An FPGA-Based Transient Error Simulator for Resilient Circuit and System Design and Evaluation

Chia-Hsiang Chen, *Student Member, IEEE*, Shiming Song, *Student Member, IEEE*, and Zhengya Zhang, *Member, IEEE*

***Abstract*—Error detection and correction (EDAC) has become more important with continued device scaling. We propose a field-programmable gate array (FPGA)-based simulator to accelerate the transient simulation of pipeline-level EDAC circuits and their interactions with circuits under test (CUTs). The simulator incorporates the CUT delay profile, the CUT error profile, and the EDAC model. The FPGA-based simulator captures the fine-grained interactions between the CUT and EDAC for the evaluation of the effectiveness of EDAC and its tuning. The simulator is constructed based on parameterized models, making it general purpose and widely applicable. We demonstrate the capability of this simulator in the evaluation of two popular pipeline-level EDAC designs, i.e., preedge EDAC and postedge EDAC, using synthesized processors that operate under generic error and noise models. The proposed error simulator uncovers key insights to help guide EDAC designs.**

***Index Terms*—Error analysis, fault simulation.**

I. Introduction

THE scaling of device geometries improves the performance of digital integrated circuits, but it also leads to the growing challenges in reliability and variability [1]. To deal with variations and aging problems, increasingly pessimistic margins have been applied in circuit and system designs, which results in performance degradation and energy wasted. Meanwhile, as critical charge is reduced, devices are becoming more susceptible to transient noise and soft errors, worsening the system reliability.

To reduce the pessimistic margins and enhance the robustness of deep-submicron designs, efficient error detection and correction (EDAC) circuits have been proposed to detect and correct transient errors. Common EDAC circuits are deployed at pipeline boundaries to detect transient errors that appear at the output of the combinational circuits of a pipeline stage [2]–[5]. Error detection triggers correction mechanisms, such as pipeline halt or instruction reissue.

Manuscript received June 10, 2014; revised August 8, 2014 and November 3, 2014; accepted December 16, 2014. Date of publication December 25, 2014; date of current version April 23, 2015. This work was supported in part by the Intel Corporation and in part by the Defense Advanced Research Projects Agency under Cooperative Agreement HR0011-13-2-0006. This brief was recommended by Associate Editor Y.-B. Kim.

The authors are with the Department of Electrical Engineering and Computer Science, College of Engineering, University of Michigan, Ann Arbor, MI 48109-2122 USA (e-mail: uchchen@umich.edu; shisong@umich.edu; zhengya@eecs.umich.edu).

Color versions of one or more of the figures in this brief are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSII.2014.2386251

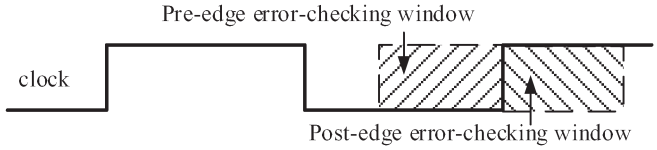

Fig. 1. Preedge checking and postedge checking for error-resilient designs.

Pipeline-level EDAC circuits can be classified into two groups based on error detection, i.e., preedge EDAC that detects an error prior to the clock sampling edge illustrated in Fig. 1 [2], [3] and halts the pipeline for error recovery, and postedge EDAC [4], [5] that detects an error after the sampling edge and reissues the instruction to flush the error. These EDAC circuits improve the reliability and reduce the design margins, but it is often difficult to precisely evaluate in the design time their effectiveness as it depends on the circuits under protection and the pattern of transient errors.

Conventional software-based transient circuit simulation becomes very slow when simulating large systems with the addition of transient errors. A field-programmable gate array (FPGA) has been used in accelerating fault emulation, with examples including an FPGA emulator for evaluating the sensitivity of single-event upsets [6] and for evaluating the sensitivity of single-event transients (SETs) [7]–[9]. The FPGA-based SET emulation is directly relevant to this brief. SET simulation or emulation usually consists of two steps: 1) the time-accurate gate-level propagation of SET errors to the first storage element; and 2) the cycle-accurate register-transfer level (RTL) propagation of SET errors across clock cycles. The gate-level SET propagation (step 1) can be done in simulation [7] or FPGA emulation using a quantized delay model [8], and the RTL SET propagation (step 2) can be done in FPGA emulation [7], [9]. These past works have demonstrated good accuracy, coverage, and orders of magnitude improvement in speed compared with software simulations.

