Available online at www.sciencedirect.com
ScienceDirect
Microelectronics Reliability 00 (2017) 1–14
Microelectronics Reliability

BTI Mitigation by Anti-Ageing Software Patterns

Haider Muhi Abbas, Basel Halak, Mark Zwolinski

Electronics and Computer Science
University of Southampton
Southampton SO17 1BJ, UK

Abstract

This paper presents a time-redundant technique to mitigate Negative and Positive Bias Temperature Instability (NBTI/PBTI) ageing effects on the functional units of a processor. We have analysed the sources and effects of ageing from the device level to the Instruction Set Architecture (ISA) level, and have found that an application may stress the critical paths in such a way that the circuit has half of its nodes always NBTI-stressed. To mitigate this behaviour, we propose an application-level solution to balance the stress and put the timing-critical gates of the critical path into a relaxed (balanced) mode. The results show that the lifetime of the system can be doubled by applying balanced stress patterns at the software level during the idle time of a processor system.

Keywords: Negative/Positive Bias Temperature Instability (NBTI/PBTI), Workload, Application Specific Systems, Idle Time, Anti-ageing.

1. Introduction

Ageing or time-dependent variations in CMOS devices represent a challenge to the design of integrated circuits, along with power consumption and performance. With technology feature sizes scaling down, critical issues related to system reliability, such as soft errors, hard errors, and process variations, [1], are emerging. Ageing-degradation can manifest itself as soft errors that become hard errors, bringing the system to a state where timing constraints are violated, and finally, the system fails to function properly. The mechanisms at the device level responsible for that degradation include: Positive/Negative Bias Temperature Instability (PBTI/NBTI), Hot Carrier Injection (HCI) and Time-Dependent Dielectric Breakdown (TDDB) and manifest themselves as changes in the device threshold voltage, the carrier mobility, and the insulating properties of gate dielectrics, [2].

In this work, we are primarily concerned with the mitigation of Bias Temperature Instability (BTI) and particularly NBTI. BTI has two different phases:

- Stress: Interface traps are generated at the interface of the substrate and gate oxide layers due to electrical stress (i.e. negative bias for PMOS and positive bias for NMOS) that leads to breaking of some of the Si-H or Si-O bonds. Consequently, the threshold voltage of the transistor increases over time.

- Relaxation/Recovery: some of the generated traps are removed from the interface. However, the relaxation phase cannot completely compensate for the effect of the stress phase and therefore the overall effect of BTI is degradation in the threshold voltage of each transistor. The amount of degradation depends on the ratio between the stress period and the total operating period (the duty cycle).

Techniques for reducing stress and increasing recovery include controlling the signal probabilities. This can be done from the inputs of the circuit (Input Vector Control) or during the synthesis process by hiding those nodes with high probabilities of being zero, or by changing the pin order of the gates on the critical paths,

[3, 4, 5]. However, if we try to relax one node, other nodes in the signal path may be stressed.

The hypothesis of this work is that a processor needs to actively relax to recover from stress, rather than simply doing nothing during idle periods. In modern applications, processors tend to have many short idle periods; thus, simple power gating would not be a good solution for stress optimization, [6]. During normal operational periods, the input states of the transistors may be constant, leaving the transistors stressed. NBTI/PBTI could then be mitigated by applying balanced-stress stimuli to the critical paths at the software-level. Running a program on the processor for a non-functional purpose has been used in on-line testing (i.e. Software-Based Self-Test (SBST) methods) as this does not require modification of the hardware design [7].

The main contributions of this work are:

- A high-level ageing prediction model is proposed which takes into account the actual signal probabilities of the system. To achieve this we have developed a tool to derive the actual stress-recovery ratio of each logic gate.
- An application-level technology-independent mitigation technique is proposed to balance NBTI/PBTI effects. This brings the circuit into a recovery state by changing the nodes that are BTI-stressed to BTI-relaxed.

The organisation of this paper is as follows: ageing causes and mitigation techniques are covered in Section 2. In section 3, NBTI/PBTI stress analysis is presented. The proposed technique for generating and applying the balancing patterns is presented in section 4. The results of running the balancing program are given in section 5. Section 6 concludes the paper.

2. Background And Related Work

2.1. BTI Stress-Recovery

BTI in a transistor is caused by the generation of traps at the channel and dielectric interface. BTI in PMOS transistors is referred to as Negative BTI (NBTI), since the gates of PMOS devices are negatively biased with respect to the source (i.e. $V_{gs} = -V_{dd}$). BTI for NMOS transistors is referred to as Positive BTI (PBTI), since the gates of NMOS transistors are positively biased with respect to the source (i.e. $V_{gs} = V_{dd}$). NBTI and PBTI have the same consequences, namely that they increase the threshold voltages and decrease the driving currents of the devices, but for different physical reasons. It is generally accepted that NBTI degradation in SiO_2 and High-κ dielectric is due to dielectric interface traps. PBTI was considered to be negligible before the introduction of High-κ MOSFET technology [8]. In High-κ processes, PBTI degradation is due to a build-up of negative charges in the High-κ layer. Although High-κ technology is impacted by both NBTI and PBTI degradation, NBTI is still much more dominant according to recent results for Replacement Metal Gate (RMG) Technology, [9, 10].

The probability of a signal being zero, SP(0), or the duty cycle, reflects the fraction of time spent in the stress state. We simulated the effect of the duty cycle on the threshold voltage using a commercial reliability model, namely the HSPICE MOSRA Built-in Model Level 3 [11]. MOSRA can evaluate both NBTI and PBTI for a circuit at the SPICE level and obtain gate or path degradation, rather than just threshold voltage degradation or leakage current increase at the transistor level, as given by the basic Reaction-Diffusion (RD) model [12]. Decreasing the duty cycle can impact positively on the V_{th} degradation of PMOS devices and negatively on the V_{th} degradation of NMOS devices. MOSRA simulations use degraded device parameters in HSPICE to calculate gate or path delay degradation in timing simulations, [13].

Simulation parameters have been tuned to match the degradation given by statistical data, [14, 15]. We applied different SP(0) values to a circuit consisting of two inverters in series to calculate the maximum path delay after 10 years for two different technologies (90nm from Synopsys and 65nm from TSMC). We activate only the NBTI effects because PBTI should barely exist at these technology nodes and if modelled would exaggerate the ageing effect. As can be seen from Figs. 1 and 2, the signal probability has an impact on path delay degradation and, thus, on the lifetime. While one node is highly stressed, it will tend to have more static NBTI and by balancing the signal probabilities, the static stress will be reduced as well. As the delay degradation is less at the end of the device lifetime than at the beginning, decreasing the path delay by a small amount could enhance the lifetime of a device by several years. Fig. 3 shows the lifetime of the two inverter chain with a target maximum delay of 0.237ns for the 90nm technology from Synopsys and 0.149ns for the 65nm technology from TSMC.

