Hindawi
Journal of Electrical and Computer Engineering
Volume 2017, Article ID 5684902, 7 pages
https://doi.org/10.1155/2017/5684902

Research Article

Critical Gates Identification for Fault-Tolerant Design in Math Circuits

Tian Ban[1,2] and Gutemberg G. S. Junior[2]

[1]*School of Electronic and Optical Engineering, Nanjing University of Science and Technology, Nanjing 210094, China*
[2]*Department of Communications and Electronics, Institut Mines-Télécom, Télécom ParisTech, 75013 Paris, France*

Correspondence should be addressed to Tian Ban; tian.ban@njust.edu.cn

Received 12 October 2016; Revised 19 January 2017; Accepted 21 February 2017; Published 2 March 2017

Academic Editor: Wen B. Jone

Hardware redundancy at different levels of design is a common fault mitigation technique, which is well known for its efficiency to the detriment of area overhead. In order to reduce this drawback, several fault-tolerant techniques have been proposed in literature to find a good trade-off. In this paper, critical constituent gates in math circuits are detected and graded based on the impact of an error in the output of a circuit. These critical gates should be hardened first under the area constraint of design criteria. Indeed, output bits considered crucial to a system receive higher priorities to be protected, reducing the occurrence of critical errors. The 74283 fast adder is used as an example to illustrate the feasibility and efficiency of the proposed approach.

1. Introduction

With the technology scaling, electronic circuits are becoming more and more prone to faults and defects. Reliability analysis of logic circuits is emerging as an important parameter in deep submicron electronic technologies [1, 2]. It is especially critical for systems designed to be applied in space, avionics, and biomedical applications. In order to design reliable nanoelectronic devices, different fault-tolerant strategies have been extensively researched over the past years [3, 4].

Modular redundancy is a representative method which can provide reliability enhancement to the detriment of area overhead. Motivated by the need of economical fault-tolerant designs, researchers have been committed to searching for better trade-offs between reliability and overhead [5]. A hybrid redundancy method is proposed in [6], which combines information and hardware redundancy to achieve better fault tolerance. Sensitive transistors are protected in [7] based on duplicating and sizing a subset of transistors necessary for soft error tolerance in combinational circuits. In [8], Ruano et al. presented a method to automatically apply Triple Modular Redundancy (TMR) on digital circuits. The idea is to meet the reliability constraint while reducing the area overhead of typical TMR implementation.

Although all the aforementioned works provide reductions in the area overhead when compared to classical hardware redundancy systems, they do not take account of the usage profile of the results. In fact, a designer may use this additional information to make better decisions about which are the critical blocks of a circuit and then assign the desired priorities to protect them.

This work first proposes a different approach to identify critical logic blocks in math circuits. It relies on the fact of many digital systems and applications to tolerate some loss of quality or optimality in the primary outputs. In most cases, the trade-off in area is also associated with improvement of performance like faster operations, less power consumption, and so forth. The main idea is that different errors may have different consequences for different digital applications. For instance, in a binary output word, errors located in the most significant bits tend to be more critical than errors located in the least significant bits.

This paper is organized as follows. Section 2 introduces the practical reliability concept and explains the advantages of using such metric for reliability analysis. In Section 3, a fast

Table 1: Reliability for the output bits of three architectures for a 4-bit adder.

Architecture	b_3	b_2	b_1	b_0	R_{nominal}	$R_{\text{practical}}$
1	99%	99%	99%	95%	92.18%	97.63%
2	95%	99%	99%	99%	92.18%	94.17%
3	98%	99%	99%	95%	91.25%	96.64%

adder circuit 74283 is applied as a case study to illustrate and validate the proposed method. As an example, the estimate of peak signal-to-noise ratio (PSNR) with different fault-prone critical gates in image processing is considered, together with both analysis and comparison of results. Finally, Section 4 outlines some conclusions and suggestions for future works.

