Validation of the Proposed Hardness Analysis Technique for FPGA Designs to Improve Reliability and Fault-Tolerance

Abdul Rafay Khatri[1], Ali Hayek[2], Josef Börcsök[3]
Department of Computer Architecture and System Programming,
University of Kassel, Kassel, Germany

Abstract—**Reliability and fault tolerance of FPGA systems is a major concern nowadays. The continuous increase of the system's complexity makes the reliability evaluation extremely difficult and costly. Redundancy techniques are widely used to increase the reliability of such systems. These techniques provide a large area & time overheads which cause more power consumption and delay, respectively. An experimental evaluation method is proposed to find critical nodes of the FPGA-based designs, named "hardness analysis technique" under the proposed RASP-FIT tool. After finding the critical nodes, the proposed redundant model is applied to those locations of the design and the code is modified. The modified code is functionally equivalent and is more hardened to the soft-errors. An experimental set-up is developed to verify and validate the criticality of these locations found by using hardness analysis. After applying redundancy to those locations, the reliability is evaluated concerning failure rate reduction. Experimental results on ISCAS'85 combinational benchmarks show that a min-max range of failure reduction (14%-85%) is achieved compared to the circuit without redundancy under the same faulty conditions, which improves reliability.**

Keywords*—*Dependability; fault injection; fault tolerance; reliability; single event effects

I. Introduction

Field Programmable Gate Array (FPGA) has been involved in various applications in the last couple of decades, such as aerospace, biomedical instrumentation, safety-critical systems, and spacecraft, due to their remarkable features. These features include parallelism, reconfiguration, separation of functions, self-healing capabilities, overall availability, low cost and low design turn-around time [1], [2]. Therefore, FPGA has become the core of many embedded applications. SRAM-based FPGA devices are sensitive to Single Event Effects (SEE), which can be caused by various sources, such as α-particles, cosmic rays, atmospheric neutrons, heavy-ion radiations and electromagnetic radiations (x-rays or gamma rays) [2], [3], [4]. When a charged particle hits a critical node of FPGA-based design, it generates the transient pulse which can produce a bit-flip effect. This phenomenon is known as Single Event Upsets (SEU). The failure rate for a component or system is the number of failures that occur per unit time.

Verilog HDL is one of the most widely used languages for implementing the design structure for Application Specific Integrated Circuit (ASIC) and FPGA-based designs [5]. Verilog HDL describes designs in various abstraction style, for example, gate, data-flow, and behavioural levels. For small designs, gate abstraction style is used, and testing & verification processes can directly and easily be applied to the designs. At this level, designs look more similar to the actual hardware design. For large designs, data-flow and behavioural abstraction styles are adopted to develop and implement the specification of the design in an HDL code [6].

The reliability of Integrated Circuits (ICs) is profoundly affected due to technology scaling. Due to shrinkage size of components, the reliability of the device is a challenge nowadays. One way to improve the reliability of these designs is redundancy, but it increases the area and time overheads. The reliability can be defined as "It is the probability that the circuit output is correct even in the presence of faults" [7]. Several SEE mitigation techniques have been presented in the literature to protect the FPGA-based designs from SEE effect. The reliability of the FPGA systems is improved by various error mitigation schemes such as multiple-redundancy with voting, Triple Modular Redundancy (TMR), hardened memory cell level, and Error Detection And Correction (EDAC) coding. Among all SEU mitigation techniques, TMR has become the most common practice because of its straightforward implementation and reliable results [8], [2], [9], [10], [11]. These mitigation methods reduce the failure rate (SER) in combinational logic in integrated circuits and improve the reliability.

An experimental set-up is presented to find the critical nodes of the designs. This set-up works on the code-level of the design, i.e. Verilog HDL. The proposed hardness analysis technique is developed under the tool "RASP-FIT" in a continuation of the previous work [6], [1]. In this work, the primary objective is to validate the proposed hardness technique which is used to find the most critical nodes of the design and to increase the fault tolerance capability and reliability of the FPGA-based designs by reducing the soft error failure rate. The redundancy of these sensitive locations is achieved by modifying the fault injection control unit. However, improving reliability by applying redundancy to the sensitive locations by code modification, area-overhead calculations, power and delay analyses are in progress. These locations are the most sensitive nodes to the permanent and transient faults. Few fault models such as bit-flip and stuck-at (1/0) are involved in the experimentation. Various benchmark circuits are considered & evaluated, and authors found that a significant improvement in reliability is achieved. The proposed approach is a simple, straightforward, and easy to use.

The remainder of this paper is organised as follows: The related work is presented in Section II. Section III presents an overview of the RASP-FIT tool and explains the proposed methodology to find sensitive locations in the design. Section IV describes the experimental set-up and their components in detail. Results are discussed in Section V. Finally, Section VI concludes the paper and presents some directions for future work.

