Fault Injection Method and Voter Design for Dual Modual Redundancy FPGA Hardening

Meisong Zheng and Zilong Wang
Institute of Automation,
Chinese Academy of Science
Beijing, 100190, China
{meisong.zheng & zilong.wang}@ia.ac.cn

Lijian Li
Institute of Automation,
Chinese Academy of Science
Beijing, 100190, China
lijian.li@ia.ac.cn

Abstract—**Hardware overhead and system reliability has always been in a dilemma in the world of FPGA designing. This paper provides a hardware overhead free dual modular redundancy (DMR) voter scheme using the abundant and unused carry chains in Xilinx FPGAs. And the experimental evaluation on MCNC benchmarks are carried on real FPGA and logic-simulation based fault injection method, respectively. Experimental results under the two methods are identical; hence validates the effectiveness of logic-simulation method and the DMR hardening scheme we have designed.**

Keywords-FPGAs; fault tolerance; fault injection; redundancy

I. Introduction

Today static-random access memory (SRAM) based field-programmable gate array (FPGAs) permits replacing application-specific integrated circuit (ASICs) in many applications [1, 2]. But FPGAs are highly sensitive to radiation-induced single-event effects (SEEs) [3, 4]; hence the utilization of FPGA in critical systems has been limited due to reliability issues.

Redundancy is an efficient FPGA hardening technique[5], triple module redundancy (TMR)[6, 7] can significantly improve the reliability of a design, but it is expensive in area required for implementation. dual module redundancy (DMR) requires less hardware overhead but could not mask but only detect errors unless with additional logic implemented[8].

Fault injection can be performed with various abstraction levels, from real FPGA device to software simulation models that runs on a customer computer [9-11]. Experiments under real radiation environment[12] are easy to design, but are limited to laboratory conditions and are more prone to damage the FPGA device. Due to the intrinsic complex of FPGAs and commercial security kept by FPGA manufacturers, the software simulation method is not easy to design and may be inaccurate.

This paper provides an area overhead-free DMR voter scheme designed with abundant carry-chains in Xilinx Virtex FPGAs. The scheme was applied to a set of MCNC benchmark circuits to evaluate the effect, and then we present an experimental evaluation of the technique’s efficiency by measuring the error rates of a circuit (cm152a from the MCNC suite) injected with every look up table (LUT) configuration bit faults on a real FPGA.

II. DMR Voter Design and Real FPGA Implementation

We use the proposed fault masking DMR scheme based on fault masking effects of AND/OR logic provided in [13]. The scheme is based on this principle: Logic 0 is control value for AND gates, logic 1 is control value for OR gates. Control value means that the output of a gate is totally controlled by certain logic value of one of its input. For example 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 0→1 faults in DMR circuits, and OR gates to 1→0 faults in a similar way.

A. Building DMR voters with carry-chains

The proposed hardening technique operates at LUT level, we designed the AND/OR logic DMR voters using abundant and underused carry-chains in Xilinx FPGAs; hence the insertion of voters does not bring about additional hardware overhead. A simplified diagram of carry-chain logic in one slice is shown in Fig. 1, it is comprised with a multiplexer and exclusive-or gate. Real FPGAs contain more complex data selections and relative connections for the multiplexer and exclusive-or gate than Fig. 1.

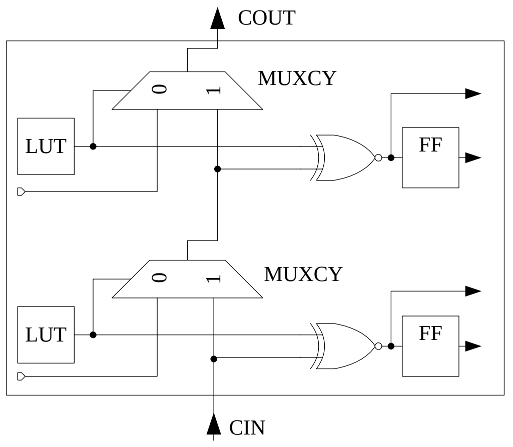

Figure 1. A simplified carry-chain in one slice

As shown in Fig. 2(a), an AND logic voter can be built with a 2 to 1 multiplexer:

$$O_{\text{AND}} = \overline{O_1} \cdot 0 + O_1 \cdot O_2 = O_1 O_2 .$$

As shown in Fig. 2(b), an OR logic voter can be built with a 2 to 1 multiplexer combined with an exclusive-or gate:

$$O_{\text{OR}} = \overline{O_2} \cdot (O_1 \oplus O_2) + O_2 \cdot O_2 = O_1 + O_2 .$$

