AN ELECTRONIC DESIGN AUTOMATION TOOL FOR EFFICIENTLY IMPROVING THE RELIABILITY OF NANO-CIRCUITS

Walid Ibrahim and Hoda Amer
College of Information Technology
UAE University
PO box 15551, Al Ain, UAE
E-mail: walidibr@uaeu.ac.ae, hamer@uaeu.ac.ae

KEYWORDS

CMOS, logic gates, reliability, simulation, redundancy, heuristic, EDA tools.

ABSTRACT

Scaling the CMOS transistors have been used consistently over the last five decades by the semiconductor industry to develop smaller, faster, and cheaper electronic devices. However, the massive scaling of CMOS devices deep into the deca-nanometer range has significantly reduced their reliability margins, and increased their vulnerability to both transient and permanent faults. Hence, incorporating some form of redundancy in the design of logic circuits is becoming a necessity for ensuring reliable operation. Nevertheless, incorporating redundancy for improving reliability is always a trade-off for increased area and more complex connectivity, which normally lead to higher delay and power consumption. Therefore, there is a need for electronic design automation tools for optimizing the design these conflicting goals. This paper introduces a highly efficient and accurate algorithm to calculate the circuit's reliability based on the vulnerability of the circuit's output signals to the failure of the individual gates. Simulation results show that, due to the error masking ability of the logic gates, some gates could have much higher impact on the circuit's reliability than the others. Improving the reliability of these gates would improve the circuit's reliability effectively, while having minimum impact on the design area, delay , and power consumption.

INTRODUCTION

Over the last few decades, the continuous scaling of the highly reliable CMOS devices was the basis for the exceptional growth of the semiconductor industry. Reducing the feature size of the CMOS transistors has always been used to implement faster, smaller, cheaper, and more power efficient commercial off-the-shelf consumer electronics. Due to the high reliability of the CMOS devices, the main focus of Electronic Design Automation (EDA) tools was the optimization of the delay, power, and area aspects of the design. Conversely, little attention was given to design reliability, except in the case of military, health, space, and other critical mission applications (Baumann 2005)

However, with the massive scaling of CMOS devices deep into the deca nanometer range, their electrical features (voltages and capacitance) have been reduced significantly, and consequently, their sensitivity to noise and variations significantly increased. A major source of variation is the randomness of the exact locations of doping atoms (McPherson 2001). Although the doping concentration is controlled by ion implantation and annealing processes, it was shown in (Borkar 2005) that the standard deviation of the number of doping atoms per device increases as the doping area shrinks. Hence, leads to device-to-device variations in key parameters, which would cause inaccurate transistor switching and affect the functionality of logic gates, flip-flops, and memory bit-cells.

Thus, having a manufacturing process that produces highly reliable electronics (99.999%) with future nano-devices could be extremely challenging, if not plainly impossible. As a result, future nano-circuits will have to rely on less than perfect devices and compensate for the anticipated transient and permanent failures by incorporating redundancy at the architectural, gate and device levels (Hamamstsu et al. 2010). The well-known approach for developing fault-tolerant architectures is to incorporate space, time, or information redundancy. Space (hardware) redundancy relies on voters and includes among others the well-known: modular redundancy (Wakerly 1976), cascaded modular redundancy (Abaham and Siewiorek 1974, and Lee et; al 2007), and multiplexing including von Neumann multiplexing (von Neumann 1956), enhanced von Neumann multiplexing (Roy et al.), and parallel restitution (Sadek et al. 2004). Time redundancy is trading space for time, such as alternating logic, re-computing with shifted operands, and re-computing with swapped operands, while information redundancy is based on error detection and error correction code (Mitra et al. 2006). All of the above approaches share the concept that improved reliability is traded off for increased area and more complex connectivity, which normally lead to higher power consumptions, and/or slower computations.