This brief is specifically focused on speeding up the transient simulation of the interactions between circuits under test (CUTs) and EDAC. Since EDAC corrects errors in the CUT and prevents them from propagating, the RTL error propagation across clock cycles (step 2) becomes unnecessary, thus simplifying the simulator or emulator design. On the other hand, EDAC is designed to correct all (or almost all) transient errors in the CUT; thus, the emulation needs to reach a very low bit error rate. Although performing gate-level error propagation in the CUT (step 1) on the FPGA is entirely feasible, we propose to abstract the CUT to a delay profile and an error profile, and

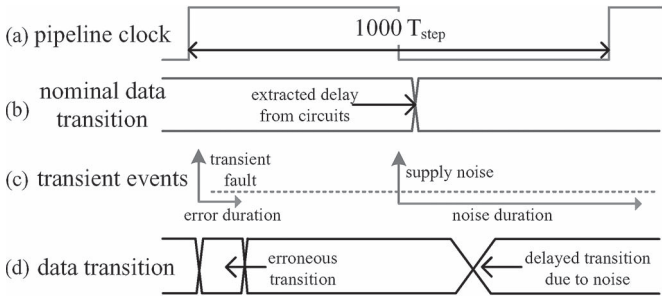

Fig. 2. Timing charts of the FPGA-based transient error simulation. (a) Pipeline clock. (b) Nominal data path delay. (c) Injected transient events (errors and noise). (d) Data transition as the result of (b) and (c).

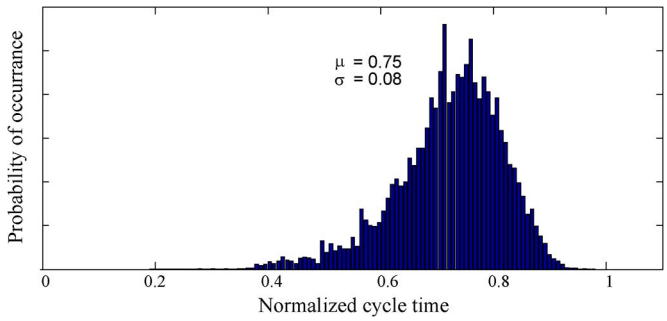

Fig. 3. Delay distribution extracted from a synthesized CORDIC processor.

EDAC to an EDAC model. The abstraction reduces the FPGA resource usage, allowing us to map many pipeline stages on the FPGA and capture the very low error-rate performance of complex systems incorporating EDAC.

In the proposed paradigm, the CUT delay profile and error profile will be first obtained from software simulation, such as a gate-level static timing analysis tool, or emulation [7], [8], whereas the lengthy transient simulation of the CUT and EDAC will be done on the FPGA that exercises the CUT delay profile, the CUT error profile, and the EDAC model, capturing the interactions between the CUT errors and EDAC for good coverage. The profiles and models of the FPGA-based simulator are programmable, making it a versatile and general-purpose error simulation platform.

II. Transient Error Simulator

This brief is focused on the simulation of systems incorporating pipeline-level EDAC. We will use CUT to refer to the combinational circuits of one pipeline stage, which is followed by a pipeline register incorporating an EDAC circuit that attempts to detect and correct errors at the CUT output. For error simulation, the continuous tracking of errors can be simplified to the monitoring of events, such as the correct output of a CUT becoming available, an error occurring at the output of a CUT, etc. These events are generated from the CUT delay profile and the CUT error profile that are obtained from software simulation or hardware emulation.