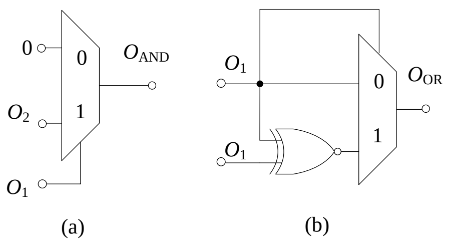

Figure 2. AND OR logic constructed by carry-chains

B. An example of cm152a hardened by the proposed scheme

For the purposed of providing a more clearly explanation, we choose cm152a from MCNC benchmark as an example. Microelectronics Center of North Carolina (MCNC) benchmark suite was published for MCNC International Workshop on Logic Synthesis, 1991. It included logic synthesis and optimization benchmark sets from ISCAS'85 and ISCAS'89 in addition to some other benchmarks collected from industry and academia [14].

Cm152a is an 8 to 1 multiplexer with eleven inputs and one output, the state of three select inputs *pi*, *pj*, *pk* controls which of the eight inputs of *pa, pb, pc, pd, pe, pf, pg, ph* will be chosen to the output *pl*. The 4-input LUT structure of cm152a synthesized by Rapid System Prototyping (RASP) synthesis and mapping tool [15] is shown in Fig. 3.

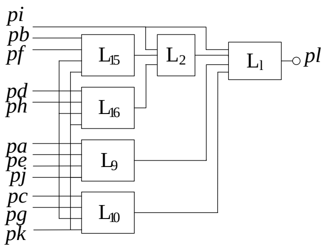

Figure 3. 4-input LUT structure circuit of cm152a

We use the DMR hardening method proposed in [13], because of all LUTs in the cm152a circuit are 0-preference (more prone to have an output of logic 0). AND logic voters are inserted as shown in Fig.4, the AND logic voters are built by a 2 to 1 multiplexer in carry-chains as illustrated in Fig 2(a).

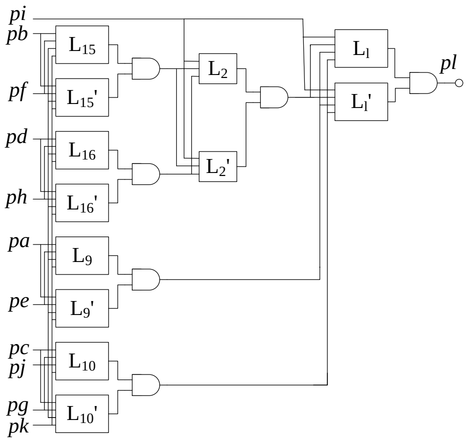

Figure 4. DMR hardening of cm152a

C. Hardening Method Implementation on Real FPGA

For consistency, we use Xilinx Virtex-4 Slice/CLB primitives to implement the cm152a circuit on a real FPGA, the topological structure carried on real FPGA are kept the same with results synthesis by RASP. Because netlists synthesized by RASP boasts more trimly structure; hence are easier to be read and disposed by our C++ procedure for logic-simulation experiments in Section III.

Take the LUT L_9 implementation as an example; we realized the function of L_9 in a Virtex-4 FPGA as follows:

```
LUT4 #(
      .INIT(16'h3120)     // Specify LUT Contents
 )LUT4_INST_L9 (
     .O(inter9),          // LUT general output
      .I0(pk),            // LUT input
      .I1(pj),            // LUT input
      .I2(pe),            // LUT input
      .I3(pa)             // LUT input
 );
```