II. Related Work

Reliability concerning soft errors has become a crucial issue in digital circuits due to technology scaling nowadays. Soft errors are transient errors that can cause digital circuits to operate incorrectly. If a soft error occurs in the combinational logic, it results in a Single Event Transient (SET). On the other hand, if it occurs in the memory cell itself, it results in a Single Event Upset (SEU). Both SET and SEU have a significant impact on circuit operation, and they should be adequately treated [12]. Soft Error Rate (SER) is a measurement evaluation metric for the sensitivity of the digital design to soft errors. SER estimation can be evaluated using two methods, i.e. dynamic and static. Fault injection and logic simulation techniques practice dynamic methods [13].

Authors in [14] demonstrated that, with increasing technology generations, soft errors caused more effects in logic devices than memory devices. Therefore, soft errors are becoming a major concern in digital systems. To overcome the effect of soft errors in combinational circuits, several fault tolerance techniques have been introduced in the literature. Fault tolerance techniques for combinational circuits are classified into three main categories: hardware redundancy-based, synthesis-based, and physical characteristics-based techniques [12].

The reliability of the FPGA systems is improved by various error mitigation scheme such as Triple Modular Redundancy (TMR). The problem with TMR technique is that it requires high area overhead nearly more than 200%. In [15], authors proposed a method for reducing the effect of SEU and called it "Selective Triple Modular Redundancy (STMR)". In this method, the conventional TMR technique is applied to selective gates of the whole design. These gates are more sensitive than other gates in the design. The signal probabilities of its inputs determine the sensitivity of a gate to an SEU. Soft error mitigation scheme based on logic implication is proposed in [16]. According to this method, the selective functionally redundant wires are attached to the combinational logic of a circuit. The procedure to find these functionally redundant wire is illustrated in this work.

Authors in [17] described the co-hardening technique, which is a technique that tries to reduce protection overheads complementing software mitigation techniques with hardware techniques in a selective way. Probabilistic Transfer Matrix (PTM) is a gate-level approach for the accurately measured reliability of combinational designs. The drawback of PTM is long simulation runtime and memory usage. Therefore, this technique is upgraded and called Efficient Computation PTM [18]. Authors in [13] proposed a new method for SER estimation based on vulnerability evaluation of the design gates. They introduced a probabilistic vulnerability window concept which considered three masking factors (i.e. logical, electrical and timing).

In this work, the validation of the proposed hardness analysis technique is presented in detail, which shows the reduction in failure rate and hence achieves the improvement in reliability. In comparison with the above works in general, our experimental set-up provides more enhancement in reliability with both permanent and transient fault models.

III. RASP-FIT Tool and Hardness Analysis

The RASP-FIT (RechnerArchitektur and SystemProgrammierung)–German name of the institute– Fault Injection Tool has the capability to instrument FPGA-based designs, for fault simulation and emulation of designs. This tool is designed specifically for the FPGA-based designs, which are written in Verilog at different abstraction levels. The tool consists of three major functions, namely, `fault_injection()`, `static_compaction()` and `hardness_analysis()`. RASP-FIT tool, with its Graphical User Interface (GUI), is developed in Matlab. All these functions are developed in Matlab under the function `RASP_FIT()`.

A. Verilog Code Modifier under RASP-FIT tool

The RASP-FIT tool has the capability to instrument FPGA-based designs, written in Verilog at different abstraction levels. This tool modifies or instruments the code by inserting faults. At each abstraction level, the way of modification of the code is different and also fault models are defined at that abstraction level [6]. In the modification process, it also adds Fault Injection, Selection and Activation (FISA) control unit in the target design. The FISA unit selects and activates the particular fault in the whole design.

Test and reliability evaluation using fault injection techniques require the modification of the design. Modification of Verilog code for various FPGA-based designs and obtaining the compact test vectors for maximum fault coverage using static compaction technique, also hardness analysis provides the information about the critical nodes of design are presented in previous work [19], [1], [6]. Fig. 1 shows the modified code of the original design with fault injection control unit. The RASP-FIT tool injects faults in every possible location in the design. This example shows the insertion of a bit-flip fault model in the design. For understanding purpose, we call it "full-faulty module" in this paper. The FISA control unit of the full-faulty module can select and activate all faults in the design. More detail about the RASP-FIT tool is described in [20].

