Study and Comparison of Various FPGA-based Fault Injection Tools with RASP-FIT Tool for HDL Designs

Abdul Rafay Khatri

***Abstract* – Fault injection technique is the most popular technique for characterising the dependability parameter of Very Large Scale Integrated (VLSI) systems and designs. Due to technology scaling, Field Programmable Gate Array (FPGA) systems are also prone to error and failure; hence fault injection tools play a vital role to test and evaluate dependability parameters. These tools are categorised into two techniques; reconfiguration and instrumentation-based techniques for FPGA fault injection tools. The primary focus is put on instrumentation-based fault injection tools and techniques in this paper. In the instrumentation technique, a specific circuit is appended to the original design to carry out fault injection analysis. This paper presents the comparison between various fault injection tools based on methodology, fault models consideration, fault injection control unit and functions they perform.**

Index Terms – Dependability analysis, Instrumentation, Fault injection, FPGA, Reliability, Fault tolerance.

I. INTRODUCTION

FAULT Injection (FI) is one of the most well-known techniques which is used in the evaluation of faults and to check the capability of design in tolerating failures. The fault injection technique performs different functions such as detection of sensitive areas of design, validation of the design for evaluating reliability, forecasting the output in the presence of a fault. Broadly, fault injection techniques can be arranged into four groups, i.e. hardware, software, simulation, and emulation-based techniques. Fault injection tools involving Field Programmable Gate Array (FPGA) are classified into two types, i.e. the simulation-based and emulation-based fault injection tools
[1].

Nowadays, FPGA-based designs are widely used in various safety-critical systems and space applications. The SRAM-based FPGA designs cover nearly 60% of the application because of multiple advantages it offers. To increase the capacity, the size of components is reduced continuously, making the SRAM-based FPGA devices and applications faster, as well as; they become prone to errors and failures. Therefore, FPGA-based devices must be tested and checked for the dependability analysis. In this case, FPGA-based fault injection techniques and tools play a vital role to verify and validate the System Under Test (SUT). In general, the simulation-based method allows greater flexibility (i.e. observability and controllability) whereas, the emulation technique provides the execution of experiments in real time on FPGA platforms [2]. There are numerous different causes for requiring FPGA in FI techniques, for example, for simulation, designs models are available. Also, for emulation on the FPGA platform, fast emulation can be achieved. By the use of full & partial reconfiguration techniques, area overhead issues are resolved. Therefore, FPGA-based FI tools are classified into two classes, i.e. instrumentation-based and reconfiguration-based fault injection tools due to advantages of the FPGA mentioned above (i.e. more area capacity and reconfiguration technique), respectively.

In the development of FI tools, a significant problem is the definition of a FI mechanism also known as FI control unit. This unit is designed by keeping some parameters in mind, such as area overhead, the speed of injection (i.e. time cost), selection and injection of particular faults [3]. This unit consists of simple or complicated circuitry.

The system reliability is one of the major concern in today's electronic applications. It is the most important characteristic, which measures the quality of systems [4]. Fault injection techniques can also evaluate the reliability of the SUT. The term reliability is described as "It is a probability, which shows the correct functioning of the design even in the presence of faults" [5], [6]. Reliability is the measure of how good a system is and how often it goes down [7]. There are specific techniques which are used more widely for the improvement of reliability such as, triple modular redundancy, hardened memory cell level, multiple redundancies with voting, and error detection and correction coding.

In this paper, two categories of FPGA-based fault injection tools are described along with the FI environment of general tools and tools for the FPGA. Modern FPGAs are enriched with the capacity and fast (full & partial) reconfiguration capabilities. Therefore, various FI techniques and tools are studied and compared according to the types of the method employed, fault model used, and functions they performed for FPGA-based designs. The proposed RASP-FIT tool (detail in Section IV) is an instrumentation-based FI tool and compared with the tools available.

