Development and Verification of Serial Fault Simulation for FPGA Designs using the Proposed RASP-FIT Tool

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

***Abstract*—Fault simulation is the critical approach for many applications such as fault detection & diagnostics, test set quality measurement, generation of test vectors, circuit testability, and many others along with the help of fault injection technique. The fault simulation approach is divided into many types. The most straightforward approach among them is a serial fault simulation. In the simulation process, the circuit under test is faulted, and a faulty copy is achieved by either using a simulator command technique or instrumentation technique. A fault simulator must examine the behaviour of specified target fault in design and classified as detected or undetected by the applied test patterns. To modify the original code is a very challenging and time-consuming task. Therefore, the RASP-FIT tool is developed, which alters the fault-free FPGA design, which is under investigation, at the Verilog HDL code level. It produces the copies of faulty design along with the top design file for several fault simulation methods. Using this tool, a serial fault simulation environment can easily be created with no much effort. In this work, a serial fault simulation method is verified and validated using the RASP-FIT tool for an ISCAS'85 benchmark design as an example.**

***Keywords*—FPGA design; fault injection; fault simulation; RASP-FIT tool; Verilog HDL**

I. Introduction

Fault simulation and fault injection approaches are the most widely used techniques for confirming the functionality of Register Transfer Level (RTL) design and provide an approach to check the quality of test-benches and test patterns [1]. In the design environment, fault simulation is also practised for validation of test quality [2], [3]. By definition, fault simulation is the technique used to simulate a design in the presence of faults.

In comparison with logic simulation, a fault simulation method has an additional complexity due to the modelled faults in the design and their behaviour during the simulation. When performing simulation for a design, the measurement of Central Processing Unit (CPU) computations is almost proportional to these three things, which are the size of the circuit, the number of test vectors applied, and the number of modelled faults injected in design. Fault simulation methods are separated into five main methods, namely, [4], [5], [6]:

- Serial fault simulation
- Parallel fault simulation
- Deductive fault simulation
- Concurrent fault simulation
- Differential fault simulation

The most straightforward approach among them is the serial fault simulation in which an individual simulation is performed at any one time. There are two primary components of serial fault simulation, i.e. fault-free designs and faulty designs (the design which is consisted of faults and is termed as faulty design). Firstly, the fault-free logic simulation is executed on the fault-free design to achieve the fault-free output responses. After that, faulty designs are simulated with faults and responses are also obtained. Both responses are saved and compared to determine whether an applied test pattern can identify a fault or not [4], [7], [6], [8].

In serial fault simulation, one fault is simulated at a time. To generate the faulty design, fault injection is first introduced, which alters the original circuit, and the circuit behaviour is evaluated in the presence of the fault. The faulty circuit is simulated to determine the inadequate responses for the currently activated fault for the given test patterns applied to a fault simulation. This process repeats until all faults in the fault list have been simulated [5], [4], [9]. In serial fault simulation, one fault is activated at a time, and test patterns are applied until the fault is detected or all test patterns have been applied. After that, another fault is selected and activated as a new fault, the circuit should be at an initial state, and then the new faulty design is simulated. Repeat this process until all faults are tested [5]. Serial fault simulation needs multiple simulations runs on a standard gate-level simulator using built-in simulator commands. In this method, the complex data structures are not required. There are certain advantages and disadvantages of serial fault simulation technique and are addressed in the sequel.

- Advantages:- A few advantages of serial fault simulation are described below:
  1) Easy to implement.
  2) Any true value simulator can be used.
  3) Less memory is required if the concepts of fault dropping, fault collapsing etc. are introduced in process.
- Disadvantages:- A few disadvantages of serial fault simulation are described below:
  1) Much repeated computation.
  2) Large CPU time required for very large scale integration design.
  3) Not feasible for large design with the large number of inputs.