B. Hardness Analysis: Identification of Sensitive Nodes

The sensitive location is the location in a System Under Test (SUT), where the occurrence of any fault results in a failure. The sensitive locations of the SUT are obtained using the proposed hardness analysis approach. According to this approach, these locations are more or less equally sensitive to bit-flip and stuck-at (1/0) faults. In this method, we obtained the more efficient test vectors/patterns which detect more faults than others. We selected a set-point value for each design and each fault model. The experimental way to obtain the sp-value is described in [21]. The procedure for

```
// Original design
module c17 (N1,N2,N3,N6,N7,N22,N23);

input N1,N2,N3,N6,N7;
output N22,N23;
wire N10,N11,N16,N19;

nand NAND2_1 (N10, N1, N3);
nand NAND2_2 (N11, N3, N6);
nand NAND2_3 (N16, N2, N11);
nand NAND2_4 (N19, N11, N7);
nand NAND2_5 (N22, N10, N16);
nand NAND2_6 (N23, N16, N19);

endmodule
```

```
// Compilable faulty design
module c17_1 (select,N1,N2,N3,N6,N7,N22_f1,
    N23_f1);
input N1,N2,N3,N6,N7;
output N22_f1,N23_f1;
wire N10,N11,N16,N19;
input[3:0] select;
reg f0,f1,f2,f3,f4,f5,f6,f7,f8,f9,f10,f11;
always @ (select) begin
if (select == 4'd0) begin
f0=fis;f1=0;f2=0;f3=0;f4=0;f5=0;f6=0;f7=0;f8
    =0;f9=0;f10=0;f11=0;end
else if (select == 4'd1) begin
f0=0;f1=fis;f2=0;f3=0;f4=0;f5=0;f6=0;f7=0;f8
    =0;f9=0;f10=0;f11=0;end
.
.
.

else if (select == 4'd11) begin
f0=0;f1=0;f2=0;f3=0;f4=0;f5=0;f6=0;f7=0;f8
    =0;f9=0;f10=0;f11=fis;end
else begin
f0=0;f1=0;f2=0;f3=0;f4=0;f5=0;f6=0;f7=0;f8
    =0;f9=0;f10=0;f11=0;end
end
nand NAND2_1 (N10,f0^ N1,f1^ N3);
nand NAND2_2 (N11,f2^ N3,f3^ N6);
nand NAND2_3 (N16,f4^ N2,f5^ N11);
nand NAND2_4 (N19,f6^ N11,f7^ N7);
nand NAND2_5 (N22_f1,f8^ N10,f9^ N16);
nand NAND2_6 (N23_f1,f10^ N16,f11^ N19);
endmodule
```

Fig. 1. Original code (left) & instrumented compilable design code (right) by RASP-FIT.

obtaining the qualified test patterns is outlined in algorithm 1. Once the efficient test patterns are obtained, then hardness analysis is performed on them using the RASP-FIT tool under hardness analysis function. The sequel describes the procedure to calculate hardness.

Algorithm 1 Dynamic compaction algorithm in a nutshell

```
1: Input patterns are applied using LFSR from total vector
   space $T_S$
2: Fault detections are counted for each pattern applied
3: Sum of detections are compared with set-point value
4: if Is sum greater than or equal to set-point value then
5:     Stored pattern as qualified test vectors $T_q$
6:     Increment the number of $T_q$ count
7: else
8:     Apply new pattern to the SUT
9: end if
10: Go to step 2
11: Stop simulation when $T_q$ count reaches 500
```

The whole experimental set-up is shown in Fig. 2 in order to find sensitive locations. In the first step, a Verilog design file is applied as an input to the RASP-FIT tool. The RASP-FIT tool generates the faulty copies of the original design with evenly distributed faults in them. The user selects the fault models and the number of defective modules. This tool generates a file (top module design) which contains the instantiations of different modules, comparator logic, dynamic compaction logic to select efficient test vectors and memory logic to store responses. Once the faulty modules are generated, the Xilinx ISE and Modelsim tools are used to create the project and simulate the designs. Repeat this procedure for every fault models, e.g. bit-flip, stuck-at 0 and stuck-at 1 fault models. The simulation results are stored in text (*.txt) file which is further applied as an input to the RASP-FIT tool to perform the hardness analysis. The simulation set-up generator, shown in Fig. 4, is explained in Section IV to validate the hardness analysis technique. The technique is described in the sequel.