The proposed FPGA-based transient simulation operates at fine-grained time steps, and it allows events to occur at these finer time steps. We use FPGA clock period $T_{\rm step}$ as the unit time step. For example, if one chooses a simulation time step of 1 ps and a clock cycle period of 1 ns, the 1-ps time step is mapped to 1 $T_{\rm step}$ on the FPGA, and the 1-ns cycle period is mapped to 1000 $T_{\rm step}$. This setup permits a simulation throughput of 1/(1000 $T_{\rm step}$). An illustration of the timing is presented in Fig. 2. Note that the accuracy of transient simulations depends on the time step size, i.e., smaller time steps yield more accurate results but lower the simulation throughput.

A. Delay Profile and Error Profile

Unlike the conventional FPGA emulation, the CUT is not implemented on the FPGA, but instead, its delay profile and error profile are stored on the FPGA. The FPGA is used as a simulator rather than an emulator.

The CUT delay profile is obtained from circuit simulators such as Simulation Program with Integrated Circuit Emphasis (SPICE) or delay-annotated RTL simulation using a test bench that exercises realistic workloads or standard benchmarks. The CUT delay profile is stored on the FPGA in two possible formats, i.e., either as a histogram or as an application trace. Fig. 3 shows a delay histogram of a coordinate rotation digital computer (CORDIC) processing stage that is obtained from the delay-annotated RTL simulation using a test bench that exercises all input patterns. The histogram is divided into delay bins, and the probability of exercising a delay in each bin is stored on the FPGA. The application trace is stored as a list of instruction-by-instruction delays. The application trace is specific to one application or benchmark. When running the simulation on the FPGA, we will select data path delays from the histogram based on the probability of each bin or run through the application trace. The histogram format is compact and can be used for long error simulations, and the application trace is used for specific tests and corner cases.

The CUT error profile consists of the error probability and the distribution of the transient error duration of the CUT output node. The CUT error profile is obtained using circuit simulators with fault injection at different points in the CUT. Extensive work has been done in the past on error injection and error propagation using simulation and hardware emulation approaches [8], [9]. This brief makes use of these approaches to build the error profile.

In our proof-of-concept simulator, we implement several generic error profiles for the SET, the coupling noise, and the voltage droop. The SET is known for its random occurrence, and it causes a transient upset to a circuit node for a period of time, as shown in Fig. 2. For a given SET probability on the CUT output node, a SET event generator can be implemented based on a linear-feedback shift register (LFSR): the LFSR generates pseudorandom numbers, and a SET event is generated when the random number matches a given constant. The length of the constant can be adjusted to tune the SET rate. Once a SET event is generated, it lasts for a duration described by a distribution that is stored as a histogram on the FPGA. The coupling noise effect is similar to the SET, and it is modeled in the same way.

The voltage droop is modeled as a sinusoidal voltage fluctuation around a nominal value [10]. The voltage droop model

is implemented as a randomly generated event that changes the CUT delay for the duration of the droop. The delay change is described by a sinusoid of a selected period and peak magnitude. Additional models can be added to capture more relevant sources for more accurate simulations. For example, since the circuit activity can be highly correlated with the voltage noise [10], a CUT activity profile can be first obtained through simulation and then used to determine the voltage noise generation in the FPGA-based simulation.

B. Simulation

The simulator keeps a time-step counter. At the start of a transient simulation, the time-step counter is reset to 0, and the CUT delay model picks a delay t_{path} (in units of T_{step}), indicating that the input is launched with an expected propagation delay t_{path}. The output data are initialized to be invalid and remains invalid until the time-step counter reaches t_{path}. The transient simulation proceeds in steps of T_{step}, and the time-step counter increments by 1 in every step. In each simulation step, each error model decides whether to generate an error event based on a selected error rate or not. If a SET event is generated, the output is invalidated for the duration of the SET event. If a voltage droop is generated, t_{path} is lengthened or shortened according to a selected sinusoidal function. The simulation controller keeps track of the time-step counter, the data path delay, and the error states, and it makes updates to the output indicator. When the time-step counter reaches clock period T_{clk}, the controller inspects the output indicator and records an error if the output is invalid. The simulation then moves to the next clock period. The time-step counter resets to 0, a new path delay is picked, and the process continues.