Therefore, Reliability Electronic Design Automation (EDA) tools are becoming essential. Accurate and efficient EDA tools would allow circuit designers to

Communications of the ECMS, Volume 33, Issue 1,
Proceedings, ©ECMS Mauro Iacono, Francesco Palmieri,
Marco Gribaudo, Massimo Ficco (Editors)
ISBN: 978-3-937436-65-4/978-3-937436-66-1(CD) ISSN 2522-2414

evaluate the effect of different redundancy schemes and select the one that optimizes trade-offs between the conflicting goals of minimizing area-power-delay and improving reliability. Several tools and analytical algorithms have been proposed to calculate the reliability of logic circuits, which will be discussed in Section II. Some of these tools are approximate while others calculate the circuit reliability accurately. Some scale linearly with circuit size and others have computational difficulties and scale exponentially. However, none of these tools studied the effect of individual gates on the system overall reliability. This is despite the fact that studying the effect of individual gates is essential for efficiently improving the circuit reliability. Instead of increasing the reliability of each gate in the circuit, the circuit designer may utilize an iterative process to improve the circuit overall reliability. In each iteration, appropriate redundancy architectures (at the gate, or the device level) are applied to improve the reliability of the most critical gates, until the target reliability margin is achieved.

In this paper, the gate criticality score with respect to the circuit reliability is defined. A heuristic algorithm is also introduced to accurately and efficiently calculate and rank the gates' criticality. The rest of the paper is organized as follows. The state of the art is introduced in Section 2 followed by definition of the criticality score in Section 3. The proposed algorithm for criticality score calculation is detailed in Section 4, followed by experimental results and conclusion remarks in section 5 and 6.

STATE OF THE ART

Several tools and analytical algorithms have been proposed to calculate the reliability of logic circuits, such as the probabilistic transfer matrices (PTM) (Patel et al. 2003, and Xiao et al. 2016). The PTM method evaluates the circuit's overall reliability by combining the PTMs of elementary gates or sub-circuits. It then performs simultaneous computation over all possible input combinations, and calculates the exact reliability for each input vector. Another advantage of the PTM (beside accuracy) is its ability to assign different reliability values to different gates. However, the PTM has a major memory bottleneck. The required memory grows exponentially with the number of input and output signals. For a circuit with n inputs and m outputs, the straightforward PTM representation requires $O(2^{n+m})$ memory space. Krishnaswamy et al. 2005, used algebraic decision diagrams compression method to reduce the memory requirements of the PTM method, and was able to evaluate circuits with about 40 input/output signals.

The probabilistic gate model (PGM) (Han et al. 2005) is another method that can be used for any type of gates and fault models. The overall reliability for a circuit is obtained by multiplying the individual reliabilities for each output. The PGM is fast and easy to implement. However, it assumes that I/O signals are uncorrelated, which significantly affects its accuracy due to the re-convergence of fan-out signals.

Bayesian Networks (BNs) is another example for exact reliability calculation tool) Ibrahim et al. 2012). A BN is a probabilistic model that represents a set of variables and their probabilistic dependencies. Using exact inference allows the BN based tools to calculate the circuit reliability accurately. Experiment results have shown that BN exact inference algorithms could be used to calculate the circuit's reliability for only tiny circuits with up to 1000 gates) Ibrahim e al. 2012). Calculating the exact reliability for larger circuits fails as the network model becomes too complex for exact inference algorithms to handle.

Other recent reliability algorithms include the Boolean difference-based error calculator (BDEC) method (Mohyuddin et al. 2008). The observability-based approach to reliability analysis is described in (Choudhury et al. 2009). Chen et al. proposed a stochastic model for accurate reliability evaluation of logic circuits (Chen et al. 2010). Another method to analyze the reliability of logic circuits was proposed by Hamiyati et al. (2016). The proposed method converts the problem of calculating the circuit reliability into a problem of finding the solution of a system of nonlinear equations obtained using Mason's rule.

All the above algorithms can be used to calculate or estimate the reliability of the circuit's output signals. However, none of these algorithms studied the effect of individual gates on the reliability of each output signal. For instance, the simulation results presented later show that in the case of circuit c1355 of the ISCAS'85 benchmark, all the 32 output signals are vulnerable to the failure of the same 91 gates, while 48 out of the c3540 50 output signals are vulnerable to two gates only. Therefore, identifying the critical gates and their relationship to the circuit's output signals and input vectors is crucial to effectively improving the reliability of the logic circuit.