Hardness or Hard to Detect (HTD) is the characteristic of those faults which can be detected very rarely. The hardness analysis receives text files as input, reads them and stores the data in a matrix form. Authors call it Fault Matrix (FM), which is an arrangement of qualified input patterns and the detection of faults for the input patterns given in Eq. 1,

$$\text{FM} = \begin{bmatrix} P_1 & F_{1,1} & F_{2,1} & \cdots & F_{N,1} \\ P_2 & F_{1,2} & F_{2,2} & \cdots & F_{N,2} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ P_i & F_{1,i} & F_{2,i} & \cdots & F_{N,i} \end{bmatrix} \tag{1}$$

Where P_1 to P_i are qualified input patterns obtained during fault-experiment, and the array of detected faults for a particular pattern are placed in a row of the matrix. When

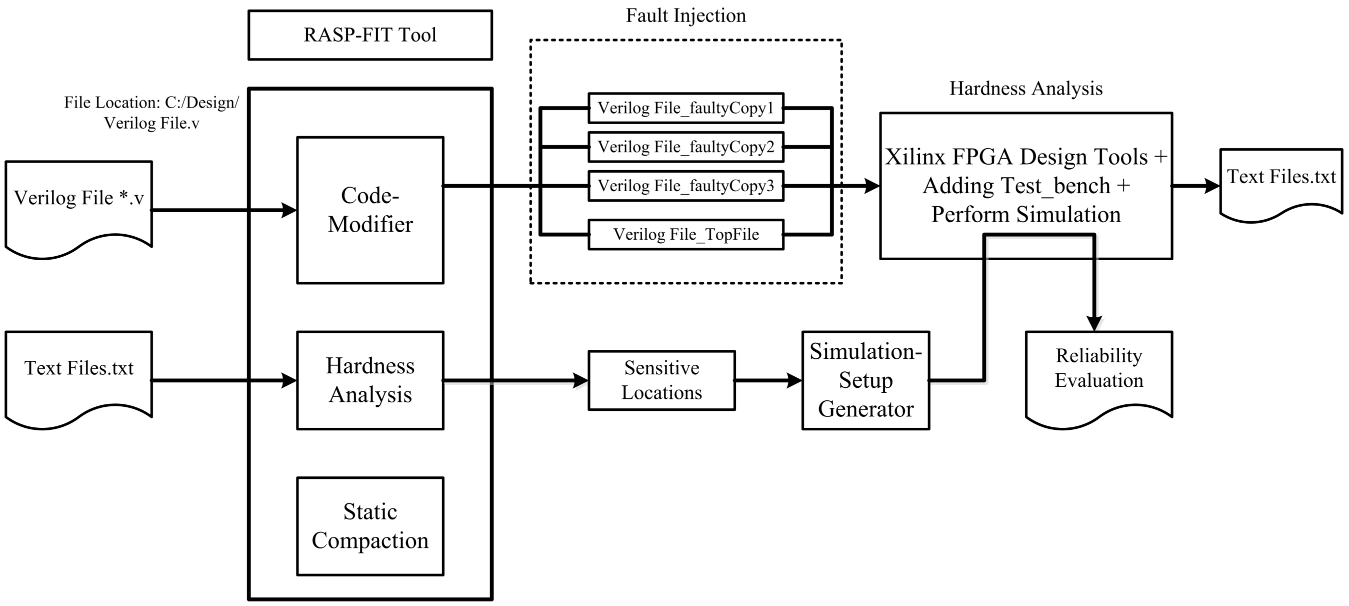

Fig. 2. Overall flow of RASP-FIT tool, hardness analysis and simulation set-up generator.

the specific fault is detected, it gets value '1', otherwise gets value '0'. Hardness *(H)* of individual fault is calculated by Eq. 2,

$$H = \left(1 - \frac{\text{No. of Fault Detections}}{\text{Total Patterns in FM}}\right) X100 \qquad (2)$$

If the hardness of a fault results in 100%, it means the fault is not detectable for any input; hence, it is called an un-testable or undetectable fault. On the other side, a hardness of 0% shows the detection of fault for all test vectors, which means that the portion of the circuit where the fault has appeared is very critical to fault attacks. The hardness of each fault for every fault model is calculated and placed in a matrix, named Hardness Matrix (HM) as shown in Eq. 3,

$$\text{HM} = \begin{bmatrix} H_{f_1,bf} & H_{f_2,bf} & ... & H_{f_N,bf} \\ H_{f_1,sa0} & H_{f_2,sa0} & ... & H_{f_N,sa0} \\ H_{f_1,sa1} & H_{f_2,sa1} & ... & H_{f_N,sa1} \end{bmatrix} \qquad (3)$$

Hardness values of each fault for every fault model are arranged column-wise. Each column is compared with the different threshold values one after the other for each fault to get the number of sensitive locations. Threshold values are used to get the number of the most sensitive areas to less vulnerable places. There are four levels considered for obtaining sensitive locations empirically, i.e. 35%, 55%, 75% and 95%. Once, we identify the sensitive locations for different threshold values, we will apply the proposed redundant model on to those locations only and obtained the modified code for the design under experiment. The modified code is more hardened than the original design. The proposed redundant model is described in the sequel.

