILE` variables, as they take a considerable amount of instruction RAM: vectors to sort were hence declared and defined in the code rather than read from a file as originally intended. *Basicmath-small* requires 14.73 kB of memory and $1,743,500$ clock cycles to execute completely, while *qsort* requires 16.18 kB of memory and $2,021,548$ clock cycles to execute.

V. Experimental Results

All the experiments have been launched on 5 cores of an Intel Xeon CPU E5-2680 v3 machine. The proposed flow has been implemented in Python. Generating the aged SDF file takes no longer than a couple of minutes, provided that all other input files are already available at the start of the experimental session. The extraction of aged critical paths, on the other hand, requires about 12 hours to complete.

Experimental results for the *basicmath_small* and *qsort* programs are reported in Fig. 2 and Fig. 3, respectively. In both figures the fault coverage trend over time, reported as the percentage of detected faults, is shown in red, while the absolute value of critical paths trend over time is reported in blue. In both cases, the total amount of critical paths grows in time: this can be explained noting that most of the paths that are sub-critical at time zero, i.e., paths whose slack is quite large but not enough for them to be considered critical, slow with time, thus becoming critical paths. This phenomenon becomes more accentuated with time, with a steep growth past the 5 years mark. This reflects on the fault coverage figures as well, showing an overall decrease from the initial 100% fault coverage down to 83.14% for the *basicmath_small* program and 83.12% for the *qsort* program.

In order to better understand how critical paths evolve in time, an additional analysis concerning how many critical paths in the fresh circuit can still be found after aging the device under test has been performed, together with one

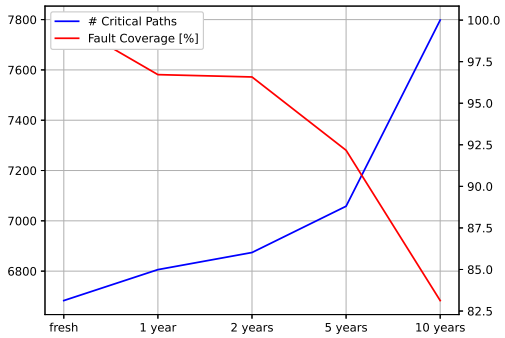

Fig. 2. Critical path and path delay fault coverage evolution over time for the *basicmath_small* program

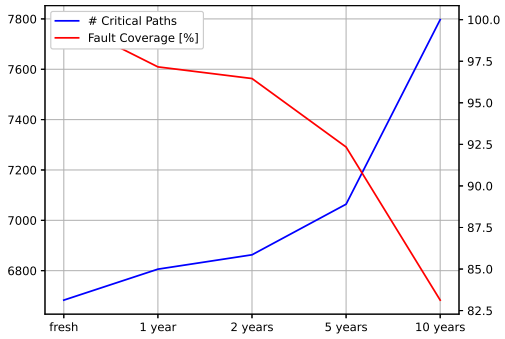

Fig. 3. Critical path and path delay fault coverage evolution over time for the *qsort* program

investigating how many new paths are introduced with aging. Such data is reported in Fig. 4 and Fig. 5 for the *basicmath_small* program and the *qsort* program, respectively. As for the previous set of data, the two programs show a similar trend over time, both in terms of how many paths remain unchanged and how many new ones are introduced. With the exception of the two years mark, the number of critical paths that never change over time decreases, losing more than 200 paths with respect to the fresh circuit, while the number of new paths grows, with slightly more than 1,300 paths added after 10 years. Focusing on the new paths introduced by aging, it is interesting to investigate their ranking, i.e., whether they are particularly slow or not. After ten years, 10% of the new paths fall within the top 30% slowest paths for both programs. This shows that, although the vast majority of paths introduced by aging are not the slowest ones, there is still a non-negligible amount of paths whose slack is small enough for them to be among the slowest critical paths. Finally, it is noted that, for both programs, every set of new paths introduced by aging includes that of the antecedent time mark, e.g., all new paths introduced after one year are found after two years, all paths introduced after two years are found after five years and all paths introduced after five years are found after ten years. Moreover, all paths introduced by aging can still be grouped