Where .INIT(16'h3120) means the configuration bits of LUT L_9 is 0x3120; .I0 to .I3 are inputs of LUT L_9, and .O is the output of LUT L_9. For the DMR hardened circuit, the duplication of LUT L_9 and the AND logic voter insertion:

```
LUT4 #(
      .INIT(16'h3120)     // Specify LUT Contents
 )LUT4_INST_L9_1 (
     .O(inter9_1),        // LUT general output
      .I0(pk),            // LUT input
      .I1(pj),            // LUT input
      .I2(pe),            // LUT input
      .I3(pa)             // LUT input
 );
 LUT4 #(
      .INIT(16'h3120)
 )LUT4_INST_L9_2 (
     .O(inter9_2),
      .I0(pk),
      .I1(pj),
      .I2(pe),
      .I3(pa)
 );
 MUXCY_L MUXCY_L_inst9 (
     .LO(inter9),          // Carry local output signal
     .CI(inter9_2),        // Carry input signal
```

Identify applicable sponsor/s here. If no sponsors, delete this text box. (sponsors)

```
        .DI(0),                 // Data input signal
        .S(inter9_1)            // MUX select, tie to '1' or LUT4 out
    );
```

with the ucf statements:

```
    INST "LUT4_INST_L9_1"  LOC = slice_X3Y11 | BEL = F;
    INST "LUT4_INST_L9_2"  LOC = slice_X3Y11 | BEL = G;
```

The ucf statements are used to constrain the duplicated LUT resides in the same slice with the original LUT. The netlist of duplicated L_9 and the AND logic voter synthesized by Xilinx ISE is shown in Fig. 5, captured from the FPGA Editor tool. The two LUTs are same in configuration bits and inputs, the output of upper LUT inter9_2 is routed back as the carry input of MUXCY, so the output of MUXCY:

$$XB=\overline{inter9\_1}\cdot 0+inter9\_1\cdot inter9\_2=inter9\_1\cdot inter9\_2$$

The implantation of MUXCY realized the AND logic voter for inter9_1 and inter9_2.

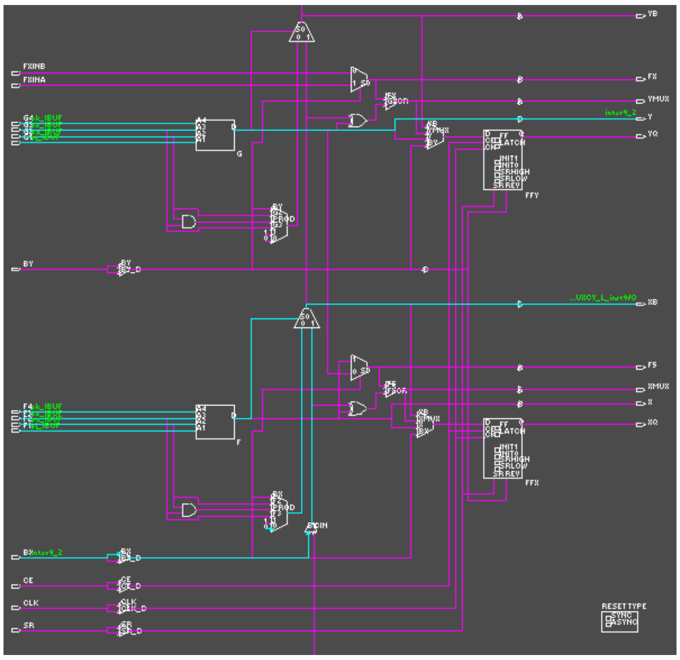

Figure 5. Duplicated LUTs voted by an AND logic voter

III. Logic-Simulation Based Fault Injection

We tested the proposed DMR method on standard MCNC'91 benchmarks. In this section, we carried on our work as a logic-simulation fault injection mechanism. The algorithm was implemented in C++ on a PC with 3.2GHz quad core CPU and 4GB memory.

A. *Experimental Flow*

The logic-simulation based fault injection experiment are designed as follows:

1) Preparing the input file: The proposed hardening method is tested on the combinational circuits of the MCNC'91 benchmark suit. The netlists are mapped into 4-input LUTs using RASP.

2) DMR hardening operation: The DMR hardening algorithm proposed in [13] is coded in C++ language on a PC.

3) Fault injection and evaluation: We reverse the configuration bit once a time for every LUT. Once a configuration bit is flipped, test vectors are applied to inputs and the outputs are compared with an error-free circuit. The number of fault will be added with one when a difference is detected. The next configuration bit will be reversed after all test vectors have been applied or a fault is detected on this reversed configuration bit.