CRITICALITY SCORE

A gate failure is defined in this paper as the inability of the gate to produce the expected output logic with respect to the logic presented at its inputs. Although logic gates could be affected by several fault types (e.g., stuck at, and stuck open, and von Neumann), not every logic error caused by a gate failure would affect the circuit behavior. Logic gates such as AND, NAND, OR, and NOR are capable of masking out the logic errors caused by the failure of predecessor logic gates and prevent them from propagating towards the circuit's output ports. For instance, in case of the AND gates with n fanins, if one out of the n input signals is received correctly as logic 0, the gate's output signal will be 0 regardless of the logic values received on the other $n - 1$ inputs. This means that the gate will be able to mask out the logic errors received on any of the $n - 1$ inputs.

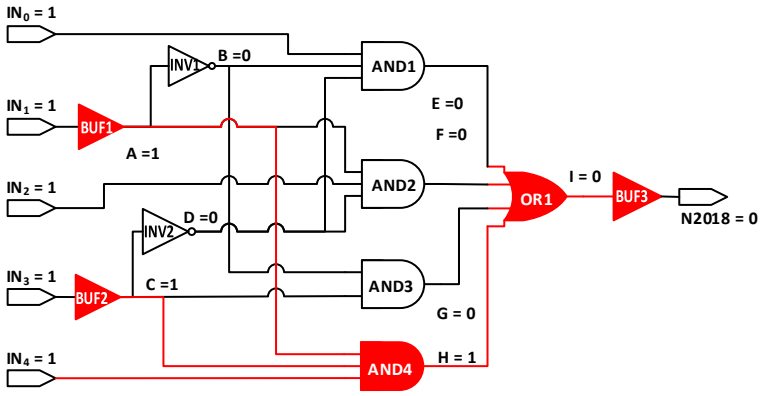

Figure 1: Critical gates and logic signals for ISCAS'85 circuit 2670, output N2018 when input vectors '11111' is applied.

Due to the error masking property, a subset of the logic gates and logic signals are more critical than the others with respect to their effect on the circuit reliability. *A critical gate* is defined as the gate that has a direct impact on the circuit's behavior. The logic error caused by the failure of a critical gate is not masked out by any subsequent gate, and directly affects the circuit's output signal. Similarly, *a critical signal* is defined as the input signal that has a direct impact on the gate's output logic. The failure (flip) of a critical input signal will cause the gate's output logic to fail as well. For example, when '11' is applied to the inputs of an AND gate, both input signals are considered critical since flipping either of them to 0 will flip the gate's output to 0 instead of 1.

Given a logic circuit with number of gates (n_{gates}), input ports (*ins*), and output ports (*outs*). There is 2^{ins} different input vectors that can be applied at the circuit's input ports. Each input vector could have a different effect on the circuit reliability, as it affects the ability of logic gates to mask out logic errors and prevents them from propagating toward the output ports. Therefore, a gate could be critical for a specific output port only when specific input vectors are applied and uncritical for the others. For a given input vector '*i*', the criticality count ($CC_{g,i}$) of gate *g* is defined as the number of output signals that are directly affected by the failure of gate '*g*'. Hence, the criticality score of gate g (CS_g) is calculated as:

$$CS_g = \sum_i^m \sum_j^v CI_{i,j}/m \times v \qquad (1)$$

Where *m* is the number of output signals, *v* is the number of applied input vectors, and *CI* is the criticality indicator. *CI* is one if the gate g is critical to output *i* when input vector *j* is applied.

The criticality status of gates and logic signals depends on several factors, including the types of the logic gates, the interconnections between them, and the applied input vector at the circuit's primary inputs. Figure 1 shows the critical signals (red lines) and gates (red gates) when input 11111 is applied to the input ports for the logic cone N2018 of ISCAS'85 circuit 2670. It shows that the output gate (BUF3) is critical, as its failure will directly affect the circuit's output signal. Likewise, OR1 is also critical, as BUF3 will not be able to mask out the logic error that might occur due to the failure of OR1. The inputs to gate OR1 are signals E = 0, F = 0, G = 0, and H = 1. Hence, only signal H is critical, as its failure will switch the output logic of OR1 from 1 to 0. Therefore, gate AND4 is also considered critical as its failure will affect signal H, and consequently the circuit's output signal. Similarly,