In previous research, Khatri et al. proposed and developed a fault injection tool and named it RASP-FIT (RechnerArchitektur und SystemProgrammierung-Fault Injection Tool) tool [10], [11], [12], [13], [14], [15], [16]. This tool is used to instrument the Field Programmable Gate Array (FPGA) based designs. FPGA-based designs are written in Hardware Description Languages (HDL). During the last few decades, HDLs have been involved in promoting several methods and techniques regarding digital system testing. These methodologies reduce the technological passage among the tools and techniques used by design and test engineers. Using HDL, the design engineers can verify, validate and test the design at an early step at the code level [5]. In this work, the serial fault simulation is performed for FPGA-based designs at the code level. Serial fault simulation is executed at the code level of the designs. To generate the faulty copies of the original designs, the RASP-FIT tool is used, which modifies the design for different fault models. The RASP-FIT is designed to perform different functions. In this paper, serial fault simulation is carried out, which proved that the RASP-FIT tool can be used to develop any fault simulation schemes/applications.

The organisation of the paper is as follows: Section II describes the brief introduction about the RASP-FIT tool and an environment for serial fault simulation. Section III presents the usage of the RASP-FIT tool in fault simulation applications, and the evaluation of the result is shown. Results are discussed in the Section IV. In the end, Section V concludes the paper and presents some future directions.

II. The RASP-FIT Tool and Simulation Environment

The RASP-FIT tool is proposed and developed by Khatri et al. using Matlab graphical user interface development environment at the University of Kassel, Germany. The RASP-FIT is a fault injection tool, which is developed to perform fault injection, testing, hardness analysis for the FPGA-based designs at the code level. At the code level, the design and test engineers can perform testing and verification at an early stage of the development cycle. The main advantage of fault injection at the code level is to create the state of the art methods and also develop the new methods with little effort. More details about this tool can be found in [11], [12], [13], [15], [16].

A. RASP-FIT Verilog Code Modifier

The Verilog code modifier function under the RASP-FIT tool consists of approximately 563 lines of code in Matlab having 20 functions. The RASP-FIT is a tabbed based tool. Verilog code modifier is tabbed under fault injection analysis and is shown in Fig. 1. To modify the design, the user needs to apply three inputs for code modification and generates compilable faulty design. These inputs are:

1) A Synthesizable Verilog design file.
2) Type of fault model for injection in the design from a drop-down menu.
3) A number of faulty copies the user wants to generate.

By clicking on the *Generate* button, faulty modules are created along with the top file. In order to differentiate one

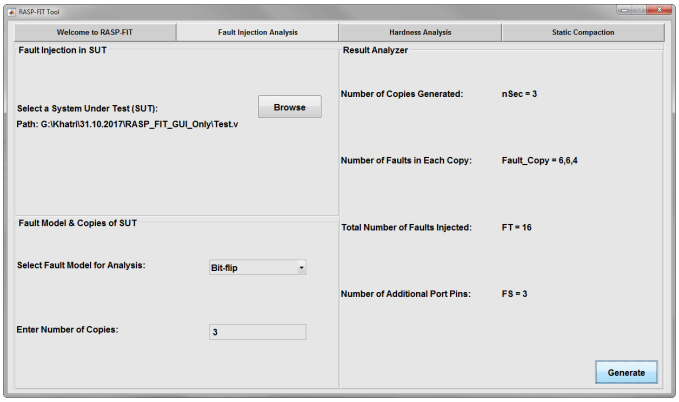


Fig. 1. Verilog HDL Code Modifier Tab under RASP-FIT Tool.

file from other files, the RASP-FIT tool saves the faulty modules under the names (moduleName_faultycopy1.v, moduleName_faultycopy2.v, and so on). The RASP-FIT also generates *Top.v* file which consists of the comparator logic, fault detection logic and digital logic Verilog code for storing the responses in the memory. These modified designs are now used for the fault simulation/emulation, digital testing and dependability analysis, with FPGA tools, without much effort. The development of this tool is presented in previous research [11], [12], [15], [13], [16], [17], [18]. In this paper, serial fault simulation is validated for ISCAS'85 benchmark designs.