The simulation can be used to evaluate a pipeline-level EDAC design. An EDAC technique specifies an error checking window CW (the duration in terms of the number of T_{step} and the position relative to the clock cycle boundary), along with an error correction mechanism. To simulate EDAC, in each T_{step} inside CW, the simulation controller inspects the output, and an error is flagged if the output is changed during the checking window. The detection of an error triggers error correction, e.g., by stalling the pipeline (moving the current cycle boundary to $2T_{\mathrm{clk}}$) or by reissuing the instruction (purging the pipeline and reissuing the current t_{path}). The simulator uses a performance counter to keep track of the number of outputs produced and an error counter to track the number of erroneous outputs. Together with the time-step counter, the simulator measures the performance and the error rate. An overview of the simulator is shown in Fig. 4.

III. Evaluation of EDAC Designs

The complete transient error simulator is implemented on a Berkeley Emulation Engine Version 3 (BEE3) platform [11] for a multistage pipeline. Each pipeline stage is modeled using a separate CUT delay profile. The SET, coupling noise, and voltage droop profiles are added in the simulation. We perform experiments on representative data paths and error profiles to evaluate common EDAC techniques.

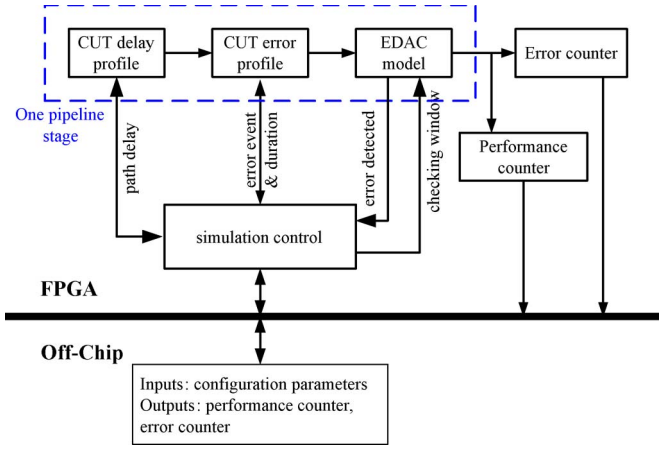

Fig. 4. FPGA-based transient simulation platform. A multistage simulation can be constructed by cascading single stages with individual delay profiles and models.

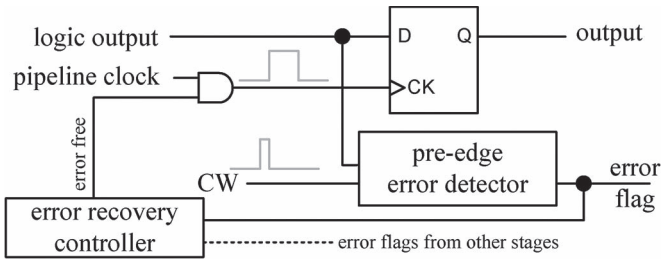

Fig. 5. Preedge error detection and recovery [3].

A. Preedge EDAC

The preedge EDAC, as shown in Fig. 5, has been proposed in [2] and [3] to monitor errors by detecting glitches in the checking window before the output is registered. The preedge EDAC is effective in detecting slow changing negative-bias-temperature-instability-induced pMOS aging and random transient faults. A longer checking window provides better protection against errors at the cost of lengthening the clock period and degrading the performance. The preedge error detection is accompanied by a pipeline stall to correct errors.

We evaluate the preedge EDAC on a CORDIC processor that is synthesized in the 45-nm CMOS technology. The processor consists of three identical pipeline stages, and each pipeline stage is considered a CUT. The CUT delay profile is shown in Fig. 3. We adjust the SET rate and duration, and we measure the effect on the CORDIC processor while tuning the length of CW. The FPGA-based transient simulator captures the CORDIC processor's error rate due to the SET, as shown in Fig. 6. The CORDIC processor's reliability decreases with a higher SET rate and a longer SET duration (labeled as TD in the figures). One important observation uncovered by the transient simulation is that the preedge EDAC is only effective when CW is comparable with or longer than the SET duration. Two orders of magnitude of reliability improvement over the unprotected case can be achieved when CW is appropriately chosen.