2. Reliability Evaluation

2.1. Nominal Reliability. Let $\mathbf{y} = b_{M-1}b_{M-2}\cdots b_1 b_0$ be a vector of M bits representing the output of a circuit. The reliability of a circuit is usually defined as the probability that it produces correct outputs, that is, the probability that all $b_i \in \mathbf{y}$ are correct 0(s) and 1(s). Given that the output bits are independent, this value, also known as *nominal reliability* [2], is conventionally expressed as in (1), where R_i stands for the reliability of b_i:

$$R_{\text{nominal}} = \prod_{i=0}^{M-1} R_i. \tag{1}$$

Let us now suppose that the circuit's output "$\mathbf{y}$" is coded by the use of a binary scheme, where b_{M-1} and b_0 stand for the most significant bit (MSB) and the least significant bit (LSB), respectively. Actually, MSB is the bit position in a binary number having the greatest numerical value. Therefore, error(s) occurring in MSB(s) will result in more remarkable disparities than in any other bit. By contrast, errors in LSB(s) may even be masked by the target application.

In spite of that, nominal reliability assigns equal reliability costs to the bits of "$\mathbf{y}$" as shown in (1). In fact, two different architectures for a logic function may have the same reliability value and one may still have a higher probability to provide more acceptable results than the other does. For instance, let us suppose that a designer obtains three different architectures for a 4-bit adder in which the output is coded using a binary scheme. Besides, he has to select one among them based on the reliability of the output. The reliabilities for the output bits of such architectures are presented in Table 1.

Analyzing the nominal reliability values for the obtained architectures, *Architecture 1* and *Architecture 2* are selected as the best solutions. Indeed, no distinction can be made between these two architectures regarding the nominal reliability value. However, as the output of this circuit is coded using a binary scheme, the first architecture would provide better results (smaller disparities) than the second one. Ideally, a more desirable analysis should take account of the amount of information that each bit of an output carries (or its importance) in order to assign progressively great costs to them. To tackle this problem, a new metric to analyze the reliability of a circuit with a multiple-bit output is presented in Section 2.2.

2.2. Practical Reliability. Practical reliability is a metric that can assess the importance of each output bit when analyzing the reliability of a circuit. It can be evaluated as shown in (2). The weight factor k_i allows a designer to adjust the importance of a specific output bit b_i to the output of the circuit. Notice that if $k_i = 1$, for all $0 \le i \le M-1$, the practical reliability expression (2) becomes the nominal reliability expression (1). In this work, a standard binary representation is considered so that k_i is calculated as shown in (3). Note that (2) can also be related to the probability that an error will cause a significant disparity on the output of a circuit (a critical error).

$$R_{\text{practical}} = \prod_{i=0}^{M-1} R_i^{k_i}, \tag{2}$$

$$k_i = \frac{1}{2^{(M-1)-i}}. \tag{3}$$

Although the proposed metric does not evaluate the true reliability of a circuit, this value takes account of both the reliability and the importance of an output bit for the target application. This is of great value for practical applications. For instance, let us analyze the architectures shown in Table 1. It can be noted that the practical reliability values are different from the values obtained with nominal reliability. Actually, even the order of the best architectures changes with the proposed metric. *Architecture 2*, which was previously deemed the best architecture together with *Architecture 1*, now is viewed as the worst choice due to the low reliability value of its MSB. In fact, practical reliability punishes architectures which present low reliability in critical bits, thus providing a designer with a more realistic result based on the target application.

3. Selectively Hardening Critical Gates

We know that critical gates should be hardened first in order to increase hardware usage efficiency and, at the same time, to minimize area overhead. The main idea here is to grade gates in math circuit to be protected based on critical factors. In this work, a critical factor explores not only the probability that an error will be introduced by a gate but also how critical this error will be in the target application as shown in Section 3.1.

