In-Field Logic Repair of Deep Sub-Micron CMOS Processors

Massoud Mokhtarpour Ghahroodi and Mark Zwolinski

Electronics and Computer Science, University of Southampton, Southampton, SO17 IBJ, UK, mz@ecs.soton.ac.uk

Abstract

Ultra Deep-Sub-Micron CMOS chips have to function correctly and reliably, not only during their early post-fabrication life, but also for their entire life span. In this paper, we present an architectural-level in-field repair technique. The key idea is to trade area for reliability by adding repair features to the system while keeping the power and the performance overheads as low as possible. In the case of permanent faults, spare blocks will replace the faulty blocks on the fly. Meanwhile by shutting down the main logic blocks, partial threshold voltage recovery can be achieved which will alleviate the ageing-related delays and timing issues. The technique can avoid fatal shut-downs in the system and will decrease the down-time, hence the availability of such a system will be preserved. We have implemented the proposed idea on a pipelined processor core using a conventional ASIC design flow. The simulation results show that by tolerating about 70% area overhead and less than 18% power overhead we can dramatically increase the reliability and decrease the downtime of the processor.

1 Introduction

Semiconductor manufacturing continues to provide smaller feature sizes, resulting in lower power, higher density, and lower cost per function. While this trend is positive, there are a number of negative side effects, including increased semiconductor parameter variability, increased sensitivity to soft errors, and lower device yields. The lifetime of the next generations of devices is also decreasing due to lower reliability margins and shorter product lifetimes. As demonstrated in [1], the average Mean Time To Failure (MTTF) of a modern-day superscalar processor has dropped by approximately 4× between the 180nm and 65nm technology nodes. Design for reliability and resilience in the long term has been identified as one of the major challenges and a key focus area in upcoming years in International Technology Roadmap for Semiconductors (ITRS) 2011 report [2].

Various reliability factors, such as safety and robustness, as well as resilience to malfunctions need to be addressed in dependable systems [3] [4]. Infrastructure systems, such as the internet, banks, stock markets, and electrical power networks, require dependability to ensure social stability and in many cases this must be 24 hours per day, 365 days per year. Providing this level of dependability and availability and making reliable electronic
systems out of unreliable CMOS components, while keeping overheads as low as possible are major challenges.

In this paper, we present a technique to retain the functionality of CMOS circuits and processors in the presence of reliability issues that lead to permanent faults, while keeping the overheads affordable. We evaluate the reliability of the proposed techniques using Markov models and the NASA SURE reliability program.

2 Major Reliability Issues in Ultra Deep-Sub-Micron CMOS

2.1 Time-Dependent Dielectric Breakdown (TDDB) or “Wear-Out”

The continued scaling of MOSFET devices requires ultra-thin gate dielectrics to control the short channel effect. This has reduced the reliability of the dielectric layer leading to dielectric breakdown over time due to the formation of a conductive path through the oxide to the substrate [5]. Studies have shown that electron fluence (current) and energy (voltage) are the driving factors for wear-out and eventual breakdown [6]. Oxide breakdown can be categorized into hard breakdown (HBD) and soft breakdown (SBD). HBD is potentially fatal to the whole design. SBD events do not cause immediate failures of the CMOS device but they will affect the performance of the circuit [7].

Typically after soft breakdown the leakage current is only slightly larger than the pre-stress tunnelling characteristic. After some time the leakage current can continue to increase, finally resulting in a hard breakdown. Even though TDDB has been studied for over three decades, the exact physical mechanism remains unclear. What is known is that the process is driven by voltage and temperature [6]. To limit the thermal damage caused by TDDB, the power dissipation needs to be reduced. To do so, either the supply voltage needs to reduced or the percolation path current needs to be decreased.