This paper is structured as follows: Section II explains the general environment of FI tools. The environment specifically for FPGA-based FI tools are presented in Section III, and it also includes the various FI tools in detail. Section IV describes the proposed fault injection tool. The comparison between the tools is presented tabularly in Section V. In the end, the conclusion of the paper is described in Section VI.

Abdul Rafay Khatri is with Department of Computer Architecture and System Programming, University of Kassel, Kassel, Germany. Email: arkhatri@student.uni-kassel.de. Manuscript received on May 06, 2019, revised on Aug 09, 2019 and accepted on Aug 23, 2019.

 ISSN – 1999-4974

II. GENERAL FAULT INJECTION SYSTEM ENVIRONMENT

Fault injection techniques can be applied to both hardware and software systems to measure fault-tolerance and robustness. The environment for hardware and software systems is different. In a hardware system, faults can be injected on a pin level or an internal level of chips; also faults can be inserted into the simulation of the system prototype. For software, faults are injected into the software program during compile time or run-time in the instructions in Central Processing Unit (CPU) registers to networks. Therefore, the fault injection environment is designed according to the system characterization. A minimal general fault injection system is composed of three fundamental modules [8], [9], [10], as shown in Figure 1.

1) Fault List Manager (FLM)
2) Fault Injection Manager (FIM)
3) Result Analyser (RA)

A. Fault List Manager (FLM)

The FLM is the basic block in the fault injection environment of any tool, which is responsible for the generation of the Fault-List and Fault-Type injected into the various part of the SUT. Fault locations are assumed to be equally probable for designs. This module generates faults at all possible locations in the target system. Furthermore, it sends information to the next critical module FIM.

B. Fault Injection Manager (FIM)

The most critical module in the FI environment is the fault injection manager. The complexity of this module is proportional to the size of the target system. Functions of this module are selecting a particular fault, activating the fault and observing its resulting behaviour on the target system [8].

C. Result Analyser

This module can be designed to perform various functions. The primary features of the module are collecting and analysing the results/responses from the whole experiment and producing the statistical reports accordingly. For example, in fault injection testing approach, static compaction technique is part of this module, which generates compact test vectors, Fault Coverage (FC) and so on.

III. ENVIRONMENT OF FAULT INJECTION TOOL FOR FPGAS

FPGA-based fault injection tools have benefits of both physical and simulation-based method, such as speed and flexibility. The design & development flow for FPGA systems consist of many steps, where design modification is possible for the fault injection analysis. The significant points of modification are in the design's code, gate-level net-list, place & route file and the bit-stream file [11]. Figure 2 shows the various points of modification of the FPGA designs [11]. These tools are classified into two techniques, i.e. reconfiguration-based and instrumentation-based FI tools. Many state-of-the-art FI tools have been proposed and presented in the literature for FPGA-based designs, which insert faults at different stages of development flow for assessing design characteristics for FPGA-based designs.

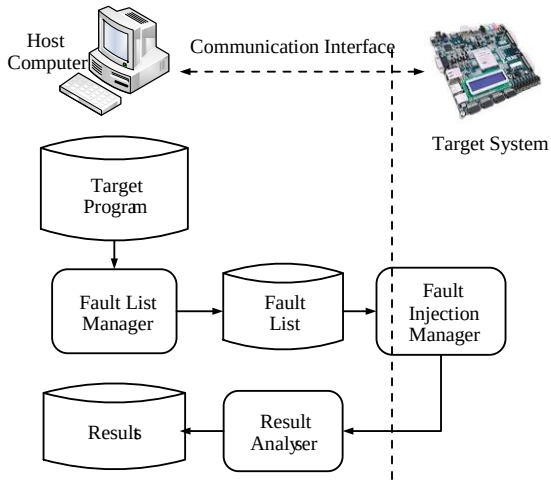

Fig. 1 General fault injection environment for FI tools [3]

A. Reconfiguration-based FI Tools

Reconfiguration or partial reconfiguration is the technique in which configuration memory of FPGA is modified or changed with some other logic to introduce defects in the SUT. In this technique, as there is no other circuit, so there is no area overhead problem, whereas, reconfiguration/partial reconfiguration has time (speed) issue. The second drawback is that these techniques can only be used with the FPGAs having these reconfiguration facilities. Mostly new FPGAs can be reconfigured partially or entirely using some global signals [13]. Few reconfiguration based FI tools are described in the sequel.