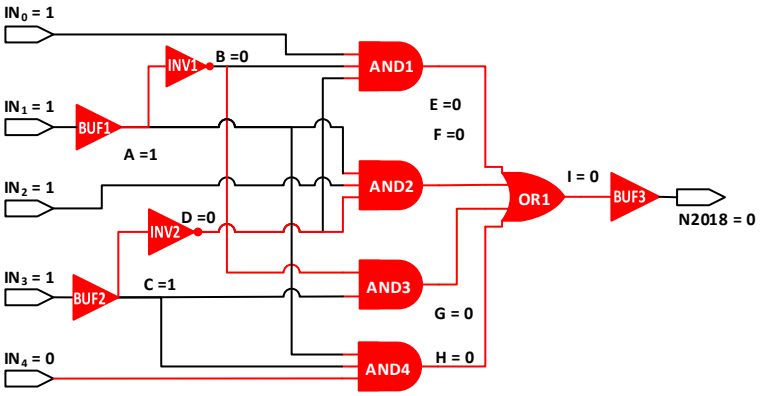

Figure 2: Critical gates and logic signals for ISCAS'85 circuit 2670, output N2018 when input vectors 01111 is applied.

BUF1, and BUF2 are also considered critical as their failure directly affect the inputs to AND4.

Figure 2 shows that effect of changing IN4 from 1 to 0. It shows that all gates became critical. This means that the failure of any gate will cause the output signal to flip. It is noteworthy that the same signal could be critical for one gate and uncritical for another (e.g., signals B, and C).

CALCULATING THE CRITICALITY SCORE

A straightforward approach to check the criticality status of a gate '*g'*, is to apply an input vector at the circuit input ports, and calculate the expected error free logic at each output port. A logic error is then injected at the output of the gate '*g*', and the logic at the circuit's output ports are then recalculated and compared with the error free logic. If the logic of an output port changed, gate '*g'* is considered critical to that output port. This approach is simple and easy to implement. However, it does not scale well as it has a complexity of $O(n \log(n))$.

Figures 1, and 2 shows that the output gate (gate connected to the output port) is always critical as its failure directly affects the output signal. It also shows that if the critical gate has a critical input signal, the gate connected to that signal is also critical (e.g., gate AND4 connected to signal H). Therefore, instead of injecting an error at each gate to verify its criticality status, the proposed algorithm use a recursive depth first algorithm to identify the critical nodes for a given input vector.

The algorithm starts by extracting the circuit's topology information from the circuit's netlist file. Using the extracted information, an acyclic graph is created. The nodes in the created graph represent the gates in the circuit, while the edges represent the connecting wires. The output edge of a node is the input edge for its direct successor(s). From here on, node and gate can be used interchangeably, as well as edge and wire.

The algorithm arranges the nodes into tiers such that all the input nodes are assigned to the first tier, and each node is assigned to a tier higher than the tier(s) assigned to its direct predecessors. The node list is then sorted in ascending order based on its logic depth (tier value). This ensures proper logic propagation by determining the output logic of a node before any of its successors are considered. After sorting the nodes, the input vector is applied to the circuit input edges, and the expected output logic at each gate is calculated. The criticality count of each node is set to 0, and the criticality status of the edges is also calculated.

The algorithm then iterates through the circuit output nodes. In each iteration, it calls a depth first recursive function with the output node as the input parameter. The recursive function takes a critical node as an input parameter (the output node is critical by definition). It increments the node's criticality count, marks the node as visited to prevent the recursive function from revisiting the same fanout node several times. The recursive function then checks the input edges of the node. If an edge is critical, the function checks if the predecessor node connected to that edge is not visited yet. If not, the recursive function recalls itself with the predecessor node as the new input parameter. The recursive function returns if the current node does not have any input edge that is critical and connected to a node that is not visited yet. After executing the algorithm for *m* input vectors, the gate's criticality score is then calculated using equation (1). The algorithm is highly scalable as the complexity per input vector is $O(\log n)$.

SIMULATION RESULTS

Simulation experiments were conducted to characterize the performance and the accuracy of the proposed algorithm using 11 circuits extracted from the ISCAS'85 benchmark. Only circuit c17 was excluded from the benchmark, as it has only six gates and two outputs. Table 1 shows the number of gates, inputs, and outputs per circuit.

Table 2 shows the 10 most critical gates for each circuit when 100000 random input vectors are applied. It shows that gate OR_145 is the most critical gate for circuit c3540 with a score of 58.6%. This means that, for the 100000 applied input vectors, on average, 58.6% of c3540's 22 output ports were vulnerable to the failure of OR2_145. Similarly, in case of circuit c432, AND_46 is the most critical gate with an average score of 42.9%, followed by AND_208 with an average criticality score of 38.6%.