2.2 Negative Bias Temperature Instabilities (NBTI) & Hot-Carrier Injection (HCI)

Negative bias temperature instabilities (NBTI) in pMOSFETs are considered as major reliability issues in Ultra Deep-Sub-Micron analogue and digital integrated circuits [8] [6]. This phenomenon occurs when a PMOS transistor is turned on at high temperatures (usually between 100 °C and 150 °C). In other words, when the gate of a PMOS transistor is negatively biased with respect to the substrate, reduced drive current and threshold voltage (Vth) shifts result [6] [9].

Another principal degradation issue of MOSFETs is hot-electron-induced depassivation of the Si-SiO2 interface that limits the operating lifetime of the transistors [10]. Hot-carrier injection degrades the transconductance of CMOS devices and shifts some of the key device parameters such the threshold voltage.

The threshold voltage can partially recover when the gate bias is switched to 0V. However this Vth recovery is logarithmically time-dependent [6] [11]. As illustrated in [12] [13] and [6], a substantial recovery in Vth is observed when the electrical stress is interrupted [14].
3 Reliability, Dependability and Availability

Reliability is typically quantified as MTBF (Mean Time Between Failure) for repairable devices and MTTF (Mean Time To Failure) for non-repairable devices. In repairable systems, MTBF is the sum of the mean times of MTTFs of the device plus the MTTR (Mean Time To Repair/Restore) [15].

A system is assumed to function properly during most of its lifetime. One way to determine if the level of faults or system malfunctions is within the acceptable range is through the utilization of an availability factor. Availability ($A$) can be defined as [15] [16]:

$$A = \frac{\text{Uptime}}{\text{Total time}} = \frac{\text{Up time}}{\text{Up time} + \text{Down time}}$$

(1)

To practically assess the availability factor of a system, the temporal elements should be replaced by other elements that represent the required functionality of the system. Depending on the situation and the desired purposes from the system, the availability factor should be defined with respect to effective Up time (work time) and Down time (repair/maintenance time). This definition of availability is called ‘inherent’ availability [15] and is usually represented by:

$$A = \frac{\text{MTTF}}{\text{MTTF} + \text{MTTR}} = \frac{\text{MTTF}}{\text{MTBF}}$$

(2)

For simplicity, two extra timing components are ignored here: waiting-for-maintenance time and recovery-time. This is because in an ideal world, these two timing components are zero. So to compute MTTR only the maintenance time for correction is considered.

To increase the reliability of systems, various fault-tolerant techniques based on redundancy in time/spatial domains have been used. For hardware-based redundancy, majority voting redundancy (TMR), stand-by redundancy, and hybrid modular redundancy (HMR) are the major techniques. It is generally believed that N-modular redundancy systems are more reliable than stand-by redundancy systems. For instance, the reliability of an N-modular redundancy system such as TMR with three redundant components (with equal reliability $R$ for each component and assuming that fault-detection coverage is 100%) can be formulated as:

$$R_{TMR} = R^3 + 3R^2(1 - R) = -2R^3 + 3R^2$$

(3)

On the other hand, the reliability of stand-by redundancy systems with two redundant components (with equal reliability $R$ for each component and assuming that fault-detection coverage is 100%) can generally be expressed as:

$$R_{\text{stand-by}} = R^2 + 2R(1 - R) = -R^2 + 2R$$

(4)

The reliability of stand-by redundancy systems is higher than the reliability of TMR systems for all $R$. However this can be a deceptive comparison: TMR systems have higher fault-detection coverage with the feature of data
comparison by nature, and higher reliability especially for shorter \( t \). Therefore, TMR systems are widely used for life-critical applications with relatively shorter mission times. Also the overheads of such systems particularly in terms of power consumption are extremely high.

4 In-Field Repair in CMOS Circuit Design

4.1 Motivation

As discussed, reliability issues such as Time-Dependent Dielectric Breakdown (TDBB), Hot-Carrier Injection (HCI) and Negative Bias Temperature Instability (NBTI) degradations are inevitable in the ultra deep-sub-micron era. These phenomena can manifest themselves as performance degradation, timing errors or hard-errors in the chip, leading to a total, permanent failure of the processor [17].