Development of Redundant Model: The sensitive location is merely a wire/net either connecting two gates or input of gates. Redundant model is developed based on the Duplication With Comparison (DWC) technique, in which the sensitive location is duplicated, and their outputs are compared using XOR gate. The output of the XOR is feed to the select input of a multiplexer, which routes the correct output. It is noted that this redundant model is used for the validation of the proposed hardness analysis technique. These sensitive locations should be made hardened to improve reliability and reduce the soft error rate. The sensitive node is replaced by the proposed redundant model as shown in Fig. 3, for the wire/net without fan-out. This work is in progress to modify the original code according to the redundant model to add hardening.

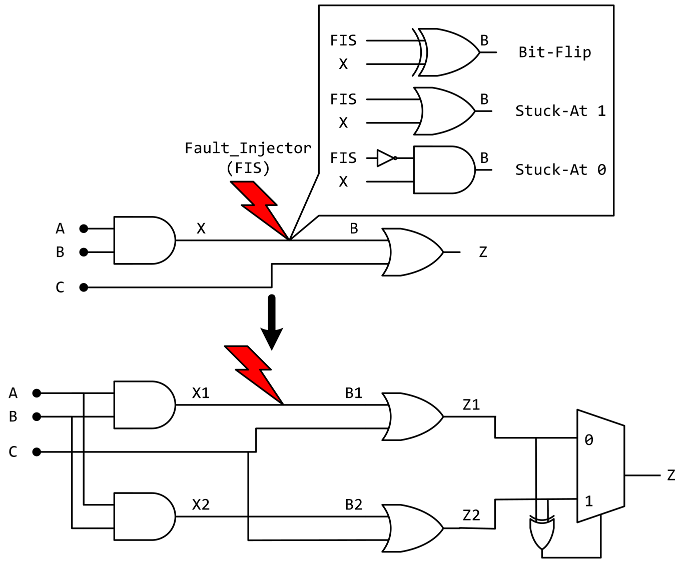

Fig. 3. Redundant fault model for validation of approach.

IV. Experimental Set-Up

To validate the efficiency of the proposed hardness analysis technique in the process of improving fault tolerance capabilities and reliability of FPGA-based designs, we created an experimental set-up. Various benchmark designs are evaluated, and the results are presented in this work. Fig. 4 shows the experimental set-up for the validation of the proposed hardness analysis technique. Fig. 4 shows the description of each components in the sequel.

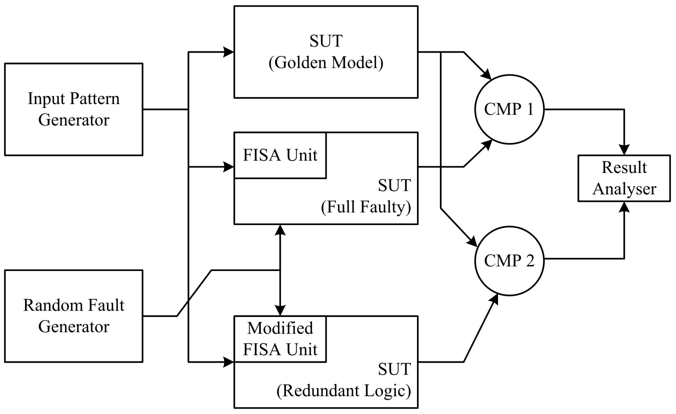

Fig. 4. Experimental set-up for the validation for the hardness analysis technique.

A. Input Pattern Generator

This block is used to generate different input patterns which are simultaneously applied to both the original and faulty copies of the SUT. For small circuits having few input ports, we generate and apply all possible combination of inputs. For a large number of inputs, random input patterns are generated and applied to the golden, full-faulty and redundant SUT simultaneously. These patterns are generated using the Linear Feedback Shift Register (LFSR), defined in the test-bench. Fig. 5 shows the code for the generation of input patterns randomly for the SUT `(c432.v)` having 36 inputs.

```
// Random input generator code for 36
     input design

reg [35:0] e = 36'hF59611d09; //seed

always @(clk) begin
e = { e[34:0],  e[35] ^ e[34] };
{N1,N4,N8,N11,... ,N73,N76} = e ;
end
```

Fig. 5. Generate random input patterns in test-bench.

B. Random FISA (Fault Generator)

In this experimental set-up, random faults are generated, selected and activated in both the full-faulty module and the redundant module. Poisson distribution is used to generate random faults because of the following reasons:

1) The event is something that can be counted in whole numbers.
2) Occurrences are independent. Therefore, one occurrence neither diminishes nor increases the chance of another.
3) The average frequency of occurrence for the period in question is known.
4) It is possible to count how many events have occurred.