3.1. Identifying Critical Gates. In order to explain and validate the proposed method, the 4-bit fast adder 74283 is employed (see Figure 1). The first module (M_1) produces the generate,

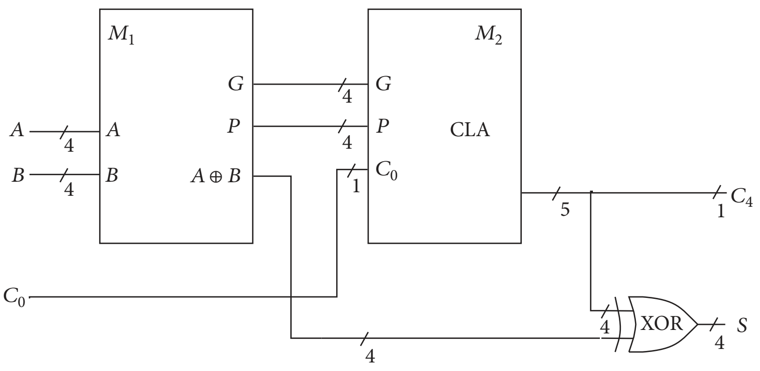

FIGURE 1: An illustration example: 4-bit adder 74283.

propagate, and XOR functions. The second module (M_2) is the carry-lookahead (CLA) realization for the carry function. Finally, the 8-bit XOR gate produces the sum function.

The fast adder 74283 has 9 inputs and 5 outputs and is composed of 36 logic gates and 4 buffers. All 40 blocks (gates and buffers) are considered as fault-prone. Further, it is supposed that these blocks (g_i ($i \in [0, 39]$)) are independent and labeled as shown in Figure 2.

The procedure of detecting which are the critical gates of this circuit takes two steps: first, a fault emulation platform, named FIFA [9], is used to inject faults due to Single Event Upsets (SEUs); next, *critical gates* are detected by analysis of errors that appear in the output vector.

The FIFA platform can generate one fault configuration per clock cycle. Further, it can inject a large number of simultaneous faults into the circuit [9]. However, in this work, it considers only the occurrence of single faults so that the platform injects just one fault each time. If the occurrence of multiple simultaneous faults is likely, the platform can be configured to deal with that.

Finally, the results, which are produced by the original and the faulty circuits, are compared bit by bit. If these results are different, it is concluded that the effects of the injected fault have been propagated to the output bits. Otherwise, it is concluded that the fault has been masked.

The fault injection emulation is performed to detect the critical factors. The idea is to inject a single fault in a gate g_i and analyze the output for all the possible input vectors. Then, for each output bit b_z, the number of errors S_z related to a single fault in g_i is evaluated (see Table 2). The columns S_{z_w} correspond to weighted versions of S_z. In our case study, as a standard binary representation is considered, S_{z_w} is obtained as shown in (4). Note that there are 2^9 possible input logic values for each faulty gate. All the simulation results are shown in Table 2.

$$S_{z_w} = 2^z \cdot S_z. \quad (4)$$

The *critical gates* are detected according to the results presented in Table 2. The more critical the gates are, the higher priorities they receive to be protected (in this case using TMR). Configuration of TMR based on this principle is more efficient in practical applications as shown in Section 3.2.

In fact, critical factors are assigned to the gates according to the number of weighted errors in Table 2. If the numbers of weighted errors are equal, gates that are closer to the primary outputs receive higher priorities. If the numbers of weighted errors and the distance to the primary outputs are both identical, gates presenting more reconvergent fan-outs are considered more critical. Gates whose three parameters are equal receive the same critical factor. Note that the rightmost column in Table 2 gives the critical factor for a gate g_i. The higher the factor number is, the more critical the gate will be. In this work, critical factors are assigned as integers $\in [0, 39]$.

3.2. Reliability Analysis and Comparison. Subsequent to obtaining the critical gates, the reliability of the redundant 74283 adder circuit is evaluated by using the SPR tool [1]. Further, the *signal reliability* of a given signal is considered as the probability that this signal carries a correct value. In fact, to assume that a binary signal x can carry incorrect information is equivalent to assuming that it can take four different values: correct zero (0_c), correct one (1_c), incorrect zero (0_i), and incorrect one (1_i).