2.2. NBTI Mitigation Techniques

One approach to mitigation is to reduce or even to eliminate design uncertainties. However, in practice,

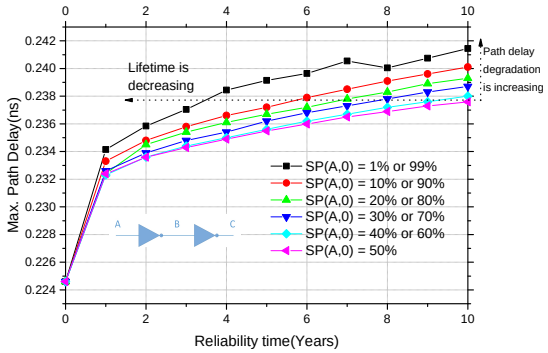

Figure 1: Path delay for a chain of two inverters for different SP(0) values at the input using 90nm Synopsys Technology with NBTI effect.

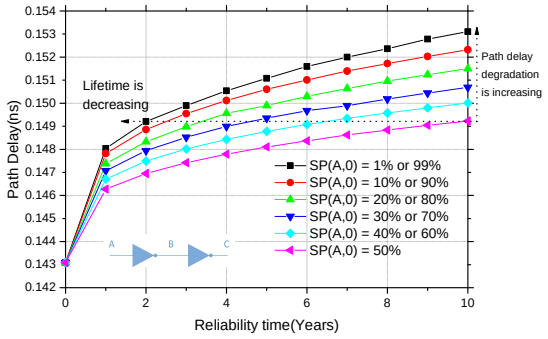

Figure 2: Path delay for a chain of two inverters for different SP(0) values at the input using 65nm TSMC Technology with NBTI effect.

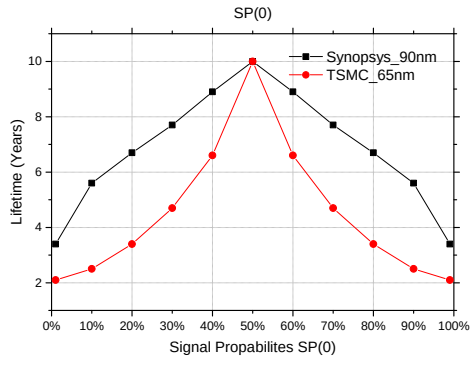

Figure 3: Lifetime for a chain of two inverters for different Signal Probabilities SP(0) values at the input.

there is more than one contributor to these uncertainties, including EDA tool limitations and complex environmental stress conditions, [16]. Another solution is conservative design under worst-case scenarios, but circuits do not always run at the worst-case condition, and such an over-designed approach is extremely costly with respect to power and area.

At gate level, BTI manifests itself as a time-dependent gate delay, leading eventually to timing violations. In [17], critical gates are identified and optimisation methodologies, e.g. gate resizing, have been proposed for these critical gates. However, NBTI has a dependence on the dynamic operation, such as the supply voltage, spatial or temporal temperature and signal probability and these parameters vary dynamically from gate to gate. Another solution, [3], uses signal probabilities to restructure the logic gates and arrival times of the input signals to reorder the pins. However, signal probabilities are assumed to be 50% for input signals, but in larger systems, the signal probability is dynamic and application-dependent. A technique to mitigate NBTI has been proposed, [18, 19], in which the signal probability is modified using Input Vector Control (IVC) or Multiple Input Vector Control (M-IVC) during the stand-by mode of the circuit. However, saving these vectors in memory has costly overheads in terms of area and power and the techniques do not consider the stress probabilities during the operational mode of the circuit.

At circuit and gate levels, different methods have been proposed to reduce NBTI degradation. Supply voltage scaling over time to reduce guard-bands and increase the lifetime of the circuit has been explored, [20]. Scaling the clock frequency has been proposed to detect and mask late transitions generated due to ageing, [21]. Reordering the gate pins and restructuring the transistor network to mitigate the NBTI degradation has also been suggested, [3, 22] .

There are many different approaches to controlling these effects. A technique called Dynamic Wear-out/NBTI Management (DNM) has been proposed, with the aim of reducing design margins [23, 24]. The DNM approach reduces the power consumption by running the circuit with the least possible supply voltage and changes the supply voltage periodically based on readings from delay sensors. However, the main challenge of this approach is the accuracy of the sensors and the area overheads that could exceed design constraints.

In general, there are three possible approaches to ageing mitigation: proactive, reactive and ageing-aware (protective). The proactive approach works as an estimator for ageing behaviour and is based on a physi-

cal model that describes ageing effects (NBTI, HCI and TDDB) at a low level, [25, 12, 4]. Usually, this approach needs to take into account the different contributions of time-dependent variations (e.g. signal probability, switching activity, temperature and supply voltage). The reactive approach is to monitor on-line the real behaviour resulting from ageing by using delay sensors [23, 24]. This approach is more precise and bypasses the complexity of modelling the ageing effects at the system level. However, the main problem of on-line sensors is area and power overheads and to moderate this only a limited numbers of nodes can be monitored. The third, protective, approach tries to alleviate the source of ageing either by reducing the operating temperature or reducing the stress probability in the critical path, [26, 17].

2.3. Data Dependency of Ageing-Induced Degradation of the Processor

Program data determines whether the nodes of the processor are stressed or not. The data includes both opcodes and operands. Firouzi et al, [26], looked at possible NOP instructions in the MIPS processor to reduce ageing. As well as the standard NOP instruction (sll r0, r0, 0) they considered other instructions that had no result (e.g. adding zero) to minimise stress. They proposed software and hardware techniques to assign the best input vector for these NOP instructions. This method would only be helpful, however, if the rate of NOP instructions is high with respect to the total number of operational instructions. To test this hypothesis, we ran different benchmarks from the MiBENCH suite, [27], for two different architectures on the gem5 simulator, [28]. Fig. 4 shows that the number of NOP instructions on the MIPS processor is significant, while on the ARM architecture it is negligible.

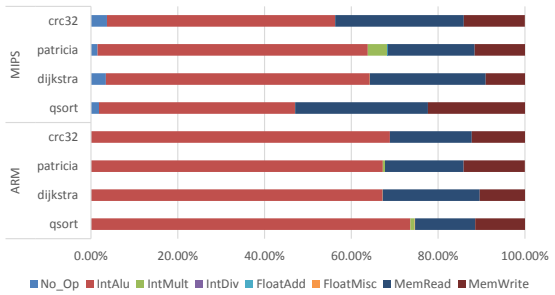

Figure 4: Percentages of classes of instruction for different benchmarks on MIPS and ARM processors.

Data from other paths can also stress the critical path. For example, assume the adder is in the critical path, and that during the execution of other operations data is routed to the adder, even though the result is not used. From a BTI perspective the critical path through the adder will be stressed. It has been claimed, [29], that the core of a processor can be brought to a failing state by executing a malicious program to age the circuit; this does not consider the signal probabilities of intermediate nodes, however (see Fig. 5). In practice, it is not possible to put all critical path nodes into a fully relaxed state (a Signal Probability of zero, SP(0) = 0%) or a fully stressed state (SP(0) = 100%), Fig. 5. On the other hand, it is possible to balance the stress by controlling the signal probabilities during the idle time of the processor.