Verilog HDL code describes the design at several abstraction levels, e.g. gate-level, data-flow, and behavioural levels. The way of modification of the code is different for each abstraction level, and also fault models are coded and modified the design at that abstraction level. There are two main components for serial fault simulations.

B. Fault-Free Design

The original FPGA design under investigation is called a fault-free module or golden module. It is a reference design for the comparison between the responses of faulty SUT and the fault-free design. Fault-free design is taken from the ISCAS'85 benchmark designs written in Verilog HDL. Fig. 2 (left) shows the original (fault-free) design code of `c17.v` benchmark design as an example.

C. Faulty Design

Fault-free designs are modified using fault injection technique and are called faulty design. The proposed RASP-FIT tool is used to create faulty designs for performing the serial fault simulation. In this work, FPGA-based designs written in Verilog HDL are considered. This tool injects various fault models in the design such as bit-flip, stuck at 1/0 fault models. Fig. 2 (right) shows the modified compilable code by the proposed tool.

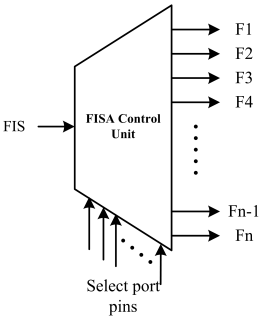
1) FISA Unit: A fault control unit is an important component for fault simulation applications. Khatri et al. proposed a demultiplexer based fault control unit [15] and called it the FISA unit. The term FISA unit stands for Fault Injection, Selection and Activation unit. It is designed for fault injection investigation to examine the injection of faults, as shown in

```
// Verilog Design
// c17 Benchmark Circuit ISCAS'85

module c17 (G1,G2,G3,G6,G7, G22,G23);

input G1,G2,G3,G6,G7;
output G22, G23;
wire G10,G11,G16,G19;

nand G_1 (G10, G1, G3);
nand G_2 (G11, G3, G6);
nand G_3 (G16, G2, G11);
nand G_4 (G19, G11, G7);
nand G_5 (G22, G10, G16);
nand G_6 (G23, G16, G19);
endmodule
```

```
module c17_1 (select,G1,G2,G3,G6,G7,G22_f1,
    G23_f1);
input G1,G2,G3,G6,G7;
output G22_f1, G23_f1;
wire G10,G11,G16,G19;
input select;
wire fis=1;
reg f0;
always @(select) begin
if (select == 1'd1) begin
f0=fis;end
else begin
f0=0;end
end

nand G_1 (G10,f0 ^ G1, G3);
nand G_2 (G11, G3, G6);
nand G_3 (G16, G2, G11);
nand G_4 (G19, G11, G7);
nand G_5 (G22_f1, G10, G16);
nand G_6 (G23_f1, G16, G19);
endmodule
```

Fig. 2. Fault-Free Design Code (Left) & Instrumented Compilable Code for Serial Fault Simulation (Right) by RASP-FIT.

Fig. 3. The FPGA design under test is written in Verilog HDL. Therefore, a fault control unit must be described in HDL code in the design. For that purpose, the function under RASP-FIT is included, which generates the code for the proposed FISA unit in each faulty copy. For large designs, as the number of injected faults increased, fault selection lines are also increased. De-multiplexer can be designed in Verilog HDL in various formats, e.g. using a case or if-else statements. De-multiplexer is a component which contains one input port (in this case Fault Injection Signal (FIS)), selecting one of many data-outputs, which is attached to the input port.




```
wire FIS = '1 '; //Declaration Part
input [1:0] select;
reg f0, f1, f 2, ... ;
always @ ( select ) begin
if ( select == 2'd0 ) begin
f0=FIS ; f1=0; f2=0; end
else if ( select == 2'd1) begin
f0=0; f1=FIS ; f2=0; end
else if ( select == 2'd2 ) begin
f0=0; f1=0; f2=FIS; end
else begin
f0=0; f1=0; f2=0; end
end
```

Fig. 3. Fault Selection and Activation Unit Schematic (Left), and Verilog Code (Right).