Once the SEU simulator injects a fault on an LUT, the fault effect depends on its redundancy property. Only on the following two cases will the injected fault really reverse the LUT configuration bit:

- LUT pair voted by an AND logic whereas the fault is 1→0.
- LUT pair voted by an OR logic whereas the fault is 0 →1.

Otherwise the fault point LUT remains its configuration memory, means that it is immune to the injected faults.

B. *Results and Analysis*

Some results of combinational MCNC'91 benchmark circuits are shown in Table I. Column 2 titled with "N_{inputs}" shows the number of primary inputs of each circuit; column 3 titled with "N_{lut}" shows the number of LUTs of each circuit; and column 4 titled with "N_{bits}" shows the total number of configuration bits of each circuit. Columns 5 titled with "F_{ORI}" shows the number of configuration bits propagates fault to the output while flipped in the original circuit; Columns 6 titled with "F_{DMR}" shows the number of configuration bits propagates fault to the output in the DMR hardened circuit. The last column titled with "Reduced" shows the percent of faults reduced by the proposed DMR method compared with the original circuits.

TABLE I. Experimental Results on Simulation

Circuits	N_{inputs}	N_{lut}	N_{bits}	F_{ORI}	F_{DMR}	Reduced
cm152a	11	6	96	79	23	70.89
c8	28	39	624	410	122	70.24
c880	60	174	2784	1712	500	70.79
alu2	10	197	3152	1666	400	75.99
miex3	14	1397	22352	10903	1314	87.95
alu4	14	1522	24352	12165	994	91.80
des	256	1591	25456	18580	4400	76.32
seq	41	1750	28000	9993	796	92.03
apex2	39	1878	30048	7531	215	97.15
Average			15207	7004	974	86.09

From Table I, we can see that 46% (7004/15207) of the configuration bits are SEU sensitive without protection; while the DMR hardened FPGA reduced that proportion to 6.4% (974/15207). Compared with the original circuit, DMR method can reduce 86% faults on average, for some circuits like *alu4* and *seq* even more than 90%.

IV. Fault Injection Experiments on Real FPGA

In order to verify the validity of the proposed method, we implemented the technique on a Virtex-4 FPGA on a customer designed test circuit board. The board is based on XC4VSX35 with a 100MHz oscillator and 4 LEDs. We implemented the cm152a circuit on the FPGA and inject faults on its configuration bits.

A. Experimental Scheme

The experiment scheme is shown in Fig. 6. The golden part is error-free circuit with no fault injection. The ORI and DMR part represents original cm152a circuit and the DMR hardened cm152a circuit with one configuration bit fault in one LUT. The configuration bit faults are injected at the time of FPGA configuration. When FPGA configuration is finished, Input vectors are applied to the 3 circuits with the frequency of 100MHz on the test board, and the outputs of ORI and DMR are compared with the golden one. Differences between the outputs drive the test board LEDs, LED1 represents error detected in ORI circuit; while LED0 represents error detected in DMR hardened circuit.

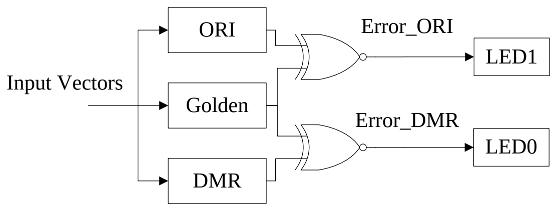

Figure 6. Real FPGA test scheme

B. Results and Discussion

The cm152a circuit is composed with one 3-input LUT and five 4-input LUTs, hence the total LUT configuration bits equals to $5\times2^4+1\times2^3=88$. So the FPGA is reconfigured 96 times, the state of LED1 and LED0 are recorded after each reconfiguration. Lighten of LED1 represents an error detected in ORI circuit; while lighten of LED0 represents an error detected in DMR circuit.

The experiment showed 79 times of ORI circuit errors (Error_ORI=79) accomplished with 23 times of DMR hardened circuit errors (Error_DMR=23). This result behaves the same with the logic-simulation fault injection mechanism described in Section-III, validates the correctness of the simulation method we have designed, and further demonstrates the reasonability of the DMR hardening method.