Poisson distributed numbers can be generated using this code in the test-bench. During the simulation, for each pattern, 40 faults are selected and activated using Random FISA unit in each copy of the SUT shown in Fig. 6 for each pattern applied. The total of 500 patterns is stored with their responses in a text file for the further calculations.

```
always
  begin
    #5; select = $dist_poisson(seed, mean);
  end
```

Fig. 6. Part of test_bench to generate random faults for FISA unit.

C. Golden Model (SUT)

Golden SUT is the original module without faults, and it is a reference design for the comparison between the faulty SUT and the SUT with redundant logic. Various combinational logic circuits from ISCAS'85 benchmark are considered for the validation of the proposed hardness analysis technique.

D. Full-Faulty SUT with FISA Unit

Faulty SUT is the modified benchmark design by injecting faults in every possible location. The RASP-FIT tool generates the faulty SUT. This tool is capable of instrumenting the Verilog design, written in all abstraction levels, by injecting various types of permanent and transient faults in all possible location. The user can choose between the types of fault models for analysis. Fig. 1 depicts the original code and modified code generated by RASP-FIT tool.

E. Redundant SUT with Modified FISA Unit

Once the information about the critical nodes is obtained, redundancy is applied to them. Fig. 3 shows the proposed redundant model. When the fault occurs on this sensitive location, the multiplexer logic masks it. However, we modified the FISA unit included in the SUT with redundant logic in this work. With this modification, the FISA unit does not activate the particular sensitive fault in the design, even though, it is selected by random FISA input generator.

F. Comparators

One comparator logic is used to compare the responses of golden SUT with the full-faulty SUT, whereas another comparator logic compares the responses of golden SUT with the SUT having modified FISA unit for sensitive locations. The value of comparator is logic '1' when both responses are different from each other and logic '0' in another case.

G. Result Analyser

Result analyser is developed in Matlab. The failure rate for both modules is stored during simulation in a text file. Result analyser program reads the text file containing responses and calculates the number of fault detections for both designs.

Reliability Improvement Calculations: Based on the definition of reliability mentioned earlier in the introduction section, the reliability of the combinational logic system can be improved by reducing the failure rate. There are two main methods which are most widely used, i.e. error detection and retry and error masking. In the proposed technique, full-faulty and the module with redundancy are run under the same conditions. Both modules are compared with the golden reference module design which is a fault-free design. Several faults are randomly selected and activated, and fault detections for both modules are stored in a text file. Reliability evaluation program is written in Matlab which takes responses of the experimental set-up stored in a text file and count the number of errors occurred for both modules. Improvement in reliability is calculated, in term of failure rate reduction, using the Eq. 4,

$$FR = \frac{SER_{fullFaulty} - SER_{redundant}}{SER_{fullFaulty}} X100\% \qquad (4)$$

Where FR is a failure rate reduction (SER reduction) achieved after making the critical locations as redundant by using modified FISA unit, $SER_{fullFaulty}$ is a fault detection for full-faulty SUT and $SER_{redundant}$ is a fault detection for redundant SUT with modified FISA unit.

V. Results and Discussion

The primary objective of the work is to obtain the high reliability with applying the selective redundancy to some sensitive locations. In this way, a cost-effective solution is obtained with high reliability. These following steps describe the procedure used to perform the proposed technique:

1) Using the RASP-FIT tool, first, perform the fault injection analysis to generate faulty code of the design using Verilog code modifier function.
2) Repeat the step 1 for all fault models, e.g. bit-flip, stuck-at 0 and stuck-at 1.
3) Create a simulation set-up using Xilinx ISE and Modelsim tool.
4) Using the RASP-FIT tool, perform hardness analysis and find the sensitive locations of the design.
5) Modify the fault control unit for the most sensitive places as described earlier.
6) Create a simulation set-up as explained in Section IV.
7) Run simulations for all fault models and save the simulation results in text files separately.
8) Import all these files in Matlab and evaluate reliability.

To explain the proposed method, we consider a simple benchmark design `c17.v` from ISCAS'85 designs as an example (Fig. 1). In this design, the RASP-FIT finds and injects a total of 12 faults in the whole design. Hardness analysis is performed, and it is found that the 4 locations are most sensitive (at Threshold = 55%) and we have to apply for the redundancy on these locations. Fig. 7 shows the critical nodes for various benchmark designs at different threshold values. The redundant module is proposed in Section III, but it is not used here because, in this work, we are validating the proposed hardness analysis technique regarding failure rates. In