D. Experimental Set-up for Serial Fault Simulation

In serial fault simulation approach, one fault is inserted into the circuit at a time, and the fault effect is observed. The number of simulations is equal to the total number of faults plus one simulation (the initial simulation of the golden circuit) [9]. The block *c17.v* (original) is the fault-free circuit and considered as an example from ISCAS'85 designs. These FPGA designs are written in Verilog HDL code at the gate-abstraction level. The other blocks *C17(f1)* to *C17(fn)* are faulty copy of the *c17.v* original design with faults *f1* through *fn* permanently instrumented as shown in Fig. 4. This method is an alternative technique for serial fault simulation to reduce CPU computation time.

The schematic diagram for the example *c17.v* is shown in Fig. 5. Fig. 6 shows the various locations of the design where faults can be injected to develop faulty designs to perform serial fault simulations. The mark (x) points out the location of the faults. Each (x) represents the fault models. Faulty copies are modified using instrumentation technique, and faults are injected in the design, for example, stuck-at 1/0 and bit-flip fault models.

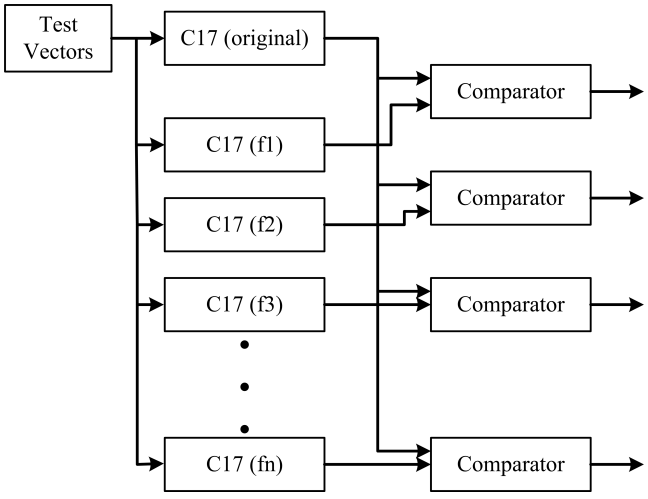


Fig. 4. Experimental Setup for Performing Serial Fault Simulation.

1) Fault Dictionary: When the fault simulation is performed by applying test vectors, and the result is obtained by comparing it to the responses of golden design. These results are organised in a simple two-column table where

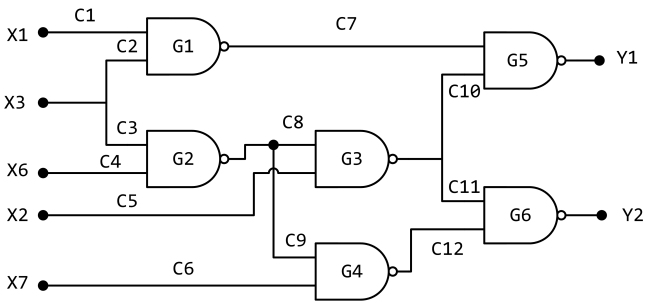


Fig. 5. Example of Design from ISCAS'85 Benchmark Circuits (Original Design).

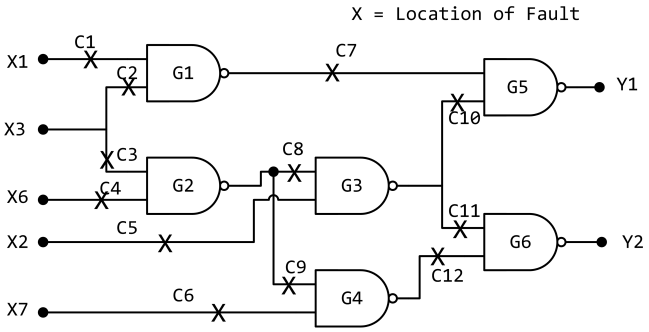


Fig. 6. Example of Design from ISCAS'85 Benchmark Circuits (with Faulty Locations).

every column correspond to circuit's faults and test vectors that detect them. This arrangement of data is called a fault dictionary. One fault can be detected by many test vectors and is called a detectable fault, whereas some faults cannot be detected by any of the test vectors, which is called undetectable faults.