The preedge EDAC triggers pipeline stalls to correct errors, resulting in throughput degradation. Transient simulation shows that the throughput of the CORDIC processor is primarily determined by the SET rate and the length of CW, as shown

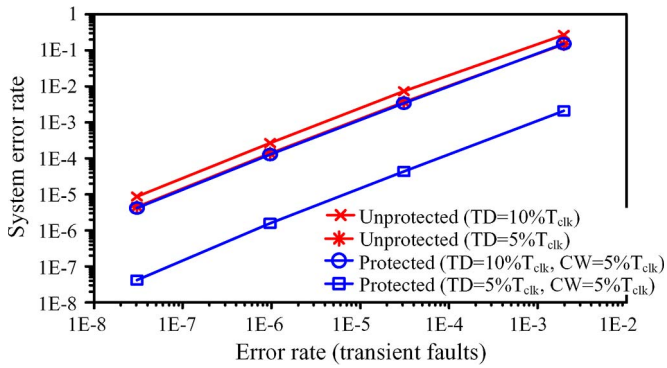

Fig. 6. Reliability improvement with the preedge EDAC (CW: checking window; TD: transient error duration; $T_{\rm clk}$: clock period).

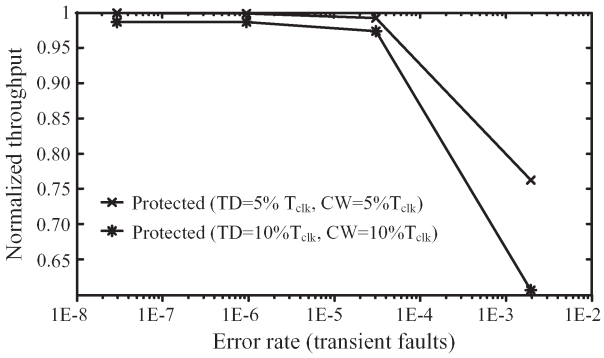

Fig. 7. Effective throughput using the preedge EDAC (CW: checking window; TD: transient error duration; $T_{\rm clk}$: clock period).

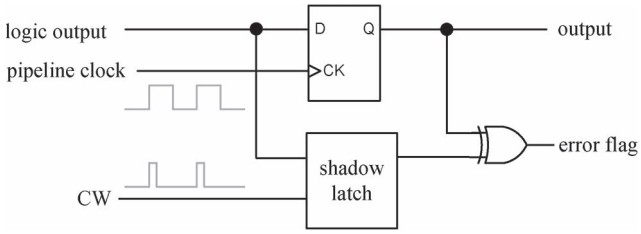

Fig. 8. Postedge error detection [4].

in Fig. 7. Lengthening CW enhances the error protection but also increases the chance of detecting data transition in the critical paths and therefore degrades the throughput. The tradeoff between the reliability and the performance obtained from the transient simulation can be used to guide practical designs.

B. Postedge EDAC

The postedge EDAC has been very popular in high-performance low-power designs. The postedge EDAC, as illustrated in Fig. 8, detects errors after the clock edge, allowing the correction of delay errors from long paths that exceed the clock cycle time [12]. The technique is often applied in conjunction with dynamic voltage and frequency scaling to increase the clock frequency for higher performance or to reduce the supply voltage for lower power consumption.

We evaluate the postedge EDAC on an Alpha processor [13] that is synthesized in the 45-nm CMOS technology. Application

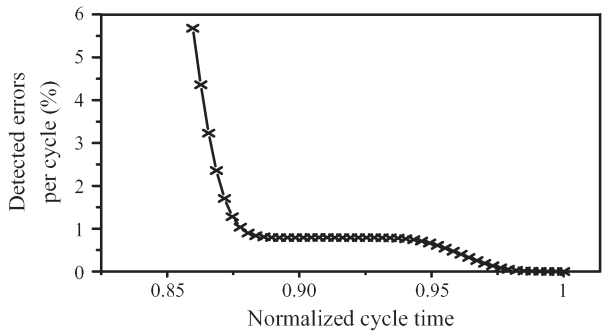

Fig. 9. Postedge error detection rate.