The challenge is to increase the maintainability of a CMOS system or a circuit, once a malfunction is detected. Since these major reliability issues are inevitable, the goal is to decrease the downtime, hence the MTTR, of such systems. The “MTBF Countdown Clock” does not start until the device is under stress i.e. the hardware is powered up. In other words, device degradation does not happen if the devices are permanently “off” because there is no electrical field in the transistor channels.

Moreover as reported in [6] and [11], some of the impacts of the aforementioned reliability issues, such as the threshold voltage shifts in CMOS devices, can be recovered by removing the stress from the devices and turning them off. Taking all these facts into account, we propose an idea for logic in-field repair to increase the availability factor by providing logic spare-blocks using within-chip cold swapping. Our scope will be the logic parts of a design rather than the memory blocks such as RAM cells, since we can protect the memory blocks using ECC methods.

In the next section we present an architectural solution for increasing the reliability of processors using logic spare-blocks. Here the key point is trading area for reliability and trying to keep the power and the performance overheads as low as possible while keeping the processor running even in the presence of a permanent fault.

4.2 In-Field Logic Repair

Hard-errors can be fatal if the whole system shuts down. If a set of failures prevents the system from carrying out all applications, a subset of less important applications can be dropped while the more important applications can be kept alive. This concept is known as graceful degradation [18] [19]. The main idea is to provide logic spare-blocks in the architecture in such a way that in the event of any permanent faults or defects, the faulty logic block can be replaced by the spare-block as depicted in Fig. 1 to maintain the same functionality or with graceful degradation preserve the vital functions of the faulty logic block.
4.3 Sphere of Replication and Levels of Granularity

The sphere of replication determines the logical boundary within which the logic blocks are physically replicated. The size and the level of granularity of the sphere of replication can vary widely and will lead to differences in *Implementation Complexity versus Availability* trade-offs. A spare-block can vary from a simple logic gate to a whole processor core; a spare-block can be an exact replica of a logic block or a functionally equivalent structure of that logic block but with a different canonical form or physical implementation, as proposed, for example, by Reviriego *et al* [20].

![Figure 1: General idea of Logic In-Field Repair: Spare logic blocks can be exact replicas of relevant logic blocks or functionally equivalent structures or have reduced functionalities.](image)

A spare-block can also be a simplified version of a certain logic block that is smaller in size and provides a reduced level of functionality or service in the presence of a permanent fault or defect on the main logic block rather than failing completely. In other words, the spare-blocks can also be designed to be utilized in the graceful degradation phase if any permanent fault or defect happens.

Obviously the easiest approach could be to take a whole processor core as a spare-block in the replication procedure. In the event of a permanent fault on such a dual core processor, the faulty core would be shut down and the other core would be powered up. However doing this at core-level would cost many clock cycles due to initialization and re-execution of several instructions, while switching between spare-blocks at pipeline level would cost just a few clock cycles hence the system would have a higher availability and lower downtime, particularly in the case of safety-critical or non-stop computing applications. Therefore, in this work, we define the logic
spare-blocks as pipeline stages in a processor core in favour of increased availability and decreased downtime.

It is noteworthy to mention that there is a huge difference in power consumption between the proposed method and N-modular redundancy methods such as DMR or TMR. Here, only the main core, the switches and the controller are always “on” and the spare-blocks are turned on only when they are needed. By the time a logic spare-block is turned on, its faulty/defective counterpart will be turned off to keep the power consumption overheads as low as possible. Moreover because the spare-blocks are always “off”, they will be immune to ageing, NBTI and HCI effects. The down-time for the system will also be much lower than replacing or swapping a faulty chip with a new one, since all the replacements will happen within the chip.

Since the switching between logic blocks is happening within the chip, the delays and the downtime are limited to a few clock cycles. Therefore the reliability of such system can be considered as equal to the reliability of standby redundant systems with two redundant components. However the power overheads of such in-field repair system is significantly lower that any stand-by or TMR system, thanks to the within-chip cold swapping mechanism.