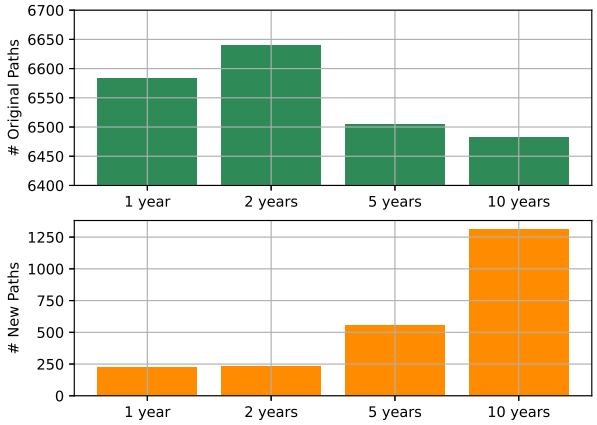

Fig. 4. Original and new critical paths evolution for basicmath_small program

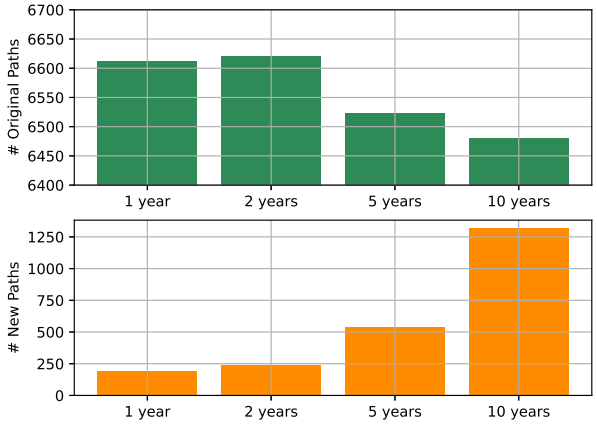

Fig. 5. Original and new critical paths evolution for qsort program

in the three categories reported in Table I, that is, an adder in the ALU divider, an adder in the Load/Store unit, and an adder used to calculate the target address when a jump instruction is to be taken.

By analyzing the above results, it is possible to state that aging a processor introduces a non-negligible amount of critical paths over time, with a steep increase after five years of use. Such paths are currently not detected through the STL that has been developed for the fresh circuit's critical paths, thus posing a problem when it comes to ensuring the safety and reliability of the processor over long periods of time. For this reason, test engineers should develop the STL also taking into account the set of critical paths that are to be tested as the circuit age, making sure that it can ensure a satisfactory fault coverage throughout the processor's operative lifetime, choosing between the possibility of having (i) an all-encompassing STL since time zero, capable of detecting failures stemming from faults in all critical paths including those due to aging, or (ii) a modular STL that is capable of

enabling chunks of code, adapting to the paths as they age.

VI. Conclusions

This paper describes a methodology on how to age an integrated circuit, with the goal of ensuring its safety and reliability over a long period of time. We propose an evaluation framework based on automatic aging tool, through which it is possible to obtain an aging model for each cell in sequential designs such as processors, leading to the generation of an aged SDF. Thanks to that, it is possible to extract lists of aged critical paths, understanding how critical paths, and thus the path delay fault coverage, change over time. Experimental data gathered on two programs from the MiBench-Embedded benchmark running on a RISC-V processor show that as the circuit ages, more and more paths that once were sub-critical become critical, leading to a 17% decrease in fault coverage after ten years. Thanks to this information, test engineers can improve STLs for path delay faults so that they include test vectors for path delay faults that may originate over time, ensuring that strict levels of reliability are met throughout the operative lifetime of the device and proving the effectiveness of this methodology. We are currently working in this direction.