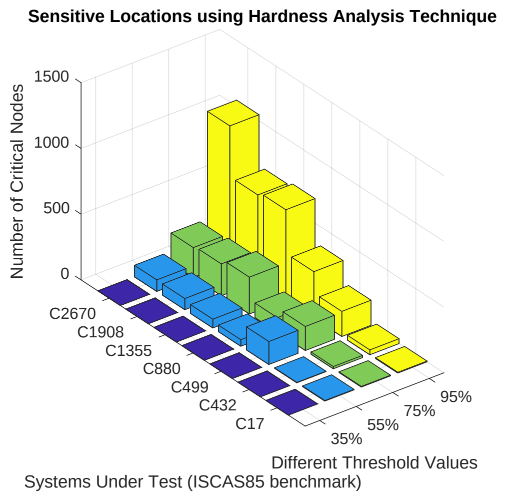

Fig. 7. Number of critical nodes for different threshold.

this work, we modified the FISA unit such that these four faults never selected and activated. Total number of fault selected and activated during the experiment can be obtained using the Eq. 5,

$$T_{fault} = P_i \times nSec \times F_{copy} \qquad (5)$$

Where T_{fault} is the total number of fault injected per experiment for each fault model, P_i is the number of patterns considered, i.e. 500, nSec is the number of faulty copies of SUT and F_{copy} is the number of faults selected and activated per copy of SUT during the experiment.

We perform the simulation of a full-faulty SUT and the SUT with modified FISA unit. Soft error rate (failure rate) for the full-faulty module is observed with random fault injection. Similarly, the detection rate for the module with redundancy is also recorded for each fault model separately. Reliability improvement is calculated and presented in Table I. The results validate that the obtained locations using the proposed hardness analysis technique are susceptible. By applying redundancy or masking/hardening techniques to these locations, it shows the significant improvement in the reliability as shown in the last column of Table I.

VI. Conclusion

In this paper, the authors introduced a method to increase the fault tolerance capability and the reliability of combinational circuits. The idea is based on first finding the sensitive locations of the design using the proposed hardness analysis technique and then applying redundancy/hardening to those nodes only. In this way, the overall fault tolerance of the original circuit is enhanced and the failure rate is also reduced as well. Experimental results on ISCAS'85 combinational benchmarks show that we can get a maximum reliability improvement of 85%. The approach is straightforward and easy to use.

Future work includes the analysis of area-overhead due to redundant logic applied to the sensitive locations. Also, the delay and power consumption analyses are the core areas after

TABLE I. IMPROVEMENTS IN RELIABILITY AND FAULT TOLERANCE CAPABILITY OF FPGA-BASED DESIGNS

SUT	Fault Model	Total Fault (T_{fault}) Injected/Experiment	Number of Sensitive Nodes	Faulty Module Detection Rate	Redundant Module Detection Rate	SER Reduction (%)
c17	Bit-flip	20,000	4/12	9759	3435	64.802
	Stuck-at 0			6104	1973	67.68
	Stuck-at 1			3689	1403	61.79
c432	Bit-flip	60,000	17/336	5641	4441	21.28
	Stuck-at 0			2581	2208	14.95
	Stuck-at 1			3005	2180	27.45
c499	Bit-flip	60,000	177/408	24689	3607	85.39
	Stuck-at 0			7058	1118	84.15
	Stuck-at 1			17602	2484	85.88
c880	Bit-flip	60,000	155/729	22780	16442	27.823
	Stuck-at 0			11703	8411	28.13
	Stuck-at 1			11077	8203	25.95
c1355	Bit-flip	60,000	67/1064	10604	8454	20.28
	Stuck-at 0			2930	2276	22.32
	Stuck-at 1			7674	6178	19.50
c1908	Bit-flip	100,000	242/1498	32962	21874	33.64
	Stuck-at 0			18765	12171	35.14
	Stuck-at 1			14197	9703	31.65

applying redundancy to the critical/sensitive nodes in the future work.

REFERENCES

[1] A. R. Khatri, A. Hayek, and J. Börcsök, *Applied Reconfigurable Computing*, vol. 9625 of *Lecture Notes in Computer Science*. Cham: Springer International Publishing, 2016.

[2] W. Xin, "Partitioning Triple Modular Redundancy for Single Event Upset Mitigation in FPGA," in *2010 International Conference on E-Product E-Service and E-Entertainment*, (Henan), pp. 1–4, IEEE, Nov 2010.

[3] M. Desogus, L. Sterpone, and D. M. Codinachs, "Validation of a tool for estimating the effects of soft-errors on modern SRAM-based FPGAs," in *2014 IEEE 20th International On-Line Testing Symposium (IOLTS)*, (Platja d'Aro, Girona, Spain), pp. 111–115, IEEE, Jul 2014.

[4] L. A. C. Benites and F. L. Kastensmidt, "Automated design flow for applying Triple Modular Redundancy (TMR) in complex digital circuits," in *2018 IEEE 19th Latin-American Test Symposium (LATS)*, pp. 1–4, IEEE, Mar 2018.