The reliability of such In-Field Repair system $R_{IFR}$, with $s$ number of spare-blocks and $R_b$ as the reliability of each block, whether an active block or a spare-block – assuming that the fault detection and the switching mechanisms are flawless – can be expressed as:

$$R_{IFR} = 1 - (1 - R_b) (s+1)$$  \hspace{1cm} (5)

$R_{IFR}$ is a function of the number of spare-blocks. Too many spare-blocks can have a detrimental effect on the reliability of such system accompanied by unacceptable overheads.

To evaluate the reliability of such system, the NASA SURE program has been used. SURE is a reliability analysis tool that can be used to compute and solve semi-Markov models based on a bounding theorem \[21\]: “The probability $D(t)$ of a system with the mission time $T$ to enter a particular death state is bounded as: Lower-bound $< D(t) <$ Upper-bound”.

The SURE tool is particularly useful in analyzing the fault-tolerance features of reconfigurable systems. To investigate the reliability of mission-critical computer systems, the probabilistic boundaries are computed to be close enough (usually within 5% of each other). Even for complex systems, the SURE bounding theorems have algebraic solutions which are computationally efficient. SURE can optionally take a specified parameter as a variable over a range of values, which can automate the sensitivity analysis of such systems \[21\].

To analyse the Markov reliability model of the system in SURE, all of the states and the transitions between them should be defined. Usually during early analysis of the processes of a system, there is no experimental reliability data available. So, for simplicity, we have to start with some assumptions about the not-exactly-known issues in the process, components, mission-time and the failure rates. For instance, let us assume a TMR system in which, each processor has a failure rate $\lambda$ over the range of 1E-6 to 1E-2 with a mission time of 1000 hours.

The SURE program calculates an upper and a lower bound on the probability of system failure. As noted, these
bounds are usually within 5% of each other, and thus they usually provide an accurate estimate of system failure. The plot in Fig. 2 shows the probability failure of a TMR system. For an IFR system with the same failure rate $\lambda$ and the same mission time, the probability of failure is about an order of magnitude less, Fig. 3.

![SURE plot for TMR systems - Mission time: 1000 hours](image)

4.4 In-Field Logic Repair at Pipeline Level

We have evaluated the proposed in-field logic repair method on a simple, custom processor core, using a 45nm technology and a conventional ASIC design flow.

The 32 bit processor core has a 3 stage pipeline and each pipeline stage is taken as one logic block, Fig. 4. Pipelines stages are connected to one another through the switch boxes. The switch boxes have been implemented using multiplexor cells available in the standard cell library. Parity error detection has been used with one parity bit for every eight bits of signals between the pipeline stages. The parity circuit has been added before the pipeline registers for every pipeline stage. The cost of the error coding and decoding logic is typically amortized over many bits.

Note that the idea is to replicate the logic parts and not the register file or any other memory block. The memory blocks are protected by ECC as is usual in such designs. The multiplexer-based switch boxes, the parity error detection circuits and the controller unit have been added to the core at RTL and the power-gating circuit has been added at the Place & Route stage. Any parity error lasting for more than a certain number of clock cycles...
is considered a permanent fault. Hence the faulty pipeline stage will be shut down, the pipeline will be flushed and the spare pipeline block will be turned on by the controller unit. The controller unit is also in charge of power management to avoid IR drop and simultaneous switching capacitance by turning on the blocks in a daisy chain style, hence avoiding any in-rush current. The controller has been implemented using the two-rail checker scheme.

The switch boxes comprise two $2\times1$ multiplexor cells for each bit, Fig 5. They are added as VHDL structural blocks and instantiated in the RTL code connecting the pipeline blocks with one another.

### 4.4.1 Self-Checking Controller

The Controller is based on a duplication and comparison scheme. The duplicated copy of the controller has complemented outputs. The comparison is done using totally self-checking (TSC) two-rail checkers (TRC). Both copies of the controller function receive the same inputs.

To differentiate transient faults from permanent faults, two different techniques are used. 1) Adjustable counters in the controller block. Whenever there is an error for more than certain number of clock cycles the error will be assumed to permanent. 2) Permanent faults will be last than one clock cycle, hence the input and output values of flip-flops can be compared within one clock cycle to identify transient effects.