The probabilities for occurrence of each one of these four values are represented as probability matrices shown as follows:

$$\begin{bmatrix} P(x = 0_c) & P(x = 1_i) \\ P(x = 0_i) & P(x = 1_c) \end{bmatrix} = \begin{bmatrix} x_0 & x_1 \\ x_2 & x_3 \end{bmatrix}. \quad (5)$$

The *signal reliability* of x, denoted as R_x, comes directly from (6), where $P(\cdot)$ stands for the probability function.

$$R_x = P(x = 0_c) + P(x = 1_c) = x_0 + x_3. \quad (6)$$

The SPR technique generates a matrix representing the output signal of a logical block, which explores the following information: the probability matrices representing the input signals for a given logical block, the logical function of such a block, and the probability that this block will not fail. In order to understand this procedure, let us consider a digital block b performing a logical function on a signal x to produce a signal y (see Figure 3). Now, assume that the probability that this operator will fail is represented by p, and $q = (1 - p)$

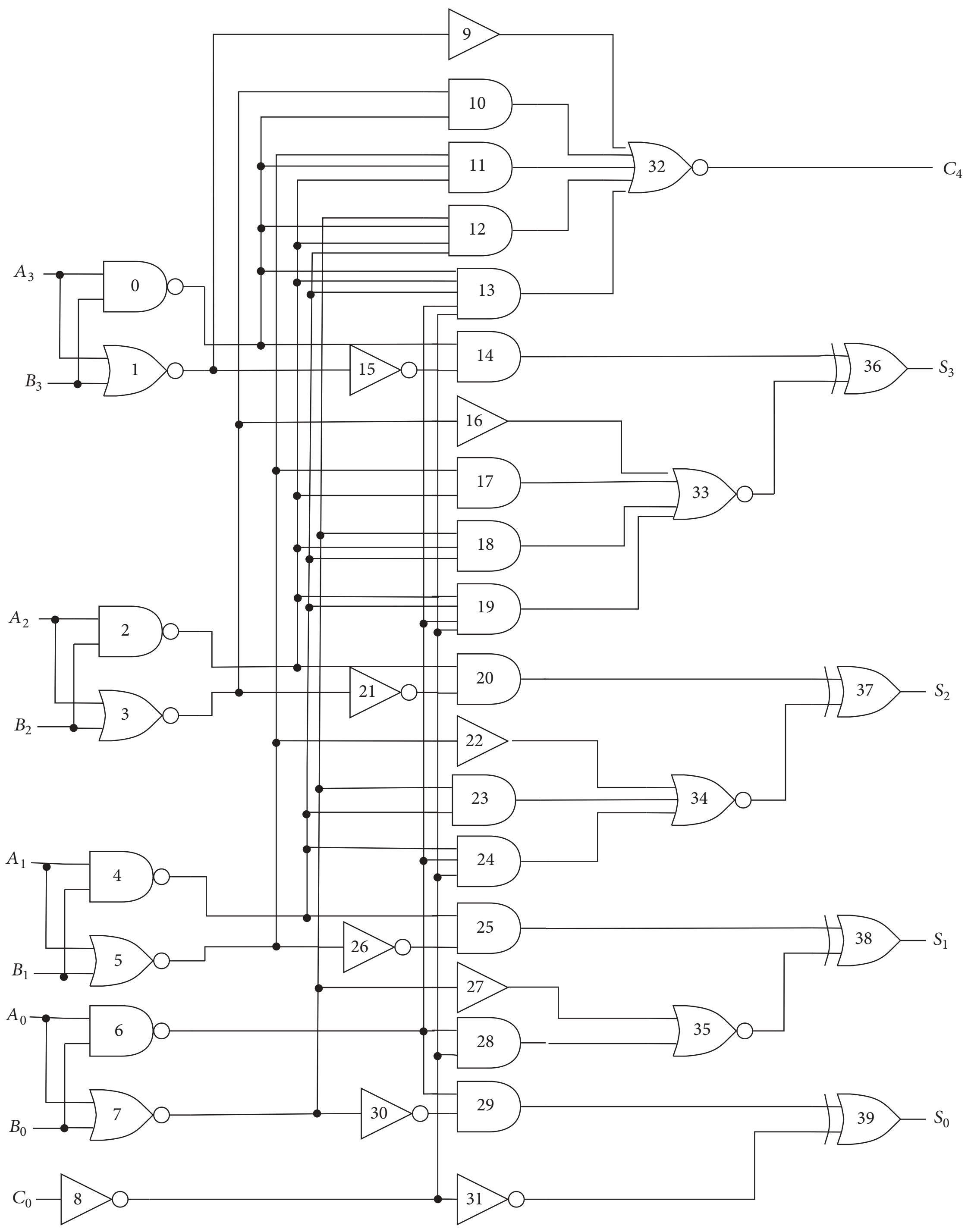

Figure 2: 74283 gate-level schematic.

represents the probability that it will not fail. Then, the reliability of y can be obtained by the following equation:

$$R_y = (x_0 + x_3) \cdot q + (x_1 + x_2) \cdot p. \quad (7)$$

As can be seen in (7), when the input signal is reliable, that is, $x_1 + x_2 = 0$, the reliability of the output signal is given by q, which stands for the probability of success of the logical block itself. This implies that, for fault-free inputs, the reliability of the output signal is given by the inherent reliability of the block that produces this signal.