[5] H. Ben Fekih, A. Elhossini, and B. Juurlink, *Applied Reconfigurable Computing*, vol. 9040 of *Lecture Notes in Computer Science*. Cham: Springer International Publishing, 2015.

[6] A. R. Khatri, A. Hayek, and J. Börcsök, "Validation of the Proposed Fault Injection , Test and Hardness Analysis for Combinational Data-flow Verilog HDL Designs under the RASP-FIT Tool," in *2018 IEEE 16th Int. Conf. on Dependable, Autonomic & Secure Comp., 16th Int. Conf. on Pervasive Intelligence & Comp., 4th Int. Conf. on Big Data Intelligence & Comp., and 3rd Cyber Sci. & Tech. Cong.*, (Athens, Greece), pp. 544–551, IEEE Comput. Soc, 2018.

[7] G. dos Santos, E. Marques, L. d. B. Naviner, and J.-F. Naviner, "Using error tolerance of target application for efficient reliability improvement of digital circuits," *Microelectronics Reliability*, vol. 50, pp. 1219–1222, Sep 2010.

[8] A. R. Khatri, A. Hayek, and J. Börcsök, "RASP-TMR: An Automatic and Fast Synthesizable Verilog Code Generator Tool for the Implementation and Evaluation of TMR Approach," *International Journal of Advanced Computer Science and Applications*, vol. 9, no. 8, pp. 590–597, 2018.

[9] P. Balasubramanian, K. Prasad, and N. E. Mastorakis, "A Fault Tolerance Improved Majority Voter for TMR System Architectures," *WSEAS Transactions on Circuits and Systems*, vol. 15, pp. 108–122, 2016.

[10] S. Müller, T. Koal, M. Schölzel, and H. T. Vierhaus, "Timing for Virtual TMR in Logic Circuits," in *IEEE 20th InternationalOn-Line Testing Symposium (IOLTS)*, pp. 190–193, 2014.

[11] S. Di Carlo, G. Gambardella, P. Prinetto, D. Rolfo, P. Trotta, and A. Vallero, "A novel methodology to increase fault tolerance in autonomous FPGA-based systems," in *2014 IEEE 20th International On-Line Testing Symposium (IOLTS)*, (Girona, Spain), pp. 87–92, IEEE, Jul 2014.

[12] A. H. El-Maleh and K. A. K. Daud, "Simulation-Based Method for Synthesizing Soft Error Tolerant Combinational Circuits," *IEEE Transactions on Reliability*, vol. 64, pp. 935–948, Sep 2015.

[13] M. Raji, H. Pedram, and B. Ghavami, "Soft error rate estimation of combinational circuits based on vulnerability analysis," *IET Computers & Digital Techniques*, vol. 9, pp. 311–320, Nov 2015.

[14] P. Shivakumar, M. Kistler, S. Keckler, D. Burger, and L. Alvisi, "Modeling the effect of technology trends on the soft error rate of combinational logic," in *Proceedings International Conference on Dependable Systems and Networks*, pp. 389–398, IEEE Comput. Soc, 2002.

[15] P. 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, pp. 2957–2969, Oct 2004.

[16] S. Almukhaizim and Y. Makris, “Soft Error Mitigation Through Selective Addition of Functionally Redundant Wires,” *IEEE Transactions on Reliability*, vol. 57, pp. 23–31, Mar 2008.

[17] F. Kastensmidt and P. Rech, *FPGAs and Parallel Architectures for Aerospace Applications*. Cham: Springer International Publishing, 2016.

[18] H. Cai, K. Liu, L. A. de Barros Naviner, Y. Wang, M. Slimani, and J.-F. Naviner, “Efficient reliability evaluation methodologies for combinational circuits,” *Microelectronics Reliability*, vol. 64, pp. 19–25, Sep 2016.

[19] A. R. Khatri, A. Hayek, and J. Börcsök, “ATPG method with a hybrid compaction technique for combinational digital systems,” in *2016 SAI Computing Conference (SAI)*, (London, UK), pp. 924–930, IEEE, Jul 2016.

[20] A. R. Khatri, A. Hayek, and J. Börcsök, “RASP-FIT : A Fast and Automatic Fault Injection Tool for Code-Modification of FPGA Designs,” *International Journal of Advanced Computer Science and Applications*, vol. 9, no. 10, pp. 30–40, 2018.

[21] A. R. Khatri, A. Hayek, and J. Börcsök, “Validation of selecting SP-values for fault models under proposed RASP-FIT tool,” in *2017 First International Conference on Latest trends in Electrical Engineering and Computing Technologies (INTELLECT)*, (Karachi, Pakistan), pp. 1–7, IEEE, Nov 2017.