Table 3 shows the scores of the 10 least significant gates. It shows that some gates could have 0 criticality score. This means that the logic errors caused by the failure of these gates are masked off by a predecessor gate and does not affect the circuit's outputs. Therefore, using redundancy schemes for improving the reliability of these gates will affect the power, delay and area of the design with minimal or no effect on reliability, as they have no effect on the output signals.
Using the criticality score information generated by the proposed algorithm, the circuit designer will able to

Table 1: ISCAS85 Benchmark Circuits Statistics.

Name	Inputs	Outputs	Gates
c432	32	7	123
c499	41	32	202
c880a	60	26	383
c1355	41	32	546
c1908	33	25	880
c2670	233	140	1269
c3540	50	22	1669
c5315	178	123	2307
c6288	32	32	2416
c7552	207	108	3513

Table 2: Score of the 10 Most Critical Gates

c7552	c6288	c5315	c3540	c2670	c1908	c1355	c880a	c499	c432
BUFF_67 **29.7%**	NOR_334 **18%**	BUFF_94 **16.3%**	OR_145 **58.6%**	BUFF_28 **7.8%**	NOT_15 **19.6%**	AND_4 **6.2%**	AND_4 **6.7%**	AND_131 **6.2%**	AND_46 **42.9%**
NOT_141 **20.3%**	AND_18 **18%**	BUFF_68 **15.2%**	AND_208 **56%**	BUFF_29 **7.8%**	NOT_10 **19.6%**	AND_5 **6.2%**	NAND_128 **6.1%**	AND_132 **6.2%**	AND_86 **38.6%**
NAND_54 **12.8%**	AND_17 **17.9%**	BUFF_95 **14.5%**	AND_84 **55.4%**	BUFF_36 **6%**	NOT_16 **19.5%**	AND_6 **6.2%**	NAND_96 **6%**	AND_133 **6.2%**	AND_126 **31.7%**
BUFF_305 **12.3%**	NOT_257 **17.9%**	NOT_385 **13.2%**	AND_111 **52.2%**	BUFF_35 **6%**	NOT_6 **13.6%**	AND_7 **6.2%**	NAND_116 **5.9%**	AND_134 **6.2%**	NAND_139 **28.5%**
BUFF_301 **12.2%**	NOT_275 **17.8%**	BUFF_139 **12.8%**	NOT_368 **51.4%**	NOT_113 **5.6%**	NOT_8 **13.6%**	AND_0 **6.2%**	NAND_5 **5.2%**	AND_135 **6.2%**	NAND_140 **22.7%**
NOT_91 **11.2%**	NOR_303 **17.8%**	BUFF_140 **12.8%**	OR_571 **50.9%**	NOT_112 **5.6%**	NOT_9 **13.5%**	AND_1 **6.2%**	NOT_59 **5.2%**	AND_136 **6.2%**	NAND_117 **22.6%**
OR_718 **11.1%**	AND_97 **17.7%**	BUFF_137 **12.8%**	NAND_58 **50.2%**	NOT_121 **4.5%**	NOT_7 **13.5%**	AND_2 **6.2%**	NAND_204 **4.1%**	AND_137 **6.2%**	NAND_68 **22.5%**
BUFF_303 **10.7%**	NOT_262 **17.7%**	BUFF_138 **12.8%**	NOT_33 **48.9%**	NOT_122 **4.3%**	NOT_3 **13.5%**	AND_3 **6.2%**	OR_19 **3.9%**	AND_138 **6.2%**	NAND_122 **22.4%**
NOT_442 **10.7%**	AND_81 **17.7%**	BUFF_69 **12.5%**	OR_568 **48.8%**	BUFF_40 **3.6%**	NOT_12 **13.5%**	AND_8 **3.1%**	AND_18 **3.8%**	AND_139 **3.1%**	NOT_47 **22.3%**
BUFF_70 **10.5%**	NOT_261 **17.7%**	OR_777 **12.3%**	AND_148 **47.1%**	BUFF_39 **3.6%**	NOT_5 **13.5%**	AND_9 **3.1%**	AND_352 **3.8%**	AND_140 **3.1%**	NAND_74 **22.2%**

identify the gates that have most effect on the circuit reliability. Improving the reliability of these gates will significantly improving the circuit reliability, while having a minimum effect on the design area, power, and delay parameters.