3. NBTI/PBTI Stress Analysis

3.1. Ageing-Sensitive Critical Path Selection

The selection of paths to reverse the stress needs to consider both the initial path delays from the post-synthesis analyses and the gate types in the paths. A non-critical path at time zero could become a critical path after a number of years because paths degrade according to different factors, for example, duty cycle, temperature, frequency and circuit topology. Estimating the path most sensitive to ageing depends on model parameters that would not be available until the system has been fabricated and tested in the required environment. So, in this research, we have tried to avoid using an ageing model to define the criteria for selecting specific paths that are potentially vulnerable to ageing. Instead, we define a threshold (θ) for the ageing-critical path delay. For example, the maximum critical path degradation has been measured for different benchmark circuits for 10 years and found to be between 12.3% and 19.5%, [30]. Thus, we can define ageing sensitive critical paths as those paths that have slack in the range of zero to ($\delta_0 \times \theta$), inclusive, or have a path delay between δ_0 and $\delta_0(1-\theta)$, inclusive.

Also we have to consider the effect of process variations on selected paths by defining the worst case possible path deviation due to the process variations as $\Delta\delta_{pv}$. Then, an ageing- and process- sensitive critical path should be selected if its path delay is in the range:

$$\delta_0 \geq Path\ Delay \geq \delta_0(1-\theta-\Delta\delta_{pv}) \qquad (1)$$

These paths have nearly balanced path delays, but they could share instances with the first critical path (for example in an adder, if the carry chain path is shared between nearly-critical paths and the first critical path, then any degradation or reversed degradation on the

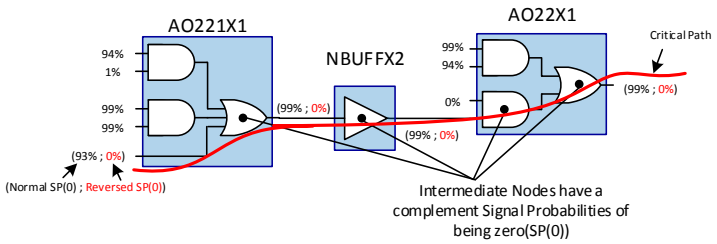

Figure 5: Zero-signal probabilities distribution on the critical path sub-circuit with Normal and Reversed state.

shared part will also affect the ageing-sensitive critical paths). Alternatively, if the critical paths have instances that are independent from one path to another, then all the ageing sensitive critical paths need to be individually analysed for ageing and possible reversal.

3.2. *SP(0) Distribution on the Critical Paths of the Processor*

Combinational logic circuits may show different degradations in each PMOS transistor because different input patterns can lead to different inputs to the CMOS transistors. Some PMOS transistors may degrade more because they have a SP(0) of 99%, but others may not degrade if they have a SP(0) of 1% at their gates. To date, however, there has been no consideration of the SP(0) of the intermediate nodes of the complex gates. For example, in [31, 29], the analysis was done only for the input transistors of gates. In the OR gate of Fig. 6, if IN1 is at "1", Q would be "1" regardless of IN2, but there is a node, QN, inside the OR gate that would be stressed.

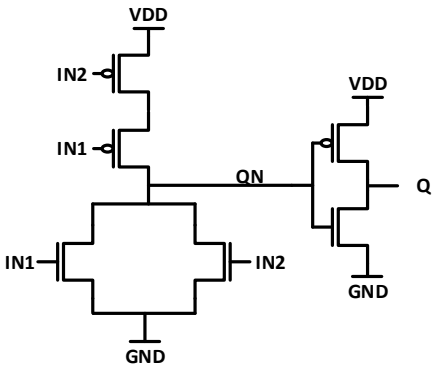

Figure 6: CMOS circuit for OR gate (NOR + INVERTER).

To measure the delay degradation on each net of the critical path, we need to consider the SP(0) of each input and of the internal nodes of the gate cells. We used the OpenRISC core for this analysis. First, synthesis was done with the full set of cells available in the 90nm Synopsys library, including those cells that have internal nodes. There are 2888 critical paths in the circuit, but various critical paths pass though the same gate cells. For example, the first 100 critical paths share more than 92% of the cells in the most critical path, Table 1. Thus, if there is any degradation in the shared part, it would affect all these critical paths. Moreover, the average SP(0) for the first 100 critical paths is around 80% when executing a "Hello World" program. This means the program will stress the critical path. In this example, the nodes are totally NBTI stressed because the probabilities of signals being zero are close to 100%. However, this does not consider the hidden nodes of the compound gates and so the average SP(0) is not correct. These hidden nodes would have complementary values and therefore have no NBTI stress but could have PBTI stress. In other words, a circuit with SP(0) close to 0% would have hidden nodes with SP(0) close to 100%. To calculate a more accurate figure, we ran different instructions on a processor synthesised using only cells that have only one-level of transistors, in order to avoid any hidden nodes, as given in Table 2. The SP(0) at each node is generally the complement of that in the previous node and the average SP(0) is around 50%. So, the object should be to reverse these signal probabilities to obtain signal probabilities that are as balanced as possible (around 50%), rather than reducing one signal probability to avoid NBTI stress. This example is only used to show the signal probabilities of the hidden nodes, as this case is not feasible in real synthesis.

Table 1: SP(0) distribution on the critical paths of the OpenRISC processor for Hello World program, using compound gates.

	1st critical path		2nd critical path		3rd critical path		10th critical path		100th critical path	
	Cell type	SP(0)	Cell type	SP(0)	Cell type	SP(0)	Cell type	SP(0)	Cell type	SP(0)
1	DFFX1	50%	DFFX1	50%	DFFX1	50%	DFFX1	50%	DFFX1	50%
2	DFFX1	98%	DFFX1	98%	DFFX1	98%	DFFX1	98%	DFFX1	98%
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
65	AO22X1	93%	AO22X1	93%	AO22X1	93%	AO22X1	93%	AO22X1	93%
66	OR2X1	93%	OR2X1	93%	OR2X1	93%	OR2X1	93%	XOR3X1	94%
67	AO22X1	93%	AO22X1	93%	AO22X1	93%	AO22X1	93%	AOI22X1	5%
68	XOR3X1	94%	XOR3X1	94%	XOR3X1	94%	XOR3X1	94%	NAND4X0	94%
69	AOI22X1	5%	AOI22X1	5%	AOI22X1	5%	AOI22X1	5%	AO221X1	93%
70	NAND4X0	94%	NAND4X0	94%	NAND4X0	94%	NAND4X0	94%	NBUFFX2	99%
71	AO221X1	93%	AO221X1	93%	AO221X1	93%	AO221X1	93%	AO22X1	14%
72	NBUFFX2	99%	NBUFFX2	99%	NBUFFX2	99%	NBUFFX2	99%	DFFX1	14%
73	AO22X1	99%	AO22X1	77%	AO22X1	99%	AO22X1	0%		
74	DFFX1	99%	DFFX1	77%	DFFX1	99%	DFFX1	0%		
Average SP(0)		83%		82%		83%		80%		80%

]

Table 2: SP(0) distribution on the first critical path of the OpenRISC processor for different instructions.