In the event of any permanent delay fault in any logic blocks (any pipeline stages in our case), the controller flushes the pipeline, avoiding the ongoing instruction to commit, turns off the faulty block, turns on the spare-block.
and re-runs the instruction. In this case, the system only loses a few clock cycles rather than longer period of time to swap the faulty chip with a new one hence avoiding a total shut-down of such system.
The reliability of such an architecture is equal to:

\[ R_{IFR(pipeline)} = |R_p^2 + 2CR_p(1 - R_p)|R_{sw}R_{ctrl} \]  

(6)

where \( R_p \) is the reliability of each logic block (pipeline stages in this case) with C as the Fault Coverage factor. 
\( R_{ctrl} \) is reliability of the controller and \( R_{sw} \) is the reliability of the switch boxes. Here is the assumption is fault coverage is 100%.

Fig. 6 shows the Markov model of this system. The system begins in state (1) where all components are operational. Either of two processors or the switch-boxes and the controller could fail. \( \lambda_p \) is the failure rate of the currently on-line pipeline stages that would make a whole functional core running, \( \lambda_p \) is also considered for the processor pipeline stages which are currently off-line, and \( \lambda_{sw} \) is the failure rate of the switch-boxes with \( \lambda_{ctrl} \) as the failure rate of the controller. Note that the sum of the rates of all failure transitions from state (1) add up to the sum of the failure rates of all non-failed components \( (\lambda_p + \lambda_{sw} + \lambda_{ctrl}) \). This property should always be true for all operational states of a reliability model.

\[ \begin{align*}
1 & \xrightarrow{\lambda_p + \lambda_{sw} + \lambda_{ctrl}} 2 \\
2 & \xrightarrow{F(t)} 3 \\
3 & \xrightarrow{\lambda_p + \lambda_{sw} + \lambda_{ctrl}} 4
\end{align*} \]

Figure 6: Markov model of IFR at Pipeline level

4.5 Results

4.5.1 Overhead Comparisons

As shown in Fig. 7 the total area overhead is around 72% on this simple core, because all the logic blocks are duplicated and the controller unit and the switch boxes are added. Because the spare-blocks are always kept off-line, the dynamic power overhead is less than 18% and the leakage power overhead is 14%. This is due to the controller unit and the switch boxes. The 9% delay overhead is caused by the error detection mechanism and the
switch boxes.

Figure 7: Area, Power and Performance Comparisons: A) Original Core B) In-Field Repair Core

In the testing phase of the circuit, the reliability issues and ageing effects have been modelled as single stuck-at faults and delay faults injected in the main pipeline decode and execute units. The core clock frequency is 100 MHz. A pre-defined set of instructions has been run on both of the original core and the IFR core. Having the same level of permanent faults in the original core could cause a fatal shut-down of the core, while the IFR core can get back to its normal functioning status within approximately 1 microsecond, as shown in Table 1.

| Design     | Faults                | Recovery Time (us) |
|------------|-----------------------|--------------------|
| IFR Core   | Stuck-at (Decode Unit)| 0.82               |
| IFR Core   | Stuck-at (Execute Unit)| 1.00              |
| IFR Core   | Delay (Decode Unit)   | 1.20               |
| IFR Core   | Delay (Execute Unit)  | 1.51               |

As shown in Fig. 8, the contribution of logic components (including all the pipeline stages: Predecode unit, Decode unit and Execute unit) to the total area and power consumption is less than 40% for this specific simple core. It is noteworthy to mention that the contribution of logic to the total area is decreasing in modern processors since the sizes of on-chip memory blocks such Caches are increasing rapidly, therefore applying this technique to more realistic and modern processors will result in proportionally lower area overhead. As reported in the ITRS 2011 roadmap report [2], the trend in processor chips shows that the contribution of memory parts in terms of area is predicted to be an order of magnitude higher than logic, therefore the area overheads of the proposed technique will be justifiable.