1) Flipper

The flipper is the tool, use to determine Single Event Upsets (SEUs) effect in SRAM-based FPGA systems using FI technique. It was developed with the help of the European space agency. Flipper injects Single Event Upsets (SEU) faults in the configuration memory of the target system using the partial reconfiguration. Flipper consists of two parts, i.e. hardware platform and software application. The hardware platform contains a flexible FPGA board for SEU analysis, and the software parts run on the host computer, which was developed for the flipper tool for fault injection experiments [14], [15].

The flipper is a reconfiguration based fault injection tool for designs implemented on FPGA. This tool requires two boards for fault injection analysis. The first one is Xilinx Virtex 2 Pro motherboard test fixture, which is used for radiation tests and the second is a piggy-back board (SRAM-FPGA for implementing Device Under Test (DUT)) [16]. Probabilistic model is involved to determine design sensitivity instead of testing all configuration bits. Fault injection process modifies frames and uses active partial reconfiguration. Flipper can inject single as well as multiple bit upsets in the DUT.

ISSN – 1999-4974

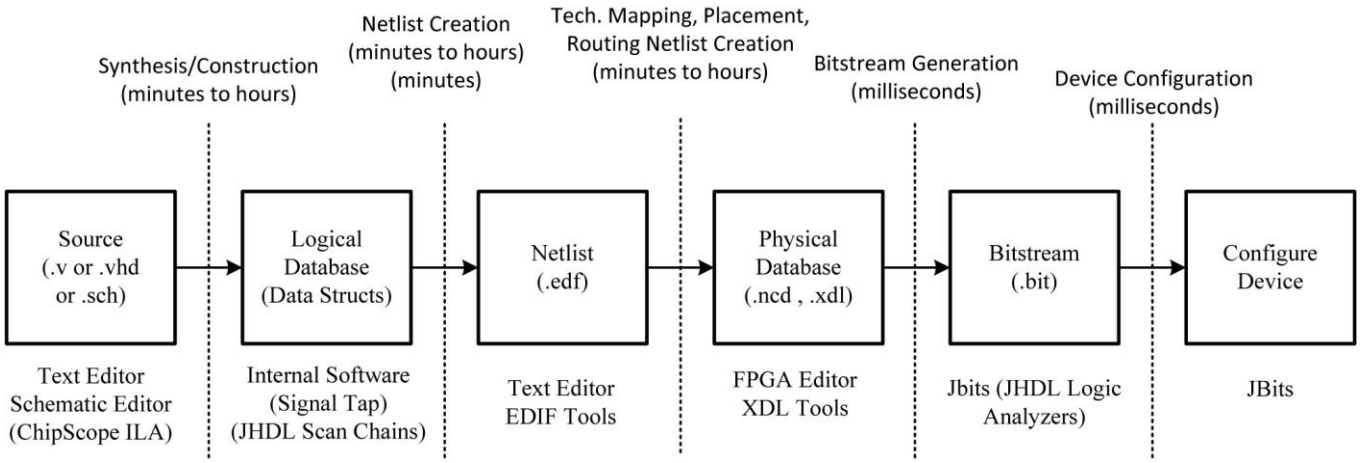

Fig. 2 The FPGA development flow showing various points of modification [12]

A graphical user interface is created for performing the software part of the host PC. The software application can be used for the following functions, such as the target design for fault injection, fault type (bit-flip fault model (SEUs)), test mode and clock rate. Input and output data can be gathered from the simulation tool (i.e. *ModelSim*) at each clock edge. Flipper tool is used to simulate the effect of ion-radiation in the DUT. Also, it measures the robustness and sensitivity of the design [14].