Let us now consider hardware redundancy as the chosen redundancy technique to protect a logic block. Suppose that the area overhead constraint allows a designer to protect up to 5 gates. According to the critical factors presented in Table 2, gates g_{32}, g_1, g_3, g_0, and g_9 are selected by the proposed

Table 2: Error analysis for the gates of 74283.

g_i	S_0	S_{0_w}	S_1	S_{1_w}	S_2	S_{2_w}	S_3	S_{3_w}	C_4	C_{4_w}	$\sum$ errors$_w$	Critical factor
0	0	0	0	0	0	0	384	3072	192	3072	6144	36
1	0	0	0	0	0	0	384	3072	320	5120	8192	38
2	0	0	0	0	384	1536	192	1536	96	1536	4608	33
3	0	0	0	0	384	1536	320	2560	160	2560	6656	37
4	0	0	384	768	192	768	96	768	48	768	3072	25
5	0	0	384	768	320	1280	160	1280	80	1280	4608	32
6	384	384	192	384	96	384	48	384	24	384	1920	14
7	384	384	320	640	160	640	80	640	40	640	2944	23
8	512	512	256	512	128	512	64	512	32	512	2560	22
9	0	0	0	0	0	0	0	0	320	5120	5120	35
10	0	0	0	0	0	0	0	0	288	4608	4608	34
11	0	0	0	0	0	0	0	0	272	4352	4352	31
12	0	0	0	0	0	0	0	0	264	4224	4224	29
13	0	0	0	0	0	0	0	0	272	4352	4352	31
14	0	0	0	0	0	0	512	4096	0	0	4096	27
15	0	0	0	0	0	0	384	3072	0	0	3072	24
16	0	0	0	0	0	0	320	2560	0	0	2560	21
17	0	0	0	0	0	0	288	2304	0	0	2304	20
18	0	0	0	0	0	0	272	2176	0	0	2176	18
19	0	0	0	0	0	0	288	2304	0	0	2304	20
20	0	0	0	0	512	2048	0	0	0	0	2048	17
21	0	0	0	0	384	1536	0	0	0	0	1536	13
22	0	0	0	0	320	1280	0	0	0	0	1280	12
23	0	0	0	0	288	1152	0	0	0	0	1152	10
24	0	0	0	0	320	1280	0	0	0	0	1280	12
25	0	0	512	1024	0	0	0	0	0	0	1024	7
26	0	0	384	768	0	0	0	0	0	0	768	6
27	0	0	320	640	0	0	0	0	0	0	640	4
28	0	0	384	768	0	0	0	0	0	0	768	6
29	512	512	0	0	0	0	0	0	0	0	512	2
30	384	384	0	0	0	0	0	0	0	0	384	0
31	512	512	0	0	0	0	0	0	0	0	512	1
32	0	0	0	0	0	0	0	0	512	8192	8192	39
33	0	0	0	0	0	0	512	4096	0	0	4096	27
34	0	0	0	0	512	2048	0	0	0	0	2048	15
35	0	0	512	1024	0	0	0	0	0	0	1024	8
36	0	0	0	0	0	0	512	4096	0	0	4096	28
37	0	0	0	0	512	2048	0	0	0	0	2048	16
38	0	0	512	1024	0	0	0	0	0	0	1024	9
39	512	512	0	0	0	0	0	0	0	0	512	3

