DAO: Dual Module Redundancy with AND/OR Logic Voter for FPGA Hardening

Meisong Zheng
Aerospace Information System Research Center
Institute of Automation, Chinese Academy of Sciences
Beijing, China
meisong.zheng@ia.ac.cn

Zilong Wang, Lijian Li
National ASIC Engineering Technology Research Center
Institute of Automation, Chinese Academy of Sciences
Beijing, China
zilong.wang@ia.ac.cn, lijian.li@ia.ac.cn

***Abstract*—As device size shrinks, SRAM-based FPGAs are increasingly prone to be affected by single-event upsets (SEUs). SEU mitigation techniques for FPGAs are mostly expensive in terms of area and power costs. This paper proposes a new design for FPGA hardening using *dual-modular redundancy (DMR)*. The duplication operates on lookup-table (LUT) level, and each pair of identical LUTs will be voted by an *AND* or *OR* logic voter. By virtue of the fault-masking effect of AND/OR logic, certain faults in duplicated LUTs will not propagate to the next level of the hardened circuit. Results on MCNC'91 benchmarks show that the proposed method can reduce 90% faults with an area overhead of 100% additional number of LUTs, and the runtime of the proposed algorithm is much shorter than other existing methods.**

Keywords- FPGA; fault tolerance; dual modular redundancy; fault sensitivity

I. INTRODUCTION

SRAM-based FPGAs can be reprogrammed by users as many times as necessary, this flexibility makes it more and more widely used in different applications. But such flexibility relies on logic functions and interconnects implemented by SRAM cells, which are sensitive to various perturbations, hence FPGAs are more vulnerable to single-event upsets (SEUs) than application-specific integrated circuits (ASICs).

Triple-module redundancy (TMR) combines with scrubbing is an effective FPGA hardening technique, but the hardware overhead of TMR method is excessive, generally over 200%, give rise to high power dissipation and low working frequency. So certain applications found alternatives as selective TMR [1][2][3][4], which can reduce the area overhead with a small loss of SEU immunity.

In-place fault mitigation algorithms, such as EPP [5], ROSE [6] and IPD [7] make use of different logic masking techniques, mostly leveraging emerging FPGA architectures, to reduce fault rate in FPGAs. Those methods bring about low or no cost in area, but the fault masking effect is not obvious either.

Dual-module redundancy (DMR) can reduce hardware overhead to only 100%, but previous DMR techniques were mostly used for comparison, a different comparison result means that there is something wrong in one of the twin module and the FPGA system has to stop to repair the error. The major disadvantage of DMR is that it can offer neither fault localization nor fault-free module auto switch when the fault is discovered, which will cause a great decline on the working efficiency. To overcome this problem a method combines DMR and concurrent error detection (CED) is proposed [8], when error occurs it needs only one clock cycle in hold operation to detect the faulty module, and after that it will operate normally again without performance penalties. But the technique to encode and decode combinational logic circuit for CED is very difficult, and for complex circuit it is an impracticable task. Therefore [8] is not an alternative option for general use.

This paper provides a lookup-table (LUT) level *DMR* architecture, which adds an *AND* or *OR* logic after each duplicated LUT pair as a voter (DAO). This architecture can mask most FPGA errors induced by SEUs by means of logic gates nature: AND gate output remains 0 once one of the gate inputs is 0; OR gate output remains 1 once one of the gate inputs is 1. Different LUT outputs have different 0/1 probability, seeking for an optimal strategy to improve fault masking effect, whether an AND or OR logic voter will be added to an LUT pair depends on its 0/1 preference. With the help of abundant tristate buffer (BUFT) resources in Virtex FPGAs, the insertion of logic voter cause no hardware overhead to the system, hence our approach achieves the minimum overhead level in DMR domain, only 100%. Experiments on MCNC'91 benchmark circuits show that the SEU mitigation capability of our approach is significant.

The rest of this paper is organized as follows. Section II presents preliminaries on FPGA fault module and LUT 0/1 preference. Section III elaborates the fault masking effect provided by AND/OR logic. Section IV shows the proposed DAO as an algorithm. The experimental results are summarized in Section V, followed by conclusions in Section VI.