Finally, the evaluation methodology is strictly dependent on the technology library adopted, and it currently only takes into account the propagation delay of cells, neglecting that of interconnections between gates. New research can be done in this area, either collecting new data on other libraries and checking if aging occurs differently based on the library and understanding how interconnections affect the aging of propagation delay through paths. Moreover, future works will adopt additional state-of-the-art techniques for aging estimations.

References

[1] T. Sakurai and A. Newton, "Alpha-power law mosfet model and its applications to cmos inverter delay and other formulas," *IEEE Journal of Solid-State Circuits*, vol. 25, no. 2, pp. 584–594, 1990.

[2] S. S. Sapatnekar, "What happens when circuits grow old: Aging issues in cmos design," in *2013 International Symposium onVLSI Design, Automation, and Test (VLSI-DAT)*, 2013, pp. 1–2.

[3] B. Halak, V. Tenentes, and D. Rossi, "The impact of bti aging on the reliability of level shifters in nano-scale cmos technology," *Microelectronics Reliability*, vol. 67, pp. 74–81, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0026271416304103

[4] Y. Zhao and H. G. Kerkhoff, "Highly dependable multi-processor socs employing lifetime prediction based on health monitors," in *2016 IEEE 25th Asian Test Symposium (ATS)*, 2016, pp. 228–233.

[5] M. Altieri, S. Lesecq, E. Beigne, and O. Heron, "Towards on-line estimation of bti/hci-induced frequency degradation," in *2017 IEEE International Reliability Physics Symposium (IRPS)*, 2017, pp. CR–6.1–CR–6.6.

[6] K. Senthamarai Kannan, "Management des performances de sûreté et de sécurité pour les applications automotives et iot," Ph.D. dissertation, Université Grenoble Alpes, 2021, thèse de doctorat dirigée par Anghel, Lorena et Portolan, Michele Nanoélectronique et nanotechnologie Université Grenoble Alpes 2021. [Online]. Available: http://www.theses.fr/2021GRALT044

[7] S. Hussain, M. A. Raheem, and A. Ahmed, "Sic-tpg for path delay fault detection in vlsi circuits using scan insertion method," in *2021 Devices for Integrated Circuit (DevIC)*, 2021, pp. 1–5.

[8] I. Pomeranz, "On the detection of path delay faults by functional broadside tests," in *2012 17th IEEE European Test Symposium (ETS)*, 2012, pp. 1–6.

[9] ——, "Gepdfs: Path delay faults based on two-cycle gate-exhaustive faults," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 7, pp. 2315–2322, 2022.

[10] M. Psarakis, D. Gizopoulos, E. Sanchez, and M. Sonza Reorda, "Microprocessor software-based self-testing," *IEEE Design Test of Computers*, vol. 27, no. 3, pp. 4–19, 2010.

[11] P. Bernardi, M. Grosso, E. Sanchez, and M. Sonza Reorda, "On the automatic generation of test programs for path-delay faults in microprocessor cores," in *12th IEEE European Test Symposium (ETS'07)*, May 2007, pp. 179–184.

[12] K. Christou, M. K. Michael, P. Bernardi, M. Grosso, E. Sanchez, and M. S. Reorda, "A novel sbst generation technique for path-delay faults in microprocessors exploiting gate- and rt-level descriptions," in *26th IEEE VLSI Test Symposium (vts 2008)*, April 2008, pp. 389–394.

[13] Wei-Cheng Lai, A. Krstic, and Kwang-Ting Cheng, "Test program synthesis for path delay faults in microprocessor cores," in *IEEE Intl. Test Conference*, 2000, pp. 1080–1089.

[14] N. I. Deligiannis, R. Cantoro, T. Faller, T. Paxian, B. Becker, and M. S. Reorda, "Effective sat-based solutions for generating functional sequences maximizing the sustained switching activity in a pipelined processor," in *2021 IEEE 30th Asian Test Symposium (ATS)*, 2021, pp. 73–78.