method as the five candidates to be protected. The method presented in [8], under the same area overhead constraint, applies redundancy in gates g_{32}, g_{36}, g_{37}, g_{38}, and g_{39}. As the occurrence of single errors is assumed, the protected blocks are considered reliable; that is, $q = 1$.

The reliability of the output bits for the original circuit and for the redundant configurations can be obtained by the SPR technique. Table 3 shows the reliability results for the respective configurations considering $q = 0.99$ for the gates not protected. It can be noted that both the nominal reliability and the practical reliability values are available. It is considered that the output of the 74283 adder comprises a 5-bit binary word so that the practical reliability can be evaluated from (2) and (3).

Analyzing the results presented in Table 3, it shows the effectiveness of the proposed approach. As commented above, the main idea is to take account of the impact of an error to the output of a circuit in order to prioritize the

Table 3: Reliability analysis of 74283 fast adder.

Reliability	No hardening	Method in [8]	Proposed method
S_0	94.07%	94.97%	94.07%
S_1	92.39%	93.26%	92.39%
S_2	91.80%	92.65%	92.43%
S_3	91.33%	92.17%	93.07%
S_4	94.60%	95.51%	97.15%
R_{nominal}	68.93%	72.24%	72.63%
$R_{\text{practical}}$	87.29%	88.89%	90.65%

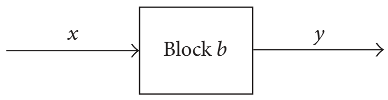

Figure 3: Generation of the output signal y from the input signal x processed by the digital block b.

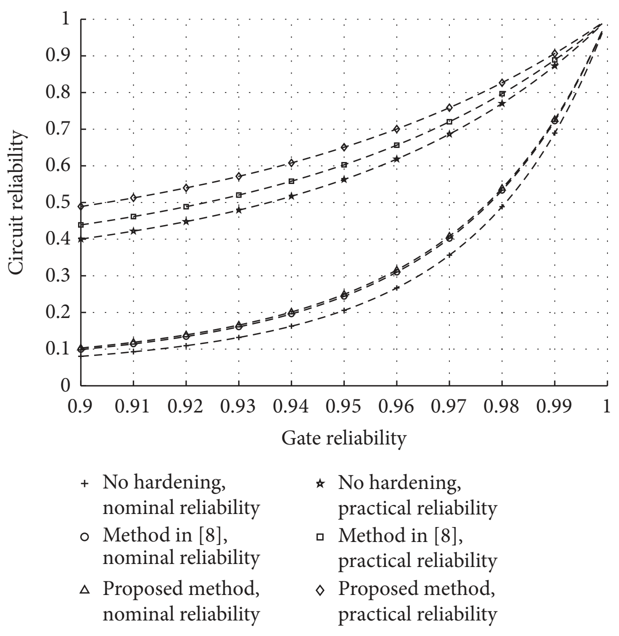

Figure 4: Simulation results for the 74283 fast adder.

reliability enhancement of the most important bits for the application. Indeed, the proposed hardening method shows a notable increase in the reliability of the most significant bits of the circuit (see Table 3). For instance, the reliabilities of S_0 and S_1 (LSBs) do not present any increase compared to the original circuit. Besides, the reliability of S_4 (MSB) presents the highest improvement as expected, once it is considered the most critical bit for this application.