II. PRELIMINARIES

A. FPGA Fault Model Foundation for This Work

Different fault models have been founded by FPGA testers looking for a higher fault-coverage rate. In general, SEUs in SRAM-based FPGAs may undermine configuration bits in

either LUTs or interconnects, resulting in LUT memory bit-flip errors and interconnect resources stuck-at errors, respectively. Reference [9] proposed that stuck-at fault model can cover all faults in bit-flip fault model and also faults at primary inputs and primary outputs of the design. This paper intends to mask errors rather than locate the fault point, so we use stuck-at faults on LUT outputs to represent the memory bit-flip errors in the same LUT. Means that for a given LUT, all memory bit-flip errors can be represented by stuck-at errors on its output. For example in Fig.1, when the LUT inputs *B*,*A* are <0,1> the configuration bit C_1 is chosen, and the 0→1 bit-flip on C_1 has the same effect with stuck-at-1 fault on LUT output *E*. From the above we carry on our work with stuck-at fault module, all 0→1 (1→0) LUT memory bit-flip faults are regarded as stuck-at-1 (stuck-at-0) faults on its output line.

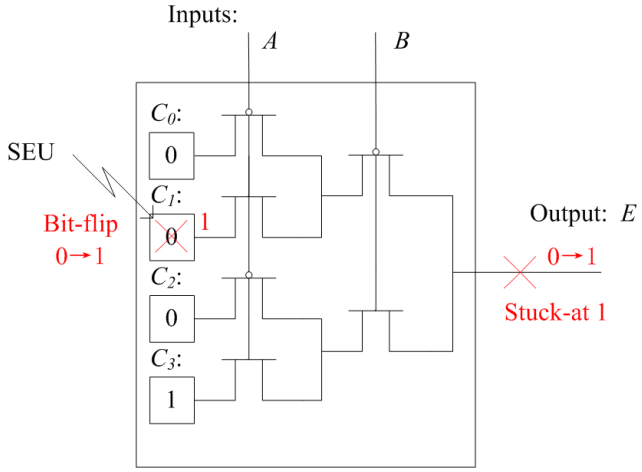

Figure 1. Fault modules in LUT

B. 0/1 Preferences of LUT Output Lines and Corresponding Fault Behavior

We use *signal probability* (P_{signal}) to represent the probability of each LUT output line to be sensitized to 1, an LUT is 1-perference if its P_{signal} is greater than 0.5, otherwise it is 0-preference. We use the work provided by [2] to calculate LUT output signal probabilities: sum up all access probabilities of configuration memories holding value 1. The access probability (P_{access}) of each LUT configuration memory can be calculated by multiplying each input P_{signal} or its complement, depending on the memory address. For example in the LUT depicted in Fig. 1, configuration bit C_1 can be accessed when the LUT inputs *B*,*A* are <0,1>, hence the access probability of C_1 can be obtained as follows:

$$P_{access}(C_1) = (1 - P_{signal}(B)) \times P_{signal}(A) \quad (1)$$

And the signal probability of LUT output *E* equals to the access probability of configuration bit C_3:

$$P_{signal}(E) = P_{access}(C_3) = P_{signal}(B) \times P_{signal}(A) \quad (2)$$

For circuits mapped to FPGA, we suppose the primary input signal probabilities are all 0.5, and the signal probability of other lines can be calculated level by level.

Only when a node with stuck-at-1(0) fault sensitized to 0(1) may the circuit show the fault at the primary outputs. So for 0-preference LUT lines we focus on masking the stuck-at-1 faults, and for 1-preference LUT lines stuck-at-0 faults are to be mitigated.

III. Fault Masking Effect Provided by Logic Gates

Logic 0 is control value for AND gates, means that when one input of an AND gate is 0, the output of the gate will remain 0 no matter what other inputs are, this property of AND gates can be used to mask stuck-at-1 faults in DMR circuits. OR gates have control value 1 hence can be used to mask stuck-at-0 faults in a similar way.

A. Fault Masking Effect of AND/OR Logic

Fig. 2 shows an AND gate with three inputs *A*, *B* and *C*, and an output *F*, suppose that the input pattern is <0,0,1> hence *F* values 0 as shown in Fig. 2(a), when the input value of *C* is disturbed by SEU and turned to 0, namely a stuck-at-0 fault happens on *C*, *F* remains value 0 on account of the control value 0 of inputs *A* and *B*; similarly, as shown in Fig. 2(b), when a stuck-at-1 fault happens on input *A*, *F* keeps right by reason of *B* still has control value of AND gate. In the above two cases, faults induced by SEUs (stuck-at-0 on *C* and stuck-at-1 on *A*) has been masked by AND gate.