Fault simulation is performed on fault-free and faulty designs. The responses are gathered in the presence of faults and presented in the tabular form. When a fault is injected in the design, then those test vectors which can detect that fault are placed in the fault dictionary [5]. For the large design with many input ports, this is not possible to simulate the design with all possible combination of inputs. Therefore, serial fault simulation is a simple approach which can be applied to the smaller designs with few faults and few test vectors. Table I depicts the example of fault dictionary. In the table, three faults {f1, f2, f3} are used during the fault simulation method, and test vectors (2,5,9) detect the f1 fault, test vectors (5,12,6) find f2 whereas f3 is not undetected with any test vector. This information helps design and test engineers to calculate Fault Coverage (FC).

TABLE I. EXAMPLE OF FAULT DICTIONARY

Fault No.	Test Vectors
f1	2,5,9
f2	5,12,6
f3	-

III. EXPERIMENTAL EVALUATION USING THE RASP-FIT TOOL

Serial fault simulation is a straight forward technique. It requires the fault-free circuit and the faulty circuits. The RASP-FIT tool is a fault injection tool which modifies the fault-free design and generates faulty modules. Along with the fault-free design, the user also needs two more inputs. The first one is a type of fault model used in serial fault simulation. The RASP-FIT can inject three fault models, i.e. bit-flip, stuck-at 0 and stuck-at 1 models separately and modifies the code accordingly. The second input is the number of copies required by the user. The RASP-FIT tool can evenly distribute the number of faults in different copies of the design, as shown in Fig. 7 (Step 1). In the second step, all these files are used to create a project using the Xilinx ISE tools, and Modelsim or Xilinx ISIM tools are utilised for simulation. After the simulation, results are stored in a text file. This text file is applied as an input to the Matlab script, which develops the fault dictionary and calculates the fault coverage for the design using Eq. 1. It is defined as the ratio of fault detected to the total fault injected.

$$FC = \frac{F_D}{F_T} \times 100\% \tag{1}$$

where F_D is the number of detected faults during the serial fault simulation and F_T shows the total faults injected during the experiment. In the last step, the test bench must be added to the project to perform the simulation. For large designs, serial fault simulation technique is not a feasible solution, but the proposed RASP-FIT tool helps design and test engineers to perform serial fault simulation.

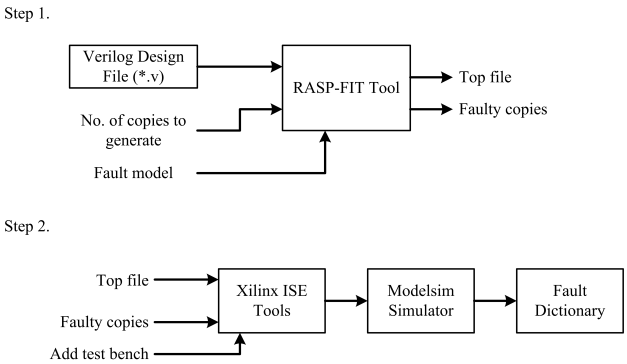


Fig. 7. Experimental Set-up using the RASP-FIT and FPGA Tool.

A. Top File Structure

The top file is generated under the RASP-FIT tool, which is used to perform serial fault simulation. The top file contains various elements, which are used to perform serial fault simulation, listed below:

1) Instantiations (golden and faulty copies).
2) Comparator logic.
3) Memory for storing responses.