		addi rD,rA,I				movhi rD,I			
	Cell type	I=0000_H	5555_H	AAAA_H	FFFF_H	0000_H	5555_H	AAAA_H	FFFF_H
1	DFFX1	50%	50%	50%	50%	50%	50%	50%	50%
2	DFFX1	99%	99%	99%	99%	99%	99%	99%	99%
3	NAND2X0	0%	0%	0%	0%	0%	0%	0%	0%
4	NAND2X0	0%	0%	0%	0%	99%	99%	99%	99%
5	NAND2X0	99%	99%	99%	99%	0%	0%	0%	0%
6	NAND4X0	0%	0%	0%	0%	99%	99%	99%	99%
7	NOR2X0	99%	99%	99%	99%	0%	0%	0%	0%
8	AOBUFX1	99%	99%	99%	99%	0%	0%	0%	0%
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
150	NAND3X0	50%	50%	50%	50%	34%	34%	34%	35%
151	NAND2X0	0%	0%	0%	0%	0%	0%	0%	0%
152	INVX0	99%	99%	99%	99%	99%	99%	99%	99%
153	NAND3X0	0%	0%	0%	0%	0%	0%	0%	0%
154	NAND2X0	99%	99%	99%	99%	99%	99%	99%	99%
155	DFFX1	99%	99%	99%	99%	99%	99%	99%	99%
Average SP(0)		49%	49%	49%	49%	48%	49%	48%	48%

3.3. Impact of Instruction/Program Level Workload on the Stress Probabilities

To study the effect of different instructions on the stress of the critical or nearly critical paths, we ran two different instructions with four different operands each, to determine whether the opcode or the data have a significant impact. The synthesis was been done using only cells that have no hidden nodes and the SP(0) at each node is generally the inverse of that of the previous node in the path. For the 155 nodes in the critical path, the average SP(0) was around 50%, as can be seen from the symmetry of the histograms in Fig. 7. The average SP(0) of the paths does not change significantly with the opcodes or operands. However, the signal probability distributions on the critical path do depend on the opcode and operands, as shown in Fig.7("movhi rD,5555_H"), where the signal probabilities tend to the extremes (10%>SP(0)>90%) even with symmetrical distributions that stress the critical paths with both NBTI and PBTI. Similarly different MiBench benchmarks show nearly symmetrical average stress, as can be seen from Fig. 8. Therefore, it would be desirable to reduce the number of nodes at the extremes of these histograms (e.g. 25%>SP(0)>75%).

Hence we conclude that a single instruction or program will not relax or stress 100% of the critical paths, but that a program-level solution could relax or stress some specific nodes. In other words, if there are some

nodes in the processor that could face a continual stress while others are not stressed all, it possible to balance that effect at the application level.

3.4. Gate Level Stress Balancing

We considered balancing the signal state of basic logic gates (inverter, NAND and NOR) compared with inverting the signal probability. We examined how inverting the signal probability would affect the ageing degradation. We have used HSPICE for simulating path delays and modelled the NBTI using MOSRA Level 3 considering two different cases:

- CASE A: Unbalanced stressed nodes – the nodes of the critical paths are either significantly NBTI-stressed (SP(0) greater than 75%) or significantly unstressed (or PBTI-stressed) (SP(0) less than 25%).
- CASE B: Balanced stressed nodes – the nodes of the critical path have SP(0) around 50%.

For the inverter, we simulated the degradation of a path of two inverters over ten years using the cases discussed above. The results show an advantage of 23.17% in the path delay and more than 50% in terms of time, Fig. 9.

For NAND and NOR gates, the same simulation as for the inverter was done but also considering two further dependencies: the signal probabilities of the secondary inputs of the gates, and the input pin order. The results show that swapping input pins can decrease the advantage obtained from balancing the signal probabilities. Also, the signal probabilities of the secondary input will not significantly affect the benefits of balancing the signal probabilities over the critical path, Table 3 and Table 4.

We also considered how the balanced stress patterns affect the remaining paths of the circuit. To answer this fundamental question, we examined the proposed technique on a two-bit adder. In this example, there is one target critical path and two nearly-critical paths, Fig. 10. In this example, we extracted the critical paths list after synthesizing the circuit using Design Compiler. Again we used the MOSRA Level 3 model in HSPICE simulations to model the degradation of the circuit, using the two above-mentioned cases. The results show that for all paths, there will be an advantage up to 50% in the expected lifetime from balancing the signal probabilities in the critical path (Fig. 11). Fig. 11 also shows that nearly critical paths share more than half of their nodes with the target critical path. Thus, any advantage in balancing the signal probabilities of the critical path will lead to an advantage in the remaining paths. If the nearly-critical paths do not share nodes with the critical path, then it is possible to control both the critical and the nearly-critical paths in parallel.

4. Proposed Technique

We propose a two-phase technique to mitigate the BTI ageing effects. In the first phase anti-ageing patterns (the balance states) are generated and these patterns are applied in the second phase by executing a stress-relief program instead of running a process idle task.

The flow of the first phase of the proposed techniques is illustrated in Fig. 12. We find the normal states (the Critical Path Stress States) of the nodes that need to be balanced by running different benchmarks and instructions. We obtain the signal probabilities of the nets being stressed to logic zero (SP(0)) from gate-level simulations of the processor executing benchmarks. We calculate the SP(0) of the critical paths that have slacks less than predefined maximum path delay degradation.

The second phase of the technique is to balance the effect of BT by reversing the average signal probabilities by applying stress-relaxing patterns to the timing-critical components in the functional unit of the processor during idle states.

4.1. Case Study:Program-Level NBTI/PBTI Balancing

We synthesized an OpenRISC 1200 processor core using the 90nm Synopsys technology [1]. The VCD (Value Change Dump) file from each post-synthesis simulation contains both switching activity, that is used to estimate the dynamic power at the design phase, and the signal probability that we used to estimate the BTI effect on performance degradation. From this we extracted the SP(0) for all the nets of the processor. To balance the effect of signal probability on the critical path, we need to find input patterns that will invert the signal states. This is effectively the same as generating test patterns for single stuck faults. We used an ATPG tool to find test patterns for stuck-at-0 faults on nets that have high SP(0) so as to set those nets to '1'. In the same way, we generated patterns for stuck-at-1 faults on the nets that have low SP(0).

The critical paths of the OpenRISC 1200 processor are in the adder. 38 nodes have an SP(0) greater than 75%; 10 nodes have an SP(0) less than 25%.

[1] The technology is not important because the technique depends on reversing the SP(0) rather than estimating the ageing. Hence, it is also independent of the BTI model.