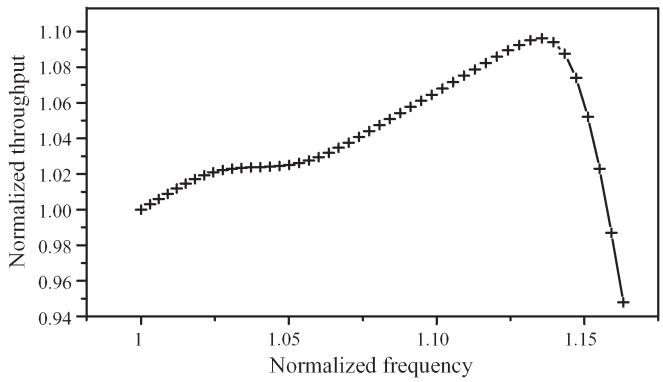

Fig. 10. Effective throughput using the postedge EDAC.

traces were obtained from the execution stage of the Alpha processor running the instructions of a recursive Fibonacci number generation. We consider the effects of the coupling noise and the voltage droop on the Alpha processor. The coupling noise is assumed to be randomly occurring, and its duration is modeled with a normal distribution of a standard deviation of 1%–3% of $T_{\rm clk}$. The voltage droop usually lasts for a longer period, and we assume that it increases the CUT delay by up to 15% of $T_{\rm clk}$ [10].

The Alpha processor is simulated with the postedge EDAC. A fixed postedge CW is selected based on the fastest path delay to avoid hold time issues. Dynamic frequency scaling is applied to take advantage of the postedge EDAC. As the clock period decreases with an increasing frequency, more errors are detected, as shown in Fig. 9. Errors can be corrected using instruction flush [4] or system nuke [5] that are aided by the microarchitecture and the operating system. However, error correction usually introduces a few cycles of penalty, and it decreases the throughput. Considering an average five cycles of penalty to flush the Alpha processor's pipeline and reissue the instruction, the effective throughput using dynamic frequency scaling can be measured. As shown in Fig. 10, the peak throughput is achieved at 1.13 times the nominal frequency. Such a design exploration involving the reliability and the performance can be quickly obtained using the FPGA-based simulator.

C. FPGA Resource Usage and Performance

The FPGA resource utilization based on a Xilinx Virtex-5 XC5VLX155T device in a BEE3 platform [11] is listed in

TABLE I
DEVICE UTILIZATION OF THE FPGA-BASED SIMULATOR (BASED ON XILINX VIRTEX-5 XC5VLX155T) AND COMPARISON WITH THE SIMULATION ON AN INTEL CENTRAL PROCESSING UNIT

	Pre-Edge EDAC	Post-Edge EDAC
Slice registers	3,004 (3%)	2,581 (2%)
Slice LUTs	2,750 (2%)	2,865 (2%)
BRAMs	96 (45%)	80 (37%)
Simulation throughput (FPGA at 100MHz)	100 kS/s	100-120 kS/s
Simulation throughput (Intel i7 at 2.4GHz)	16.4 kS/s	13.5-16.2 kS/s

Table I for the designs of three pipelined stages, with simulation time step $T_{\rm step}$ set to 1/1000 of a pipeline clock period. The slice register and the lookup table (LUT) utilization are very low. The BEE3 [11] base system uses block random access memory devices (BRAMs), which makes the BRAM utilization appear higher. At a 100-MHz FPGA clock frequency and a simulation time step $T_{\rm step}$ set to 1/1000 of a pipeline clock period, the preedge EDAC simulation can be done at a throughput of 100 kilosamples per second (kS/s), and the postedge EDAC simulation can be done at a throughput ranging from 100 to 120 kS/s, depending on the error rate. The closest comparison of this brief is to perform a similar transient simulation in software. The advantage of using the FPGA is the high parallelism that allows large-scale circuits and EDAC to be simulated at 100 kS/s or above at a resolution of 1000 steps per simulated clock cycle. As shown in Table I, the simulation throughput of a three-stage CORDIC with the preedge EDAC is 100 kS/s on the 100-MHz FPGA-based simulator compared with 16.4 kS/s on an Intel 2.4-GHz Core i7 processor. The simulation throughput of an Alpha processor with the postedge EDAC is 100–120 kS/s on the FPGA-based simulator, as compared with 13.5–16.2 kS/s on the Intel 2.4-GHz Core i7 processor. The improvement ranges from a factor of 6 to a factor of 8. This experimental evidence has demonstrated the possible speedup with the proposed FPGA platform. Note that these two examples are relatively small. The improvement is expected to be even greater for more complicated data paths as more parallel designs are possible on the FPGA and if the FPGA design could be synthesized to achieve a higher clock frequency.