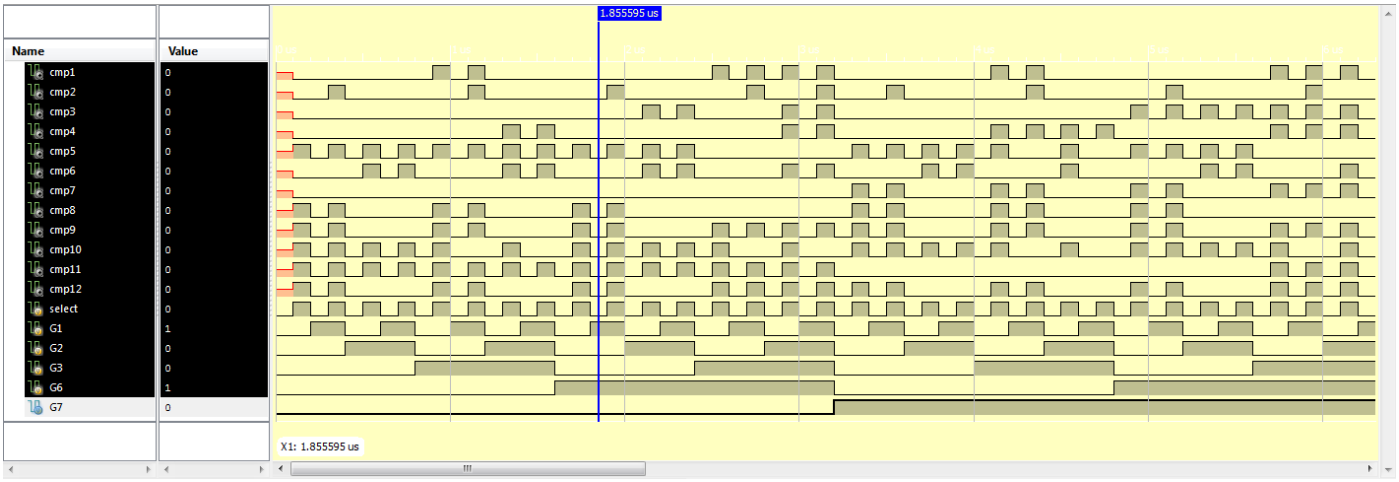


Fig. 8. Waveform for Serial Fault Simulation of *c17.v* Design.

IV. Result and Discussion

In this work, the RASP-FIT tool is used to perform and verify serial fault simulation applications. The objective is to validate the claim that, using RASP-FIT tool, the user can perform various fault simulation methodologies. An example from ISCAS'85 benchmark designs is considered to explain the serial fault simulation operation. As explained earlier, in serial fault simulation, one fault is injected at a time and input is applied to see whether the fault is detected or undetected. To save the time of simulation, authors use the experimental set-up as shown in Fig. 4. One fault is injected into the each copy of the design. The schematic of the `c17.v` design is shown in Fig. 5 and the Verilog code is illustrated in Fig. 2 along with the faulty copy containing one fault. The fault location is also marked on Fig. 6.

The simulation is performed using the Xilinx ISIM or Modelsim tool and waveform is shown in Fig. 8. Fault dictionary is constructed and shown in Table II. This example design is a simple design which consists of only 12 faults, and a total of 32 test vectors are applied as input to the design. Therefore, it is not a feasible idea for large designs having thousands of faults and many input ports to apply all possible combination of input. As every fault is detected at least one time hence fault coverage is 100%.

TABLE II. Fault dictionary for the example under simulation

Fault No.	Test Vectors	Detected (X) / Undetected (-)
f1	4,5,12,13,14,15,20,21,28,29,30,31	X
f2	1,5,9,13,15,17,21,25,29,31	X
f3	10,11,14,15,24,...,31	X
f4	6,7,14,15,20,21,23,28,...,31	X
f5	0,...,11,16,...,20,22,24,...,27	X
f6	2,3,6,7,10,11,14,15,19,22,26,27,30,31	X
f7	16,17,20,21,24,28,29,30,31	X
f8	0,1,4,5,8,9,16,17,20,21,24,25	X
f9	0,1,4,5,8,9,12,...,17,20,21,24,25,28,...	X
f10	0,...,4,6,8,...,12,14,16,...,20,22,24,28,30	X
f11	0,...,15,28,29,30,31	X
f12	0,1,4,5,8,9,12,...,17,20,21,24,25,29,...,31	X
F_T= 12	Total TV = 32	FC = 100%