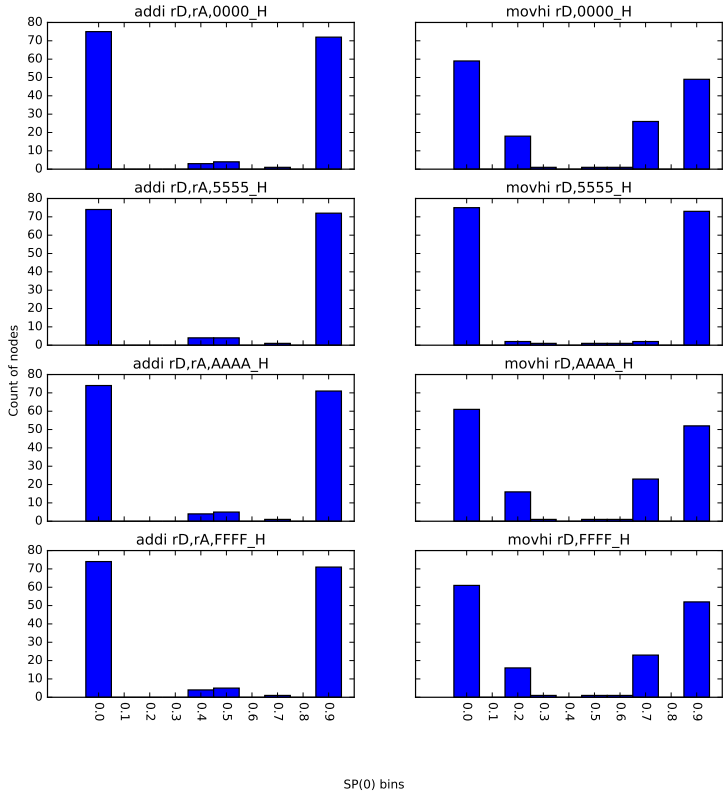

Figure 7: SP(0) distribution on the first critical path of the OpenRISC processor for different instructions.

Table 3: Path delay degradation advantages for a chain of two NAND2 considering balanced and unbalanced signal probabilities.

SP(0) of the 2nd input	Swap input order	CASE A degradation	CASE B degradation	Advantage over path delay	Advantage over years
50%	NO	9.27%	7.25%	21.75%	6 years
50%	YES	9.49%	7.62%	19.73%	5 years
99%	NO	7.35%	5.82%	20.80%	5 years
99%	YES	8.50%	6.88%	19.10%	5 years
1%	NO	10.38%	8.34%	19.69%	5 years
1%	YES	10.46%	8.18%	21.82%	5 years

The ATPG tool found 8 test patterns to set these nodes to balanced stress conditions. Each pattern will apply balanced stress to one or more nodes; the full set is needed for every node. As the results given in Section 5 show, the percentages of stressed nodes will be reduced significantly after applying these patterns. Table 5 shows these patterns as they would be applied to inputs A[31 .. 0] and B[31 ..0] of the adder. The patterns could be applied either in a test mode or by writing a program.

The OpenRISC 1200 Instruction Set Architecture has only a 16-bit immediate mode. These balance-stress patterns have a 32-bit widths and so are stored in consecutive memory locations starting with address K. The program, Fig. 13, transfers K (immediate value) to register 1 for use as an offset address. Then two patterns are loaded from memory to registers 2 and 3. The two patterns are applied to the adder with an ADD opera-

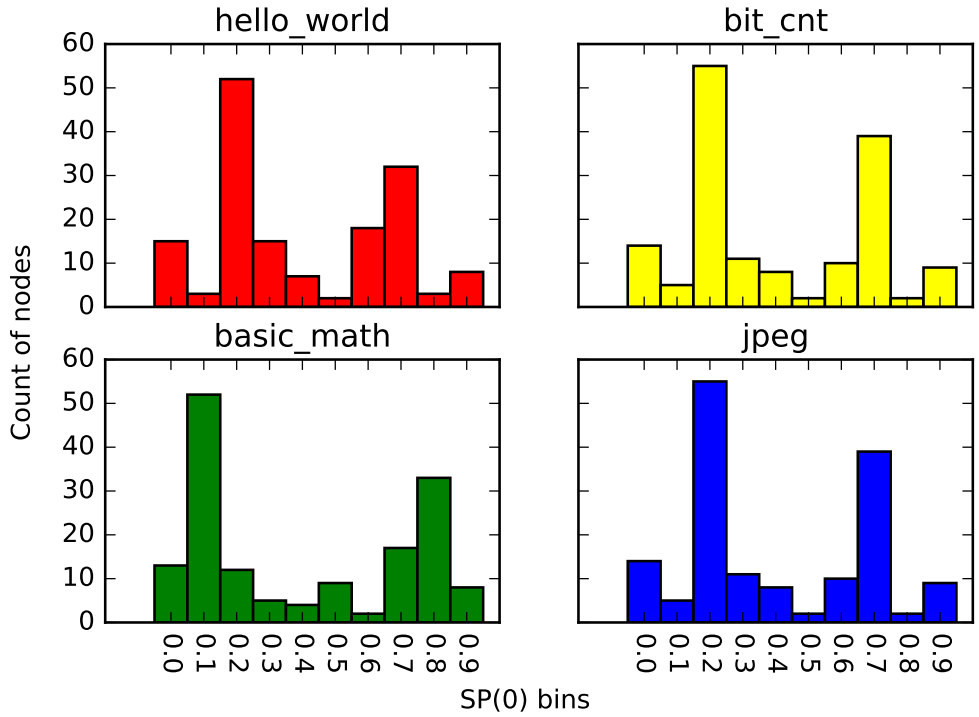

Figure 8: SP(0) distribution on the first critical path of the OpenRISC processor for different programs.

Table 4: Path delay degradation advantages for a chain of two NOR2 considering balanced and unbalanced signal probabilities.

SP(0) of the 2nd input	Swap input order	CASE A degradation	CASE B degradation	Advantage over path delay	Advantage over years
50%	NO	8.55%	6.81%	20.35%	5 years
50%	YES	9.10%	7.57%	16.80%	4 years
99%	NO	8.55%	6.81%	20.35%	5 years
99%	YES	9.76%	8.77%	10.18%	3 years
1%	NO	5.83%	4.54%	22.08%	6 years
1%	YES	7.38%	5.68%	23.08%	3 years

Table 5: OpenRISC balance-stress patterns.

A[31:0]				B[31:0]			
48	CB	3E	67	D4	40	7A	4C
24	65	9F	2D	EA	20	3D	3C
DA	F9	F1	D1	21	50	64	F2
25	B7	C6	93	C4	E8	48	35
DE	4F	08	A3	F9	CD	6D	DE
BF	58	FA	23	6A	33	27	3F
67	5B	2E	9C	58	C3	65	7F
AC	8C	03	18	A4	DC	93	8D

tion. The same sequence is applied for the remaining patterns. Finally, this program sits in a loop to be run during the idle states of the system.

Further optimization is possible to the above program to reduce the memory access and thus to reduce the power consumption of the running program, Fig. 14.

Another consideration is that this program may not have the privileges to run while another program is running. So the scheduler should give the lowest priority to this program and run it when the system is idle. However, if it is decided that this program should run as a routine in response to an interrupt, then the context of interrupted process needs to be saved. In this case the

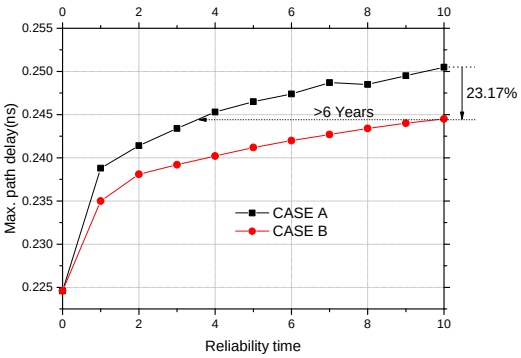

Figure 9: Path delay degradation for a chain of two Inverters considering balanced and unbalanced signal probabilities.