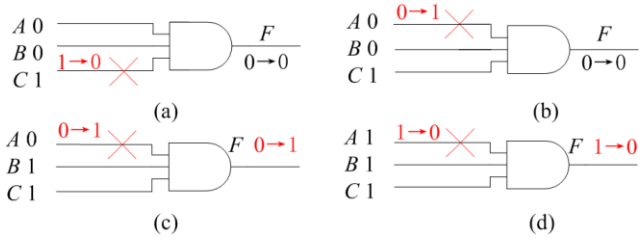

Figure 2. Fault masking effect of AND gate

On the contrary, suppose that the input pattern is <0,1,1> as shown in Fig. 2(c), the gate output *F* still values 0, when stuck-at-1 fault happens on input *A*, *F* turn out 1 because neither *B* nor *C* has control value of AND gate, and the final result is wrong. Same analysis fits the case depicted in Fig. 2(d). In these two cases, faults induced by SEUs (stuck-at-1 and stuck-at-0 faults on *A*) cannot be masked by AND gate.

In summary, only when one of the inputs holds control value 0 can an AND gate mask stuck-at faults on other inputs. Similarly, only when one of the inputs holds control value 1 can an OR gate mask stuck-at faults on other inputs.

B. AND/OR Gates Work as Voters in DMR Circuits

In this section, we will discuss the fault masking effect provided by AND logic in DMR circuits, and OR logic will work in the same way.

As shown in Fig. 3, an LUT is duplicated and the outputs of primary LUT (*P*) and redundancy LUT (*R*) are voted by an AND gate, suppose the primary LUT output sequence is

Identify applicable sponsor/s here. *(sponsors)*

{010101}, then *P* and *R* are all the same in case of no fault happens, so the output of AND gate *V* has a sequence of {010101}, same to the unduplicated LUT output sequence.

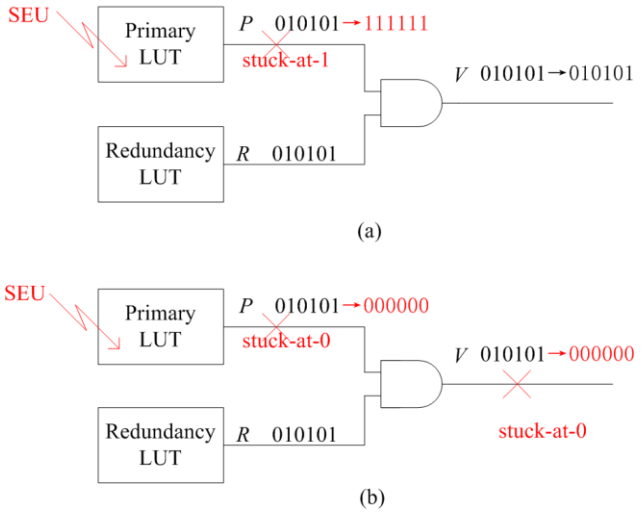

Figure 3. AND gate as voter

In Fig. 3(a), assume the primary LUT was hit by energetic particles and results in a stuck-at-1 fault on its output *P*. For the AND gate, the second, fourth and sixth input vectors are all <1, 1> and the output V values 1, same as fault-free circuits; the first, third and fifth input vectors change to <1, 0> and the output V values 0, same as fault-free circuits too. In this case, stuck-at-1 faults on *P* are masked by the AND gate, such faults on *R* can be masked in the same way according to the symmetry.

In Fig. 3(b), assume the primary LUT was hit by energetic particles and results in a stuck-at-0 fault on its output *P*. For the AND gate one of its input *P* change to control value, so the output V values 0 no matter what value the other input R is. In circuit illustrated in Fig. 3(b) half of the output sequences are different with fault-free circuit, and the error rate is 50%. In this case, stuck-at-0 faults on *P* cannot be masked by the AND gate, same to stuck-at-0 faults on *R* according to the symmetry.

In conclusion, for dual modular redundancy LUTs, AND logic can only mask stuck-at-1 faults induced by SEUs. OR logic can only mask stuck-at-0 faults under the same principle. As 0-preference lines are more prone to be affected by stuck-at-1 faults, they should be voted by an AND gate; and 1-preference lines are more prone to be affected by stuck-at-0 faults, they should be voted by an OR gate.

IV. FPGA HARDENING PROCESS

A. *Implementation of DAO on MCNC Benchmarks*

In this section, the algorithm for the DAO technique is described.

To begin with, the MCNC benchmark circuits are mapped to 4-inputs LUT module using RASP (Rapid System Prototyping) synthesis and mapping tool [10].We read the mapped circuit and calculate circuit grade in line 1. Lines 2-14 apply DAO method for the circuit level by level. Line 4 calculates signal probability of the chosen node and then the node is duplicated in line 5. As aforementioned, AND logic voter should be added to 0-preference LUT pair whereas OR to 1-preference, this operation is carried on line 6-10. Line 11 establishes connections between the twin LUTs and their voter, Line 12 replaces original line connections with the voted result.