CONCLUSIONS

Using different redundancy architectures to protect future nano-circuits against the expected high level of permeant and transient errors is crucial. This paper introduced an algorithm to help circuit designers identify the most critical nodes in the circuit with respect to the circuit's outputs. Simulation results show that some gates are more critical to a circuit's reliability as they affect more output signals. Improving the reliability of these gates would effectively improve the circuit's reliability while having a minimum impact on the other design parameters. The algorithm could also assign different priorities for different output signals and calculate the criticality score accordingly, as gates that affect the higher priority outputs get higher criticality score.

REFERENCES

Abraham, J. A., and Siewiorek, D. P. 1974 "An algorithm for the accurate reliability evaluation of triple modular redundancy networks," *IEEE Transactions on Computers*, vol. 23, Jul. 1974, pp. 682–692.

Baumann, RC. 2005. "Radiation-induced soft errors in advanced semiconductor technologies," IEEE Transactions on Devices and Materials Reliability, vol. 5. Sep. 2005, pp. 305-316.

Borkar, S. 2005. "Designing reliable systems from unreliable components: The challenges of transistor variability and degradation." IEEE Micro, vol. 25, Nov.-Dec. 2005, pp. 10–16.

Chen H., and Han, J. 2010. "Stochastic computational models for accurate reliability evaluation of logic circuits." In Proceedings of IEEE/ACM Great Lakes Symposium on VLSI, Providence, RI, USA, May 2010, pp. 61–66.

Choudhury, M. R., and Mohanram, K. 2009. "Reliability Analysis of Logic Circuits." IEEE Transactions on Computer Aided Design, vol. 28, Feb. 2009, pp. 392–405.

Hamamatsu, M., Tsuchiya, T., and Kikuno, T. 2010. "On the reliability of cascaded TMR systems." In Proceedings of IEEE 16th Pacific Rim International Symposium on Dependable Computing, Tokyo, Japan, Dec. 2010, pp. 184–190.

Hamiyati V., Peiravi, A. 2014. "A graph based approach for reliability analysis of nano-scale VLSI logic circuits. 2014." Microelectronics Reliability, vol. 54, 2014, pp. 1299–1306.

Han, J., Taylor, E.R., Gao, J.B., and Fortes, J.A.B. 2005. "Faults, error bounds and reliability of nanoelectronic circuits." In Proceedings of 2005 IEEE International Conference on Application-Specific Systems, Architecture Processors (ASAP'05), Samos, Greece, Jul. 2005, pp. 247–253.

Ibrahim, W., and Beiu, V. 2011. "Using Bayesian networks to accurately calculate the reliability of complementary metal oxide semiconductor gates." IEEE Transactions on Reliability, vol. 60, Sep. 2011, pp. 538–549.

Table 3: Score of the 10 Most Critical Gates

c7552	c6288	c5315	c3540	c2670	c1908	c1355	c880a	c499	c432
0.0000	0.0187	0.0001	0.0001	0.0001	0.0001	0.0010	0.0024	0.0010	0.0225
0.0000	0.0181	0.0000	0.0000	0.0000	0.0000	0.0010	0.0024	0.0010	0.0219
0.0000	0.0180	0.0000	0.0000	0.0000	0.0000	0.0010	0.0023	0.0010	0.0205
0.0000	0.0173	0.0000	0.0000	0.0000	0.0000	0.0010	0.0023	0.0010	0.0198
0.0000	0.0169	0.0000	0.0000	0.0000	0.0000	0.0010	0.0020	0.0010	0.0128
0.0000	0.0167	0.0000	0.0000	0.0000	0.0000	0.0010	0.0020	0.0010	0.0123
0.0000	0.0157	0.0000	0.0000	0.0000	0.0000	0.0010	0.0015	0.0009	0.0119
0.0000	0.0147	0.0000	0.0000	0.0000	0.0000	0.0009	0.0013	0.0009	0.0089
0.0000	0.0136	0.0000	0.0000	0.0000	0.0000	0.0009	0.0009	0.0009	0.0079
0.0000	0.0112	0.0000	0.0000	0.0000	0.0000	0.0009	0.0008	0.0009	0.0075

Krishnaswamy, S., Viamontes, G.F., Markov, I.L., and Hayes, J.P. 2005. "Accurate reliability evaluation and enhancements via probabilistic transfer matrices," In Proceedings of Design Automation and Test Europe Conference, DATE'05, Munich, Germany, Mar. 2005, pp. 282–287.