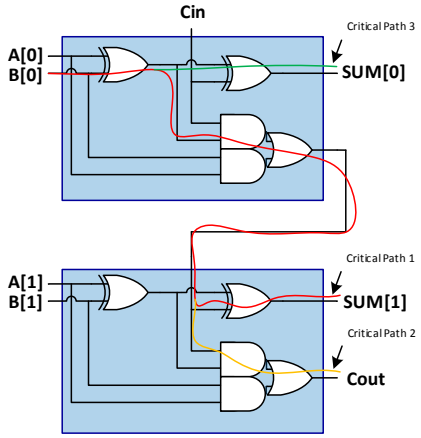

Figure 10: The most three critical paths on the two-bit adder.

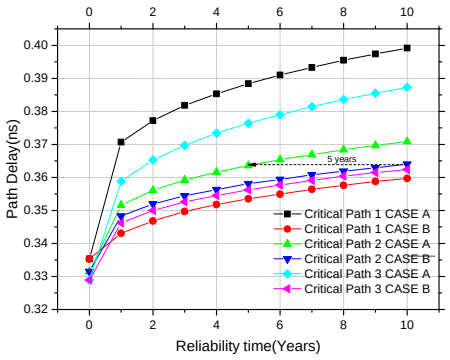

Figure 11: Path delay degradation of the most three critical paths in the two-bit adder considering unbalanced and balanced signal probabilities over the critical path.

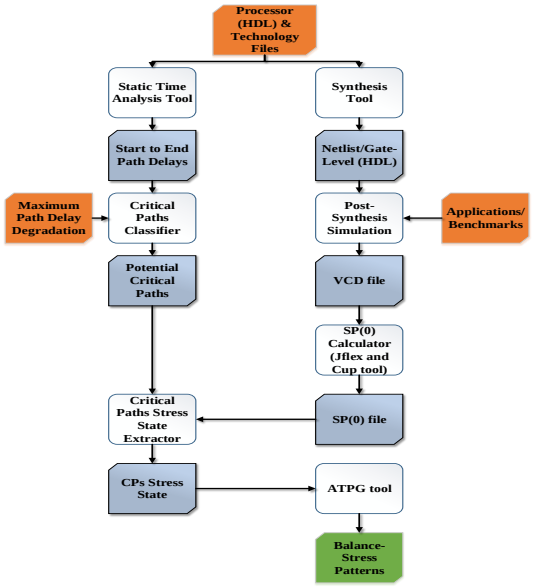

Figure 12: BTI balance-stress-patterns generating flow

program should push the registers onto the stack at the start of the routine and pop them off at the end of the routine to save the context of the interrupted program, Fig. 15

5. Evaluation and Discussion

To evaluate the effect of running the balancing program and how the signal probability will be affected on the critical path, we ran the balancing program along with different benchmarks and varied the percentages of the running time of the balancing from 10% to 50%. To compare results, we calculated the number of stressed nodes as follows:

$$\text{Percentage of stressed nodes} = \frac{\#\text{ stressed nodes}}{\text{critical path nodes}}$$

where the stressed nodes are those critical path nodes that have an SP(0) greater than 75% or less than 25%. As would be expected, to obtain a balanced state for the stressed nodes requires the balancing program to run for 50% of the time. Needless to say, it is not always possible to have this idle time or to add redundant time for the purpose of relaxing BTI. Fig. 16 shows the effect on the stressed nodes of running the BTI balancing program for different percentages of the overall time.

Fig. 17 shows the effect of running the BTI balancing program for different times on the percentages of the stressed nodes in critical paths of the OpenRISC processor. The results show that balancing one critical path

```
l.movhi r1, K
# Stimulate the first pattern
l.lws r2, 0(r1)
l.lws r3, 1(r1)
l.add r4, r2, r3
# Stimulate the second pattern
l.lws r2, 2(r1)
l.lws r3, 3(r1)
l.add r4, r2, r3
    .
    .
# Stimulate the eighth pattern
l.lws r2, 14(r1)
l.lws r3, 15(r1)
l.add r4, r2, r3
l.rfe # Return From Exception
```

Figure 13: Balancing program

```
# Stimulate the first pattern
l.movhi r2, 48CB
l.ori r2, r2, 3E67
l.movhi r3, D440
l.ori r3, r3, 7A4C
l.add r4, r2, r3
# Stimulate the second pattern
l.movhi r2, 2465
l.ori r2, r2, 9F2D
l.movhi r3, EA20
l.ori r3, r3, 3D3C
l.add r4, r2, r3
    .
    .
# Stimulate the eighth pattern
l.movhi r2, AC8C
l.ori r2, r2, 0318
l.movhi r3, A4DC
l.ori r3, r3, 938D
l.add r4, r2, r3
l.rfe # Return From Exception
```

Figure 14: Optimised balancing program

will balance other near-critical paths as they share nodes with the first path. On the other hand, if the near-critical paths do not share many nodes with the targeted critical path, it is possible to apply balancing patterns in the same way for the first critical path independently of the other paths. Although balancing signal probabilities would work with embedded systems that run specific applications, it is also possible to use the technique for a general-purpose processor. Fig. 18 shows how the percentages of stressed nodes on the first critical path of the OpenRISC processor reduce when executing the balancing program along with a different program from the MiBENCH benchmarks.

Next, to verify that the balancing program will reduce the degradation in the path delay of the processor, we simulated the adder using HSPICE and modelled NBTI using the MOSRA Level 3. We stimulated the circuit with two cases:

- CASE A (Normal Stressed Mode): Stress patterns with the equivalent signal probabilities of the Hello World program.

- CASE B (Balanced Mode): Balanced stressed nodes (i.e. the nodes in the critical path having an SP(0) around 50% by running the anti-ageing program along with the Normal Stressed Mode program).

The results show that running the balancing program would decrease the path delay by 20.24% compared with normal operation and double the expected lifetime as shown in Fig. 19

5.1. Discussion

In our analysis, we expect, for example, 11% degradation in six years as can be seen in Fig. 19, so a simple solution could be guardbanding. Guardbanding is inevitable, not only for ageing but also for PVT variations. However, adding more guardbanding would negate the advantage of using a smaller technology size. So we have to find an active protective approach as well as estimating, or sensing and reacting to degradations.

The idea of this work is to utilise the short idle periods in a processor, [6]. These are used to reverse the BTI stress rather than running empty loops. In our case study, the OpenRISC processor, the critical paths are in the adder and we can propagate patterns simply by loading the patterns into a register and executing an addition operation. This program should replace the idle task and should be executed whenever the operating system tries to schedule the idle task. In general, if the timing critical component is not the adder, then we have to replace the operation accordingly. If the critical paths are not controllable at the instruction level (e.g. in a control unit that may have many flip-flops) then we need an architectural solution rather than a software solution to propagate the patterns and currently we are working on this issue.