2) FT-UNSHADES/-2:

It stands for Fault Tolerant UNiversity of Sevilla HArdware DEbugging tolerant System (FT-UNSHADES) is a software/hardware platform. This tool detects and analyses the fault tolerance capability of the SUT [17]. New and updated version of this tool, which uses partial reconfiguration of Xilinx Virtex technology and has an improved user interface is called FT-UNSHADES2. This tool deals with designs written in VHDL.

FT-UNSHADES hardware framework requires a golden model (Module Under Test), faulty MUT, comparator, test shell and input stimuli. In this framework, a fully controlled test environment is designed. The golden and faulty modules are implemented on the SRAM-FPGA on the Xilinx Virtex-II FPGA board. The golden module is protected from the radiation, whereas, another module is exposed to the radiations. These radiations produce the SEUs (bit-flip) in the design. The responses of both SUTs are compared to detect faults and evaluate reliability and fault tolerance of the design [18].

B. Instrumentation-based FI Tools

In the instrumentation-based technique, the additional circuitry for fault injection analysis is added to the target system, and it is called 'saboteur'. Saboteur can be defined as a component which remains inactive during its normal operation but once it is activated, introduces faults in the target system during the fault injection process. Saboteur can be constructed from a simple gate or some complex circuitry. The advantage of practising this technique is that it does not have time constraints. The foremost shortcoming of this technique is large area-overhead, which is not an issue with the new FPGAs, as they have a larger capacity to implement the large designs [13].

The design and development cycle for the FPGA-based designs consists of several steps where this instrumentation technique can be applied, e.g. in net-list generated after the synthesis process, bit-stream file, and at the HDL design code. Instrumentation-based fault injection tools are developed in the last decades are considered for the study in this paper.

1) NETFI/-2

NET-list Fault Injection (NETFI) provides an automated way of fault injection in synthesizable net-list at Register Transfer Level (RTL) of designs and systems, which are implemented on the FPGA. This tool can be worked on designs written in VHDL and Verilog HDLs. NETFI covers most of the critical memory cells of a design within one clock cycle for the fault injection analysis. In this method, a large net-list under test is decomposed into several smaller subnet-list to avoid the problem of large area overhead. To verify and validate the tool operation, NETFI is used in estimating the soft error rate of a DUT during execution of a benchmark program. NETFI-2 is an extended version of NETFI, which is faster than the previous version NETFI. Target system is instantiated on the FPGA and the FI campaign is carried out using NETFI-2. It minimizes the area overhead. The controller of the tool is based on MicroBlaze microprocessor, which provides an efficient and flexible controller. As it is based on soft-core processor so it is conveniently programmed in a software [19], [20], [21].

C. MODNET

In the NETFI, the generated net-list after the synthesis process is applied as an input to the software tool, named MODNET (MODify NET-list), developed to automate the instrumentation process. The output of this tool is a modified net-list with a large number of additional input signals. These signals are used to retrieve and inject SEUs and SETs in all logic designs and memory cells [19],
[20], [21], [2].

 ISSN – 1999-4974

1) FITO

Fault Injection TOol (FITO) instruments/modifies Verilog HDL code for FPGA-based designs. FITO supports fault injection for Verilog HDL at different abstraction levels, e.g. RTL and gate-level [22].

2) FITO contains the following main parts,

a) Source Code Modifier & FLM:- Verilog file is first input to this part of the tool which generates fault and time list, along with the instrumented synthesizable source code.
b) Fault Injection Manager:- FIM is a crucial part and is implemented on the FPGA. This part performs real-time fault injection and generates trace data for both fault-free and faulty systems.
c) Result Analyser:- This part of the tool is also developed on a host computer, which takes the trace data as input and performs fault calculations.