4.5.2 Reliability Comparisons

To compare the reliability of such a system, the SURE program has been used. The probability failure of a simplex core with a failure rate of $\lambda$ in the range of 1E-6 to 1E-2 with a mission time of 1000 hours is shown in Fig.
Using the same failure rate $\lambda$ and the same mission time, the graph in Fig. 10 shows that the probability of failure of an IFR system is much lower than for a simplex core (depicted in Fig. 9) particularly for components with higher failure rates. In other words, for components/processors with higher failure rates, a simplex system has a higher failure probability than an IFR-like system. Also considering the range of 1E-6 to 1E-2 for the failure rates, the probability failure of a simplex core starts at 0.001 for a failure rate of $10^{-6}$, while the probability failure of the IFR version of the core starts at $10^{-6}$. For failure rates around 0.001, the probability failure of the simplex core is very close to 1, however for the IFR core it is approximately 0.3. Therefore, overall, it can be seen that the IFR core is far more reliable compared to the simplex one, at the cost of tolerating minor power-performance overheads plus area overheads.

5 Conclusions

In this paper a logic in-field repair technique for UDSM CMOS processor design has been presented. The “MTBF Countdown Clock” does not start until the device is under stress i.e. the hardware is powered up and device degradation does not happen if the devices are permanently “off”. By trading area for increased reliability and by providing spare-logic blocks which are off and will be turned on in case a logic block loses its functionality, any fatal shut-down will be prevented.

We have implemented the proposed idea on a pipelined processor core using the conventional ASIC design flow. The simulation results show that by tolerating about 70% area overhead and less than 18% power overhead we can dramatically increase the reliability and decrease the downtime of the processor. As reported in ITRS 2011 roadmap report, the trend in processor chips shows that, the contribution of memory parts in terms of area is predicted to be an order of magnitude higher than logic, therefore the area overheads of the proposed technique will be justifiable. The reliability benefits of such architecture has been assessed using Markov models and the NASA SURE reliability program.

Meanwhile in the turned-off blocks, the threshold voltage can partially recover to its initial value when the gate bias is switched to 0V, therefore a relative Vth recovery can also be rewarded that can alleviate the probability of
any delay faults if the used logic-blocks or the group of critical-paths are needed to be turned on again due to the same ageing-induced delay faults in the secondary spare-blocks.

References

[1] J. Srinivasan, S. V. Adve, P. Bose, and J. A. Rivers, “Lifetime reliability: toward an architectural solution,” *Micro, IEEE*, vol. 25, no. 3, pp. 70 – 80, May-June 2005.

[2] International Technology Roadmap for Semiconductors, “International technology roadmap for semiconductors semiconductor industry association, www.itrs.net/links/2011itrs/home2011.htm (2011), accessed on May 1 2012,” *Semiconductor Industry Association*, vol. 2012, 2011.

[3] J. C. Laprie;, “Dependable computing and fault tolerance : Concepts and terminology,” in *Fault-Tolerant Computing, 1995. ‘Highlights from Twenty-Five Years’.*, Twenty-Fifth International Symposium on, jun 1995, p. 2.

[4] N. Kanekawa, E. H. Ibe, T. Suga, and Y. Uematsu, “Dependability in electronic systems: Mitigation of hardware failures, soft errors, and electro-magnetic disturbances.” Springer, Nov. 2010.
Figure 10: SURE plot for Probability Failure of the IFR core

[5] K. C. Wu, M. C. Lee, D. Marculescu, and C. S. Chieh, “Mitigating lifetime underestimation: A system-level approach considering temperature variations and correlations between failure mechanisms,” pp. 1269 –1274, March 2012.

[6] D. M. Fleetwood and R. D. Schrimpf, Eds., Defects in Microelectronic Materials and Devices. CRC Press, Nov. 2008.

[7] H. Nan, L. Li, and K. Choi, “TDDB-based performance variation of combinational logic in deeply scaled CMOS technology,” in Quality Electronic Design (ISQED), 2012 13th International Symposium on, March 2012, pp. 328 –333.