We also need to consider how process variations could affect the critical path ranking. If we get this wrong, we might heal a non-critical path and leave the real critical path unaffected. For this reason, we have

```
# Push r2,r3,r4 onto the stack
push {r2-r4}
# Stimulate the first pattern
l.movhi r2, 48CB
l.ori r2, r2, 3E67
l.movhi r3, D440
l.ori r3, r3, 7A4C
l.add r4, r2, r3
# Stimulate the second pattern
l.movhi r2, 2465
l.ori r2, r2, 9F2D
l.movhi r3, EA20
l.ori r3, r3, 3D3C
l.add r4, r2, r3
    .
    .
# Stimulate the eighth pattern
l.movhi r2, AC8C
l.ori r2, r2, 0318
l.movhi r3, A4DC
l.ori r3, r3, 938D
l.add r4, r2, r3
# Pop r2,r3,r4 and the program counter(
    pc) from the stack, then branch to
    the new pc.
pop {r2-r4,pc}
```

Figure 15: Optimised balancing program for saving the context of the interrupted program

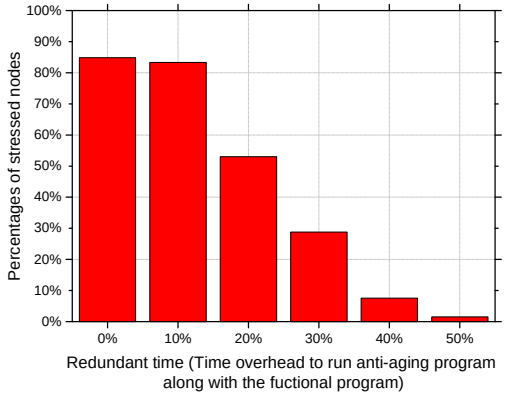

Figure 16: The effect on the stressed node percentage of running the BTI balancing program along with a "hello world" program with different run times of the balancing program.

to consider not only the critical path but also the nearly critical paths that could become critical with PVT and time-dependent variations. In our case study, we have predefined $(\theta + \Delta\delta_{pv})$ to be 20% of the maximum path delay at time zero (δ_0), as described in section 3.1, which covers the first 100 critical paths. However, we

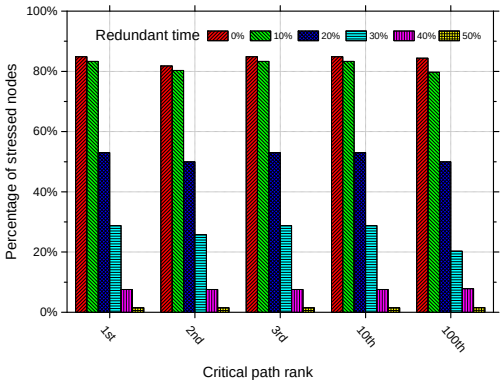

Figure 17: The effect of running the BTI balancing program along with a "Hello World" program on the stressed node percentage on different critical paths of the OpenRISC Processor with various run times of the balancing (anti-ageing) program.

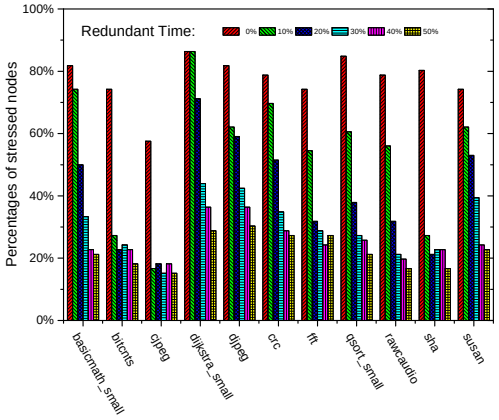

Figure 18: The effect of running BTI balancing program along with a different program from MiBENCH benchmarks on the stressed node percentage on the first critical path of the OpenRISC Processor with different run times of the balancing program

found that the first 100 critical paths share more than 92% of the cells with the most critical path and in this case balancing the most critical path also includes the 92% shared with the nearly critical paths. However, if the nearly critical paths do not share a big percentage of their cells with the first path, then we have to consider every single path in our analysis and generate patterns for them to balance signal probabilities in parallel. So, even with process variations, this technique would target the nearly-critical paths. If the nearly-critical paths do not share cells with the most critical path, it is important to define a threshold that considers the process and ageing variation contributions and to control these

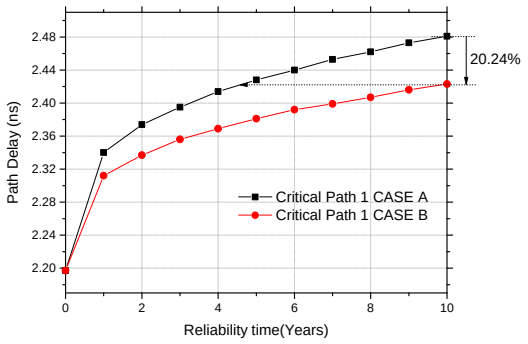

Figure 19: The advantage of running the anti-ageing program (CASE B) on the path delay of OpenRISC processor.

paths in parallel.

Finally, running a program to balance the BTI stress raises other design issues. Time overheads and power consumption need to be optimised by reducing the memory access or by using a program that has only immediate mode operands, as memory accesses increase the power consumption. Another issue that needs to be considered is when and for how long the program needs to run. The obvious answer is during the idle time of the processor but also we need to consider the operating system actions during the idle state.

6. Conclusion

Application-specific high-level ageing analysis has been done to find a technique for CMOS ageing mitigation. In this work, the stress probability has been found at the application level down to the gate level. A cross-layer mitigation technique is proposed to apply stress-relaxing patterns to the critical paths of a functional unit of a processor during idle times. This paper presents a two phase technique to mitigate the BTI ageing effects. The first phase generates balancing patterns. The second phase applies these patterns by executing a program to balance the stress on the critical paths of the embedded systems to alleviate BTI effects instead of running an empty process idle task. In future work we will apply this technique as an architectural solution to control the paths in the non-software-controllable units of the processor. Also, we will apply stress balancing in multiprocessors or many-processors systems. The operating system scheduler of these systems will have a higher opportunity to run anti-ageing programs by assigning an anti-ageing process to a processor in an idle state, concurrently with other processors that are running user or system tasks.

Acknowledgement

This work was supported by the Higher Committee of Education Development in Iraq (HCEDIraq) as a PhD scholarship program.

[1] M. Kamal, A. Afzali-Kusha, S. Safari, M. Pedram, Design of NBTI-resilient extensible processors, Integration, the VLSI Journal 49 (2015) 22–34. doi:10.1016/j.vlsi.2014.12.001.
URL http://linkinghub.elsevier.com/retrieve/pii/S016792601400087X

[2] J. Li, M. Seok, Robust and in-situ self-testing technique for monitoring device aging effects in pipeline circuits, 2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC) (2014) 1–6doi:10.1109/DAC.2014.6881529.
URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6881529

[3] K.-C. Wu, D. Marculescu, Joint logic restructuring and pin reordering against nbti-induced performance degradation, in: Proceedings of the Conference on Design, Automation and Test in Europe, European Design and Automation Association, 2009, pp. 75–80.

[4] Y. Wang, H. Luo, K. He, R. Luo, H. Yang, Y. Xie, Temperature-aware nbti modeling and the impact of standby leakage reduction techniques on circuit performance degradation, Dependable and secure computing, IEEE transactions on 8 (5) (2011) 756–769.