[15] L. Anghel, R. Cantoro, R. Masante, M. Portolan, S. Sartoni, and M. S. Reorda, "Self-test library generation for in-field test of path delay faults," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1–1, 2023.

[16] R. Cantoro, P. Girard, R. Masante, S. Sartoni, M. S. Reorda, and A. Virazel, "Self-test libraries analysis for pipelined processors transition fault coverage improvement," in *2021 IEEE 27th International Symposium on On-Line Testing and Robust System Design (IOLTS)*, 2021, pp. 1–4.

[17] F. A. da Silva, R. Cantoro, S. Hamdioui, S. Sartoni, C. Sauer, and M. Sonza Reorda, "A systematic method to generate effective stls for the in-field test of can bus controllers," *Electronics*, vol. 11, no. 16, 2022. [Online]. Available: https://www.mdpi.com/2079-9292/11/16/2481

[18] D. K. Schroder and J. A. Babcock, "Negative bias temperature instability: Road to cross in deep submicron silicon semiconductor manufacturing," *Journal of Applied Physics*, vol. 94, no. 1, pp. 1–18, 2003. [Online]. Available: https://doi.org/10.1063/1.1567461

[19] T. Ning, "Hot-electron emission from silicon into silicon dioxide," *Solid-State Electronics*, vol. 21, no. 1, pp. 273–282, 1978. [Online]. Available: https://www.sciencedirect.com/science/article/pii/003811017890148X

[20] H. J. Lee and K. K. Kim, "Analysis of time dependent dielectric breakdown in nanoscale cmos circuits," in *2011 International SoC Design Conference*, 2011, pp. 440–443.

[21] J. McPherson, "Time dependent dielectric breakdown physics – models revisited," *Microelectronics Reliability*, vol. 52, no. 9, pp. 1753–1760, 2012, special Issue 23rd European Symposium on the Reliability of Electron Devices, Failure Physics and Analysis. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0026271412001916

[22] T. T.-H. Kim, P.-F. Lu, K. A. Jenkins, and C. H. Kim, "A ring-oscillator-based reliability monitor for isolated measurement of nbti and pbti in high-k/metal gate technology," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 23, no. 7, pp. 1360–1364, 2015.

[23] J. Keane, X. Wang, D. Persaud, and C. H. Kim, "An all-in-one silicon odometer for separately monitoring hci, bti, and tddb," *IEEE Journal of Solid-State Circuits*, vol. 45, no. 4, pp. 817–829, 2010.

[24] M. Altieri Scarpato, "Estimation de la performance des circuits numériques sous variations pvt et vieillissement," Ph.D. dissertation, Université Grenoble Alpes, 2017, thèse de doctorat dirigée par Beigné, Édith et Lesecq, Suzanne Nano electronique et nano technologies Université Grenoble Alpes (ComUE) 2017. [Online]. Available: http://www.theses.fr/2017GREAT093

[25] N. Weste and D. Harris, *CMOS VLSI Design: A Circuits and Systems Perspective*, 4th ed. USA: Addison-Wesley Publishing Company, 2010.

[26] A. Sivadasan, S. Mhira, A. Notin, A. Benhassain, V. Huard, E. Maurin, F. Cacho, L. Anghel, and A. Bravaix, "Architecture- and workload-dependent digital failure rate," in *2017 IEEE International Reliability Physics Symposium (IRPS)*, 2017, pp. CR–8.1–CR–8.4.

[27] J. Watt, R. Borhani, and A. K. Katsaggelos, *Machine Learning Refined: Foundations, Algorithms, and Applications*, 1st ed. USA: Cambridge University Press, 2016.

[28] M. Kuhn and K. Johnson, *Applied Predictive Modeling*. Springer, 2013.

[29] ETH Zurich and Università di Bologna, "PULPino microcontroller system," https://github.com/pulp-platform/pulpino, 2022.

[30] J. Bennett, “Mibench-embedded benchmark,” https://github.com/embecosm/mibench/tree/master, 2023.