V. Conclusion

This paper presents a new DMR voter built with the abundant carry-chain resources in Xilinx FPGAs, which brings about no hardware overhead when insertion. We also provided a method to implement the DMR scheme and voter on real FPGAs using Xilinx Virtex primitives. Both of simulation experiments and real FPGA configuration bits flipping are carried on MCNC'91 benchmarks and results showed that the DMR method is effective in terms of fault masking and overhead saving. Also, identical results on the two experiments proved that the simulative fault injection method is reasonable.

References

[1] A. Lifa, P. Eles, and Z. Peng, "A Reconfigurable Framework for Performance Enhancement With Dynamic FPGA Configuration Prefetching," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on,* vol. 35, pp. 100-113, 2016.

[2] Y. R. Qu and V. K. Prasanna, "High-Performance and Dynamically Updatable Packet Classification Engine on FPGA," *Parallel and Distributed Systems, IEEE Transactions on,* vol. 27, pp. 197-209, 2016.

[3] P. Bernardi, M. S. Reorda, L. Sterpone, and M. Violante, "On the evaluation of SEU sensitiveness in SRAM-based FPGAs," in *On-Line Testing Symposium, 2004. IOLTS 2004. Proceedings. 10th IEEE International*, 2004, pp. 115-120.

[4] Altera, "Reliabilty report 55 1h," 2013.

[5] J. Tarrillo, F. Lima Kastensmidt, P. Rech, C. Frost, and C. Valderrama, "Neutron Cross-Section of N-Modular Redundancy Technique in SRAM-Based FPGAs," *Nuclear Science, IEEE Transactions on,* vol. 61, pp. 1558-1566, 2014.

[6] F. Lima, C. Carmichael, J. Fabula, R. Padovani, and R. Reis, "A fault injection analysis of Virtex FPGA TMR design methodology," in *Radiation and Its Effects on Components and Systems, 2001. 6th European Conference on*, 2001, pp. 275-282.

[7] U. Kretzschmar, A. Astarloa, J. Lazaro, M. Garay, and J. Del Ser, "Robustness of different TMR granularities in shared wishbone architectures on SRAM FPGA," in *Reconfigurable Computing and FPGAs (ReConFig), 2012 International Conference on*, 2012, pp. 1-6.

[8] F. G. de Lima Kastensmidt, G. Neuberger, R. F. Hentschke, L. Carro, and R. Reis, "Designing fault-tolerant techniques for SRAM-based FPGAs," *Design & Test of Computers, IEEE,* vol. 21, pp. 552-562, 2004.

[9] E. Johnson, M. Caffrey, P. Graham, N. Rollins, and M. Wirthlin, "Accelerator validation of an FPGA SEU simulator," *Nuclear Science, IEEE Transactions on,* vol. 50, pp. 2147-2157, 2003.

[10] G. L. Nazar, P. Rech, C. Frost, and L. Carro, "Radiation and Fault Injection Testing of a Fine-Grained Error Detection Technique for FPGAs," *Nuclear Science, IEEE Transactions on,* vol. 60, pp. 2742-2749, 2013.

[11] H. M. Quinn, D. A. Black, W. H. Robinson, and S. P. Buchner, "Fault Simulation and Emulation Tools to Augment Radiation-Hardness Assurance Testing," *Nuclear Science, IEEE Transactions on,* vol. 60, pp. 2119-2142, 2013.

[12] H. Quinn, K. Morgan, P. Graham, J. Krone, and M. Caffrey, "Static Proton and Heavy Ion Testing of the Xilinx Virtex-5 Device," in *Radiation Effects Data Workshop, 2007 IEEE*, 2007, pp. 177-184.

[13] Z. W. Meisong Zheng, Lijian Li, "DAO Dual Module Redundancy with ANDOR Logic Voter for FPGA Hardening " 2015.

[14] R. Njuguna, "A Survey of FPGA Benchmarks," vol. http://www.cse.wustl.edu/~jain/cse567-08/index.html, November 24, 2008 2008.

[15] J. Cong, J. Peck, and Y. Ding, "RASP: A general logic synthesis system for SRAM-based FPGAs," in *Proceedings of the 1996 ACM fourth international symposium on Field-programmable gate arrays*, 1996, pp. 137-143.