[5] Y. Wang, X. Chen, W. Wang, V. Balakrishnan, Y. Cao, Y. Xie, H. Yang, On the efficacy of input vector control to mitigate NBTI effects and leakage power, Proceedings of the 10th International Symposium on Quality Electronic Design, ISQED 2009 (2009) 19–26doi:10.1109/ISQED.2009.4810264.

[6] M. Arora, S. Manne, I. Paul, N. Jayasena, D. M. Tullsen, Understanding idle behavior and power gating mechanisms in the context of modern benchmarks on cpu-gpu integrated systems, in: 2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA), IEEE, 2015, pp. 366–377.

[7] S. Di Carlo, M. Gaudesi, E. Sanchez, M. Sonza Reorda, A functional approach for testing the reorder buffer memory, Journal of Electronic Testing 30 (4) (2014) 469–481. doi:10.1007/s10836-014-5461-9.
URL http://dx.doi.org/10.1007/s10836-014-5461-9

[8] T. Grasser, Bias temperature instability for devices and circuits, Springer Science & Business Media, 2013, Ch. 1, p. 6.

[9] S. Mahapatra, Fundamentals of Bias Temperature Instability in MOS Transistors: Characterization Methods, Process and Materials Impact, DC and AC Modeling, Vol. 52, Springer, 2015.

[10] A. Rahman, P. Bai, G. Curello, J. Hicks, C.-H. Jan, M. Jamil, J. Park, K. Phoa, M. S. Rahman, C.-Y. Tsai, et al., Reliability studies of a 22nm soc platform technology featuring 3-d tri-gate, optimized for ultra low power, high performance and high density application, in: Reliability Physics Symposium (IRPS), 2013 IEEE International, IEEE, 2013, pp. PI–2.

[11] B. Tudor, J. Wang, W. Liu, H. Elhak, Mos device aging analysis with hspice and customsim, Synopsys, White Paper.

[12] S. Bhardwaj, W. Wang, R. Vattikonda, Y. Cao, S. Vrudhula, Predictive modeling of the nbti effect for reliable design, in: Custom Integrated Circuits Conference, 2006. CICC'06. IEEE, IEEE, 2006, pp. 189–192.

[13] B. Tudor, J. Wang, Z. Chen, R. Tan, W. Liu, F. Lee, An accurate and scalable mosfet aging model for circuit simulation, in: Quality Electronic Design (ISQED), 2011 12th International Symposium on, IEEE, 2011, pp. 1–4.

[14] W. Wang, V. Reddy, A. T. Krishnan, R. Vattikonda, S. Krishnan, Y. Cao, An integrated modeling paradigm of circuit reliability for 65nm cmos technology, in: Custom Integrated Circuits Conference, 2007. CICC'07. IEEE, IEEE, 2007, pp. 511–514.

[15] J. B. Velamala, K. B. Sutaria, T. Sato, Y. Cao, Aging statistics based on trapping/detrapping: Silicon evidence, modeling and long-term prediction, in: Reliability Physics Symposium (IRPS), 2012 IEEE International, IEEE, 2012, pp. 2F–2.

[16] G. Jerke, A. B. Kahng, Mission profile aware ic design: a case study, in: Proceedings of the conference on Design, Automation & Test in Europe, European Design and Automation Association, 2014, p. 64.

[17] W. Wang, Z. Wei, S. Yang, Y. Cao, An efficient method to identify critical gates under circuit aging, 2007 IEEE/ACM International Conference on Computer-Aided Design (2007) 735–740doi:10.1109/ICCAD.2007.4397353.
URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4397353

[18] Y. Wang, X. Chen, W. Wang, V. Balakrishnan, Y. Cao, Y. Xie, H. Yang, On the efficacy of input vector control to mitigate NBTI effects and leakage power, Proceedings of the 10th International Symposium on Quality Electronic Design, ISQED 2009 (2009) 19–26doi:10.1109/ISQED.2009.4810264.

[19] S. Jin, Y. Han, L. Zhang, H. Li, X. Li, G. Yan, M-ivc: Using multiple input vectors to minimize aging-induced delay, in: 2009 Asian Test Symposium, IEEE, 2009, pp. 437–442.

[20] L. Zhang, R. P. Dick, Scheduled voltage scaling for increasing lifetime in the presence of nbti, in: Design Automation Conference, 2009. ASP-DAC 2009. Asia and South Pacific, IEEE, 2009, pp. 492–497.

[21] M. Omana, D. Rossi, N. Bosio, C. Metra, Low cost nbti degradation detection and masking approaches, IEEE Transactions on Computers 62 (3) (2013) 496–509.

[22] P. F. Butzen, V. Dal Bem, A. I. Reis, R. P. Ribas, Transistor network restructuring against nbti degradation, Microelectronics Reliability 50 (9) (2010) 1298–1303.

[23] P. Singh, E. Karl, D. Sylvester, D. Blaauw, Dynamic NBTI management using a 45 nm multi-degradation sensor, IEEE Transactions on Circuits and Systems I: Regular Papers 58 (9) (2011) 2026–2037. doi:10.1109/TCSI.2011.2163894.

[24] V. Huard, F. Cacho, F. Giner, M. Saliva, a. Benhassain, D. Patel, N. Torres, S. Naudet, a. Jain, C. Parthasarathy, Adaptive Wearout Management with in-situ aging monitors, 2014 IEEE International Reliability Physics Symposium (2014) 6B.4.1–6B.4.11doi:10.1109/IRPS.2014.6861106.
URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6861106

[25] W. Wang, V. Reddy, A. T. Krishnan, R. Vattikonda, S. Krishnan, Y. Cao, Compact modeling and simulation of circuit reliability for 65-nm cmos technology, Device and Materials Reliability, IEEE Transactions on 7 (4) (2007) 509–517.

[26] F. Firouzi, S. Kiamehr, M. B. Tahoori, NBTI mitigation by optimized NOP assignment and insertion, 2012 Design, Automation & Test in Europe Conference & Exhibition (DATE) (2012) 218–223doi:10.1109/DATE.2012.6176465.
URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6176465

[27] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, R. B. Brown, Mibench: A free, commercially representative embedded benchmark suite, in: Workload Characterization, 2001. WWC-4. 2001 IEEE International Workshop on, IEEE, 2001, pp. 3–14.

[28] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, et al., The gem5 simulator, ACM SIGARCH Computer Architecture News 39 (2) (2011) 1–7.

[29] N. Karimi, A. K. Kanuparthi, X. Wang, O. Sinanoglu, R. Karri, Magic: Malicious aging in circuits/cores, ACM Transactions on Architecture and Code Optimization (TACO) 12 (1) (2015) 5.

[30] D. Lorenz, G. Georgakos, U. Schlichtmann, Aging analysis of circuit timing considering nbti and hci, in: On-Line Testing Symposium, 2009. IOLTS 2009. 15th IEEE International, IEEE, 2009, pp. 3–8.

[31] J. Abella, X. Vera, A. González, Penelope: The NBTI-aware processor, Proceedings of the Annual International Symposium on Microarchitecture, MICRO (2007) 85–96doi:10.1109/MICRO.2007.11.