IV. CONCLUSION

We proposed an FPGA-based simulator for the EDAC design and evaluation. The general-purpose simulator consists of configurable CUT delay profile and error profile to be widely applicable. EDAC techniques, including preedge EDAC and postedge EDAC, are simulated on this platform based on a synthesized CORDIC processor and an Alpha processor. The simulations shed light on key design choices, such as the length of the preedge checking window and its impact on reliability and performance. The FPGA-based simulation complements circuit and gate-level simulation and system emulation as a useful tool for evaluating EDAC designs.

REFERENCES

[1] S. Borkar, "Design perspectives on 22nm CMOS and beyond", in *Proc. Annu. Design Autom. Conf.*, 2009, pp. 93–94.
[2] M. Zhang *et al.*, "Sequential element design with built-in soft error resilience," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 14, no. 12, pp. 1368–1378, Dec. 2006.
[3] M. Agarwal, B. C. Paul, M. Zhang, and S. Mitra, "Circuit failure prediction and its application to transistor aging," in *Proc. VLSI Test Symp.*, 2007, pp. 277–286.
[4] S. Das *et al.*, "RazorII: In situ error detection and correction for PVT and SER tolerance," *IEEE J. Solid-State Circuits*, vol. 44, no. 1, pp. 32–48, Jan. 2009.
[5] K. A. Bowman *et. al.*, "A 45 nm resilient microprocessor core for dynamic variation tolerance," *IEEE. J. Solid-State Circuits*, vol. 46, no. 1, pp. 194–208, Jan. 2011.
[6] C. Lopez-Ongil, M. Garcia-Valderas, M. Portela-Garcia, and L. Entrena, "Autonomous fault emulation: A new FPGA-based acceleration system for hardness evaluation," *IEEE Trans. Nucl. Sci.*, vol. 54, no. 1, pp. 252–261, Feb. 2007.
[7] M. Aguirre, V. Baena, J. Tombs, and M. Violante, "A new approach to estimate the effect of single event transients in complex circuits," *IEEE Trans. Nucl. Sci.*, vol. 54, no. 4, pp. 1018–1024, Aug. 2007.
[8] M. G. Valderas, L. Entrena, R. F. Cardenal, C. L. Ongil, and M. P. Garcia, "Set emulation under a quantized delay model," *J. Electron. Testing*, vol. 25, no. 1, pp. 107–116, Feb. 2009.
[9] L. Entrena *et al.*, "Soft error sensitivity evaluation of microprocessors by multilevel emulation-based fault injection," *IEEE Trans. Comput.*, vol. 61, no. 3, pp. 313–322, Mar. 2012.
[10] K. A. Bowman, C. Tokunaga, T. Karnik, V. K. De, and J. W. Tschanz, "A 22 nm all-digital dynamically adaptive clock distribution for supply voltage droop tolerance," *IEEE J. Solid-State Circuits*, vol. 48, no. 4, pp. 907–916, Apr. 2013.
[11] J. D. Davis, C. P. Thacker, C. Chang, C. Thacker, and J. Davis, "BEE3: Revitalizing computer architecture research," Microsoft Research, Mountain View, CA, USA, Tech. Rep. MSR-TR-2009-45, 2009.
[12] K. A. Bowman and J. W. Tschanz, "Resilient microprocessor design for improving performance and energy efficiency," in *IEEE/ACM Int. Conf. Comput.-Aided Des.*, 2010, pp. 85–88.
[13] R. L. Sites, *Alpha Architecture Reference Manual*. Bedford, MA, USA: Digital, 1998.