V. Conclusion

Fault simulation technique assists designers and test engineers in several applications such as design's verification, test patterns generations and many other applications. The RASP-FIT is a simple, automatic and user-friendly fault injection tool, which works at the code level of the designs at several abstraction levels. This tool can inject faults in the whole design, and produce the compilable code. This tool helps the design and test engineers to perform various fault simulation applications. In this work, a serial fault simulation method is verified and validated. It is shown that, with the help of this tool, serial fault simulation can easily be performed. In future, parallel, deductive, concurrent and differential fault simulation approaches will be implemented and validated using the RASP-FIT tool.

References

[1] N. Bombieri, F. Fummi, and V. Guarnieri, "FAST-GP: An RTL functional verification framework based on fault simulation on GP-GPUs," in *2012 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 562–565, IEEE, Mar 2012.

[2] A. Parreira, J. P. Teixeira, and M. Santos, "A novel approach to fpga-based hardware fault modeling and simulation," *Design and Diagnostics of Electronic Circuits and Syst. Workshop*, pp. 17–24, 2003.

[3] S. Misera and R. Urban, "Fault Simulation and Fault Injection Technology Based on SystemC," in *Design and Test Technology for Dependable Systems-on-Chip*, pp. 268–293, 2011.

[4] L.-T. Wang, Cheng-Wen Wu, and Xiaoqing Wen, *VLSI TEST PRINCIPLES AND ARCHITECTURES*. 2006.

[5] Z. Navabi, *Digital System Test and Testable Design*. Boston, MA: Springer US, 2011.

[6] M. L. Bushnell and V. D. Agrawal, *Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits*. Kluwer Academic Publishers, 2002.

[7] F. Corno, G. Cumani, M. Sonza Reorda, and G. Squillero, "Effective techniques for high-level ATPG," in *Proceedings 10th Asian Test Symposium*, pp. 225–230, IEEE, 2001.

[8] S. Devadze, *Fault Simulation of Digital Systems*. PhD thesis, TALLINN UNIVERSITY OF TECHNOLOGY, 2009.

[9] Ian A. Grout, "Test Pattern Generation and Fault Simulation," in *Integrated Circuit Test Engineering*, ch. Chapter 10, pp. 235–255, London: Springer-Verlag, 2006.

[10] A. R. Khatri, M. Milde, A. Hayek, and J. Börcsök, "Instrumentation Technique for FPGA based Fault Injection Tool," in *5th International Conference on Design and Product Development (ICDPD '14)*, (Istanbul, Turkey), pp. 68–74, 2014.

[11] 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.

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

[13] A. R. Khatri, A. Hayek, and J. Börscök, "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 Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/PiCom/DataCom/CyberSciTech)*, (Athens, Greece), pp. 544–551, IEEE, Aug 2018.

[14] 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.

[15] 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.

[16] A. R. Khatri, A. Hayek, and J. Börcsök, "Fault Injection and Test Approach for Behavioural Verilog Designs using the Proposed RASP-FIT Tool," *International Journal of Advanced Computer Science and Applications*, vol. 10, no. 4, pp. 57–63, 2019.

[17] A. R. Khatri, A. Hayek, and J. Börcsök, "Validation of the Proposed Hardness Analysis Technique for FPGA Designs to Improve Reliability and Fault-Tolerance," *International Journal of Advanced Computer Science and Applications*, vol. 9, no. 12, pp. 1–8, 2018.

[18] A. R. Khatri, "A Technical Guide for the RASP-FIT Tool," *International Journal of Advanced Computer Science and Applications*, vol. 10, no. 12, 2019.