Furthermore, it can be noted that, under the same area overhead, the nominal reliability increases by almost the same amount with both methods (see Figure 4). In fact, nominal reliability assigns equal reliability costs to the output bits of the 74283. This means that the output bits are considered as having the same importance to the system, so that the nominal reliability value does not distinguish in which bit the reliability was actually increased. In spite of that, practical reliability results can handle this problem and can indeed provide a sharper distinction between these two hardened architectures as shown in Figure 4.

4. Conclusion

In this paper, we presented a method to selectively apply hardening method to arithmetic circuits. Critical constituent gates are detected by taking account of not only the probability of error occurrence but also the impact of such error to the system. Indeed, bits considered critical to the target application receive higher priorities to be protected when the proposed method is employed.

Simulation results show the effectiveness of the proposed approach. This indicates that such critical gates should be hardened with priorities in order to increase hardware usage efficiency and to minimize area overhead simultaneously. The results could also be combined to approximate computing algorithms dedicated to fault-tolerant design [10]. Future works include approximating logic design based on gate grading results.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was partially supported by National Natural Science Foundation of China (Grant no. 61401205).

References

[1] D. T. Franco, M. C. Vasconcelos, L. Naviner, and J.-F. Naviner, "Signal probability for reliability evaluation of logic circuits," *Microelectronics Reliability*, vol. 48, no. 8-9, pp. 1586–1591, 2008.

[2] P. Zhu, J. Han, L. Liu, and F. Lombardi, "A stochastic approach for the analysis of dynamic fault trees with spare gates under probabilistic common cause failures," *IEEE Transactions on Reliability*, vol. 64, no. 3, pp. 878–892, 2015.

[3] T. Ban and L. Naviner, "Progressive module redundancy for fault-tolerant designs in nanoelectronics," *Microelectronics Reliability*, vol. 51, no. 9–11, pp. 1489–1492, 2011.

[4] P. K. Samudrala, J. Ramos, and S. Katkoori, "Selective triple modular redundancy (STMR) based single-event upset (SEU) tolerant synthesis for FPGAs," *IEEE Transactions on Nuclear Science*, vol. 51, no. 5, pp. 2957–2969, 2004.

[5] V. Hamiyati Vaghef and A. Peiravi, "Node-to-node error sensitivity analysis using a graph based approach for VLSI logic circuits," *Microelectronics Reliability*, vol. 55, no. 1, pp. 264–271, 2015.

[6] D. A. Tran, A. Virazel, A. Bosio et al., "A new hybrid fault-tolerant architecture for digital CMOS circuits and systems," *Journal of Electronic Testing*, vol. 30, no. 4, pp. 401–413, 2014.

[7] A. T. Sheikh, A. H. El-Maleh, M. E. S. Elrabaa, and S. M. Sait, "A fault tolerance technique for combinational circuits based on selective-transistor redundancy," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 1, pp. 224–237, 2016.

[8] O. Ruano, J. A. Maestro, and P. Reviriego, "A methodology for automatic insertion of selective TMR in digital circuits affected by SEUs," *IEEE Transactions on Nuclear Science*, vol. 56, no. 4, pp. 2091–2102, 2009.

[9] L. A. B. Naviner, J.-F. Naviner, G. G. Dos Santos Jr., E. C. Marques, and N. M. Paiva, "FIFA: a fault-injection-fault-analysis-based tool for reliability assessment at RTL level," *Microelectronics Reliability*, vol. 51, no. 9–11, pp. 1459–1463, 2011.

[10] H.-J. Wunderlich, C. Braun, and A. Schll, "Fault tolerance of approximate compute algorithms," in *Proceedings of the 34th VLSI Test Symposium*, p. 1, Las Vegas, Nev, USA, 2016.