Algorithm DAO_Implementation (Circuit C)

```
1:    MaxLevel ← Calculate_Level (C);
2:    for each level lev from 0 to MaxLevel do
3:      for each LUT l at level lev do
4:        calculate the signal probability P_signal of's output
5:        create a copy of l
6:        if (P_signal(l)⩾0.5) then
7:          create an AND voter
8:        else
9:          create an OR voter
10:     end if
11:     connect the output of l and its copy to the voter
12:     connect the voted output to the original connections of l
13:   end for
14:  end for
```

B. *AND/OR Logic Insertion in Virtex FPGAs*

Virtex FPGAs has two BUFTs associate with each configurable logic block (CLB), which can be used to build AND/OR voter circuits. The AND logic voter can be constructed by two BUFTs with a pull-down resistor as shown in Fig. 4(a), and OR logic voter can be constructed by two BUFTs with a pull-up resistor as shown in Fig. 4(b).

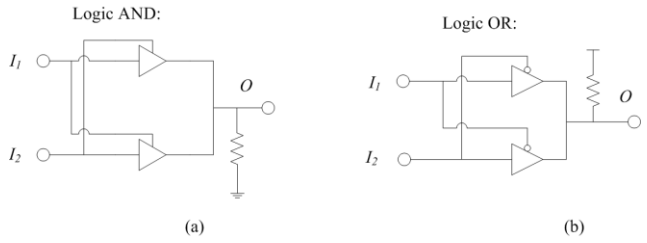

Figure 4. Logic voters built by BUFTs

V. EXPERIMENTAL RESULTS

We tested the proposed DAO method on standard MCNC'91 benchmarks and the results on 8 largest combinational circuits are shown in Table I. The algorithm was implemented in C++ and tested on a PC with a 3.2GHz quad core CPU and 4GB memory. We compared the overall performance of the original benchmark circuits, benchmark circuits hardened by DAO and benchmark circuits hardened by RTMR method provided by [2], and the results are listed in sub-columns titled "*ORI*", "*DAO*" and "*RTMR*" in Table I. To evaluate the fault masking effect, 1000 faults were injected to each circuit, and the numbers of faults that can propagate to the primary outputs were recorded and listed in columns marked as "No. of faults".

Failure rate represents how many times the circuit has failure in 10^9 hours, it can be calculated by the following formula:

$$FIT = R_{\text{SRAM}} \times N_{\text{LUT}} \times 2^K \times P_{\text{fault}} \quad (3)$$

Where R_{SRAM} is the raw error rate of an SRAM bit, typically 0.001-0.01 FIT/bit [11], we assumed R_{SRAM} =0.01 FIT/bit in our estimation. N_{LUT} is the number of LUTs every benchmark circuit employed which lists in columns marked as "No. of LUTs", *K* is the number of inputs of LUTs, we use 4-LUT module hence *K*=4. P_{fault} is the fault percentage recorded from the test experiment (No. of faults/1000). Failure rates computed with (3) according to the number of LUTs and number of faults are displayed in the columns marked as "FIT".

Hardware overheads of each circuit are listed in columns marked as "No. of LUTs". Since DAO use dual module redundancy strategy, the numbers of LUTs are doubled compare with the original circuits, and virtually identical on average with the RTMR method.

From Table I, it can be concluded that both of the DAO and RTMR approach reduces fault numbers significantly. For most circuits except des and ex5p, DAO costs a bit more overhead than RTMR and behaves better on fault masking.

Compared with the original circuit, DAO reduces fault rate by 90.78% (137.54 vs. 12.68) on average, similar to RTMR (91.12%, 137.54 vs. 12.2) and achieves 4.3×, 3.6× and 1.9× improvement respectively compared to the EPP (20.73%) [5], ROSE (25%) [6] and IPD (48.26%) [7] methods.

TABLE I. SUMMARY OF EXPERIMENTAL RESULTS

Circuit	No. of LUTs			No. of faults			FIT		
	ORI	DAO	RTMR	ORI	DAO	RTMR	ORI	DAO	RTMR
alu4	1522	3044	2959	275	4	7	66.97	1.95	3.31
apex2	1878	3756	2974	350	1	0	105.17	0.60	0.00
des	1591	3182	3850	699	122	68	177.94	62.11	41.89
misex3	1397	2794	2723	512	5	17	114.44	2.24	7.41
pdc	4575	9150	8661	248	5	13	181.54	7.32	18.01
seq	1750	3500	3371	413	6	7	115.64	3.36	3.78
spla	3690	7380	7222	460	9	13	271.58	10.63	15.02
ex5p	1064	2128	2442	394	39	21	67.07	13.28	8.21
average	2183	4366	4275	418.88	23.88	18.25	137.54	12.68	12.20

As shown in Table II, runtime of DAO is 1.362s for an average circuit scale of 2183 LUTs. Actually, algorithm in DAO needs only once calculate for every LUT node. In-place mitigation methods like EPP, ROSE and IPD always needs time-consuming algorithms hence their runtime are all several times longer than DAO. RTMR utilize fault simulation to select sensitive LUTs at last, the author did not give out real run time, but 10,000 faults simulation for each circuit will not take a short time.