Lee, S., Jung, J., and Lee, I. "Voting structures for cascaded triple modular redundant modules," *IEICE Electronics Express*, vol. 4, Nov. 2007, pp. 657–664.

McPherson, J.W. 2001. "Scaling-Induced Reductions in CMOS Reliability Margins and the Escalating Need for Increased Design-In Reliability Efforts," In Proceedings of the IEEE 2nd International Symposium on Quality Electronic Design, San Jose, CA, USA, Mar. 2001, pp. 123–30.

Mitra, S., Zhang, M., Seifert, N., Mak, T.M., and Kim, K.S. 2006. "Soft error resilient system design through error correction." In Proceedings of the 2006 IFIP International Conference on Very Large Scale Integration, Nice, France, Oct. 2006.

Mohyuddin, N., Pakbaznia, E., Pedram, M. 2008. "Probabilistic error propagation in logic circuits using the Boolean difference calculus." In Proceedings of IEEE Computer Design, Lake Tahoe, CA, USA, Oct. 2008. pp. 7–13.

Patel, K. N., Markov, I. L., and Hayes, J. P. 2003, "Evaluating circuit reliability under probabilistic gate-level fault models," In Proceedings of the International Workshop on Logic Synthesis, Laguna Beach, CA, USA, May 2003, pp. 59–64.

Roy, S., and Beiu, V. 2004. "Multiplexing schemes for cost-effective fault-tolerance," In Proceedings of IEEE Conference on Nanotechnology. IEEE-NANO'04, Munich, Germany, Aug. 2004, pp. 589–592.

Roy, S., and Beiu, V. 2005. "Majority multiplexing—Economical Redundant Fault-Tolerant Designs for Nanoarchitectures." IEEE Transactions on Nanotechnology, vol. 4, no. 4, Jul. 2005, pp. 441–451.

Sadek, A.S. , Nikolić, K., and Forshaw, M. 2004, "Parallel information and computation with restitution for noise-tolerant nanoscale logic networks." Nanotechnology, vol. 15, Jan. 2004, pp. 192–210.

von Neumann, J. 1956. "Probabilistic logics and the synthesis of reliable organisms from unreliable components." In C.E. Shannon, and J. McCarthy (Eds.): Automata Studies, Princeton Univ. Press, Princeton, NJ, USA, 1956, pp. 43–98.

Wakerly, J. F. 1976, "Microcomputer reliability improvement using triple-modular redundancy," *IEEE Proceeding*. vol. 64, Jun. 1976, pp. 889–895.

Xiao, J., Lee, W., Jiang, J., Yang, X. 2016. "Circuit reliability estimation based on an iterative PTM model with hybrid coding." Microelectronics Journal, vol. 52, Jun. 2016, pp. 117–123.

AUTHOR BIOGRAPHIES

Dr. Walid Ibrahim received his B.Sc. in electrical engineering from Cairo University (Egypt) in 1992 and his Ph.D. in Computer Engineering from Carleton University (Ottawa, Canada) in 2002. In September 2004 he joined the College of Information Technology, UAE University (Al Ain, UAE), where he is currently a Professor of Computer Engineering. He is also the Director of the Emirates Institute for Learning Outcomes Assessment and the UAEU Learning Outcomes Assessment Coordinator. Before joining the UAE University, Dr. Ibrahim held several software R&D positions in multinational telecommunication and semiconductor companies including Nortel Networks, Alcatel, and PMC-Sierra. Dr. Ibrahim research interests include reliability of nano-architectures, reliability enabled EDA tools, ultra-low power designs, VLSI testing and design for testability, applied optimization techniques. His email address is walidibr@uaeu.ac.ae

Mrs. Hoda Amer received her B.Sc. in Computer Science from the American University in Cairo in 1993, and the M.Sc. in Computer Engineering from Carleton University (Ottawa, Canada) in 2001. In September 2004 she joined the College of Information Technology, UAE University (Al Ain, UAE), where she is currently an instructor with the Computer Science and Software Engineering Department. Her email is hamer@uaeu.ac.as