[8] 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, Jul 2003.

[9] A. T. Krishnan, V. Reddy, S. Chakravarthi, J. Rodriguez, S. John, and S. Krishnan, “NBTI impact on transistor and circuit: models, mechanisms and scaling effects [MOSFETs],” in Electron Devices Meeting, 2003. IEDM
[10] K. Hess, L. F. Register, W. McMahon, B. Tuttle, O. Aktaş, U. Ravaioli, J. W. Lyding, and I. C. Kizilyalli, “Theory of channel hot-carrier degradation in MOSFETs,” *Physica B: Condensed Matter*, vol. 272, no. 14, pp. 527 – 531, 1999.

[11] M. Denais, C. Parthasarathy, G. Ribes, Y. Rey-Tauriac, N. Revil, A. Bravaix, V. Huard, and F. Perrier, “On-the-fly characterization of NBTI in ultra-thin gate oxide PMOSFET’s,” in *Electron Devices Meeting, 2004. IEDM Technical Digest. IEEE International*, Dec. 2004, pp. 109 – 112.

[12] M. Ershov, S. Saxena, H. Karbasi, S. Winters, S. Minehane, J. Babcock, R. Lindley, P. Clifton, M. Redford, and A. Shibkov, “Dynamic recovery of negative bias temperature instability in p-type metal semiconductor field-effect transistors,” *Applied Physics Letters*, vol. 83, no. 8, pp. 1647 –1649, Aug 2003.

[13] G. Chen, K. Y. Chuah, M. F. Li, D. S. H. Chan, C. H. Ang, J. Z. Zheng, Y. Jin, and D. L. Kwong, “Dynamic NBTI of PMOS transistors and its impact on device lifetime,” in *Reliability Physics Symposium Proceedings, 2003. 41st Annual. 2003 IEEE International*, March-April 2003, pp. 196 – 202.

[14] M. Denais, A. Bravaix, V. Huard, C. Parthasarathy, C. Guerin, G. Ribes, F. Perrier, M. Mairy, and D. Roy, “Paradigm shift for NBTI characterization in ultra-scaled CMOS technologies,” in *Reliability Physics Symposium Proceedings, 2006. 44th Annual., IEEE International*, March 2006, pp. 735–736.

[15] M. Lazzaroni, *Reliability Engineering: Basic Concepts and Applications in ICT*, 1st ed. Springer, Sep. 2011.

[16] P. K. Lala, *Self-Checking and Fault-Tolerant Digital Design*. Morgan Kaufmann, 2000.

[17] R. Mishra, S. Mitra, R. Gauthier, D. Ioannou, D. Kontos, K. Chatty, C. Seguin, and R. Halbach, “On the interaction of ESD, NBTI and HCI in 65nm technology,” in *Reliability physics symposium, 2007. proceedings. 45th annual. ieee international*, April 2007, pp. 17 –22.

[18] M. Glass, M. Lukasiewycz, C. Haubelt, and J. Teich, “Incorporating graceful degradation into embedded system design,” in *Design, Automation Test in Europe Conference Exhibition, 2009. DATE ’09.*, April 2009, pp. 320 –323.

[19] B. Randell, P. Lee, and P. C. Treleaven, “Reliability issues in computing system design,” *ACM Comput. Surv.*, vol. 10, no. 2, pp. 123–165, Jun. 1978. [Online]. Available: [http://doi.acm.org/10.1145/356725.356729](http://doi.acm.org/10.1145/356725.356729)

[20] P. Reviriego, C. Bleakley, and J. Maestro, “Diverse double modular redundancy: A new direction for soft-error detection and correction,” *Design Test, IEEE*, vol. 30, no. 2, pp. 87–95, April 2013.

[21] R. W. Butler and S. Johnson, “Techniques for modeling the reliability of fault-tolerant systems with the Markov state-space approach,” *NASA Reference Publication*, vol. 1348, 1995.