TABLE II. COMPACTION OF RUNTIMES

Circuit	No. of LUTs	runtimes/s		
		DAO	*EPP*	*IPD*
alu4	1522	0.804	190	1466
apex2	1878	0.997	63	1137
des	1591	0.909	15	1430
misex3	1397	0.634	81	1235
pdc	4575	3.756	4253	3429
seq	1750	0.919	16	1659
spla	3690	2.49	1446	3270
ex5p	1064	0.389	18	795
average	2183	1.362	760	1803

VI. CONCLUSION

This paper presents a new FPGA SEU mitigation technique DAO, in which AND or OR logics are used as DMR voters. Compared to traditional DMR methods, this architecture needs neither reset to recover from errors nor additional error judgment circuit to switch to the fault-free part. Compared to the STMR method, DAO keeps a close SEU immunity level at a similar area overhead, but the runtime is much less. Compared to in-place mitigation techniques, DAO provides a more effective SEU immunity level and much less runtime.

Results on MCNC'91 benchmark show that circuits hardened by DAO can reach a very high SEU immunity level at a reasonable area overhead. Since SEUs are probability events, DAO can guarantee the stability of system operation combines with a certain frequency of FPGA scrubbing.

REFERENCES

[1] C. Carmichael. "Triple Module Redundancy Design Techniques for Virtex FPGA," Xilinx Application Notes 197, San Jose, USA: Xilinx, 2001.

[2] V. Chandrasekhar, S. N. Mahammad, V. Muralidaran, and V. Kamakoti, "Reduced Triple Modular Redundancy for Tolerating SEUs in SRAM-based FPGAs," in *Proc. MAPLD*, Wash., D.C., Sep. 2005.

[3] P. K. Samudrala, J. Ramos, and S. Katkoori, "Selective triple Modular redundancy (STMR) based single-event upset (SEU) tolerant synthesis

for FPGAs," *IEEE Trans. Nucl. Sci*., vol. 51, no. 5, pp. 2957-2969, Oct. 2004.

[4] B. Pratt, M. Caffrey, J. F. Carroll, P. Graham, K. Morgan, and M. Wirthlin, "Fine-Grain SEU Mitigation for FPGAs Using Partial TMR," *IEEE Trans. Nucl. Sci.*, vol. 55, no.4, pp. 2274-2280, 2008.

[5] H. Keheng, H. Yu, and L. Xiaowei, "Reliability-Oriented Placement and Routing Algorithm for SRAM-Based FPGAs," *IEEE Tran. VLSI Systems*, vol. 22, no.2, pp. 256-269, Feb. 2014.

[6] H. Yu, F. Zhe, H. Lei, and R. Majumdar, "Robust FPGA resynthesis based on fault-tolerant Boolean matching," in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design*, Nov. 2008, pp. 706-713.

[7] L. Ju-Yueh, F. Zhe, and H. Lei, "In-place decomposition for robustness in FPGA," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, pp. 143–148 , Nov. 2010.

[8] F. G. de Lima Kastensmidt, G. Neuberger, R. F. Hentschke, L. Carro, and R. Reis, "Designing fault-tolerant techniques for SRAM-based FPGAs," *IEEE, Design & Test of Computers*, vol. 21, no.6, pp. 552-562, 2004.

[9] J. Borecky, M. Kohlik, P. Kubalik, and H. Kubatova, "Fault Models Usability Study for On-line Tested FPGA," in Proc. *DSD, 14th Eur. Conf.*, pp. 287-290, 2011.

[10] J. Cong, J. Peck, and Y. Ding, "RASP: A general logic synthesis system for SRAM-based FPGAs," in *Proc. ACM 4th int. symp. on FPGAs*, Monterey, CA, pp. 137-143, 1996.

[11] Device reliability report, 2nd ed., Xilinx Inc., August 16, 2013.

Contact Information:

Name: Meisong Zheng
Address: Institute of Automation, Chinese Academy of Sciences Beijing, China, 100190
Phone numbers: 18201400865
Email address: zhengmeisong@126.com
meisong.zheng@ia.ac.cn