This tool performs fault injection experiments in various abstraction levels as mentioned before. In gate-level models, types of faults introduced are permanent and transient faults. It injects stuck-at faults in the code. The controlling of fault injection and activation is done by an additional signal *Fault Injection Signal* (FIS). For stuck-at 1, an OR gate is used having one FIS input, whereas, an AND gate is used an inverted FIS signal for stuck-at 0 fault models implementations. For the SEU fault model, an XOR gate is used with a variety of approaches as described in [22].

3) *FuSE:*

FuSE stands for Fault injection Using Simulation/Emulation. SEmulation is performed by SEmulator® engine, which is a hardware accelerator for HDL simulations. This tool performs both the simulation and emulation of the SUT. More detail can be found in [23], [24]. FuSE tool can inject faults in an HDL code during simulation and also inject faults in the synthesis net-list downloaded into FPGA.

One of the core components of this engine is HPE_Desk, which provides a user-friendly interface for the simulation & emulation processes. FuSE framework is used as a stand-alone VHDL-based fault injection tool under the usage of SEmulator® engine. FuSE considers stuck-at 1/0 fault model along with bit-flip, bridging or delay faults for single or multiple signals. The FuSE fault injection tool accelerates the process of fault simulation/emulation. Different fault models are used using saboteur injection, and results are observed.

4) *Direct Fault Injection:*

Direct Fault Injection (DFI) is an emulation based tool for FPGA designs. It can be used for both VHDL and Verilog designs. DFI is a tool used to insert SEU in flip-flops of the processor design within a single clock cycle. The fault injection control unit consists of a multiplexer with a finite state machine to inject and activate faults. Large sensitive areas in the processor are mostly caches and registers in the arithmetic logic unit, etc. The disadvantage of using DFI is that it can be used for processors, for which HDL code must be available. This tool deals with VHDL code in particular [19].

In this tool, the consequences of SEUs are emulated into the memory of processors. Faults are injected by adding saboteur and implemented on the FPGA with the original design. ASTERICS (Advanced System for the Test under Radiation of Integrated Circuits and Systems) is a platform which consists of previously developed THESIC+Platform by TIMA labs. The architecture of ASTERICS includes two FPGAs. The first FPGA is a COM processor (Power PC) and the second FPGA is called chip-set. The first FPGA handles the communication between the host computer and this platform. Also, it includes programmable watchdog timers which check errors in the DUT. The chip-set FPGA contains the DUT and the interface between the design and the platform. This connection is used to verify the operation of the DFI tool on the benchmark, which consists of the LEON processor written in VHDL.

5) *FIFA*

FIFA stands for Fault Injection and Fault masking Analysis (FIFA) approach. It is a hardware IP, designed to estimate the robustness of digital circuits by fault injection technique. It injects and observes the response of single or multiple faults. This tool injects faults using the FPGA at the RTL level [25]. Consider the digital circuit under analysis is 'OP'. In this platform, two copies of 'OP' are considered: fault-free (OP-REF) and faulty (OP-FAULTY). There are two steps for analysis of fault. At first, the fault is injected, and the response is gathered, and later compared with the output of the OP-REF from the same input provided to both. If any mismatch is not observed for a particular fault, then it can be said that the fault is masked and the circuit is robust to the fault. In this way, the robustness of digital circuits is evaluated [26].

In this tool, fault injection mechanism is based on saboteurs, which consist of a multiplexer with an XOR gate with error signal which alters the value of the connected input, whereas error signal is ANDED with two signals for a select signal of a multiplexer. In general, the bit-flip fault model is practised in this tool.

6) *FIDYCO*

FIDYCO stands for Flexible on-chip fault Injector for run-time Dependability validation with targetspecific COmmand language. It is a fault injection tool for FPGA-based designs. In this tool, there are two main parts that fault injector and the target system. Both parts are implemented on the FPGA. Target systems are written in the VHDL language. This tool is available as an open platform and flexible platform in which every type of component can be tested [27].

FIDYCO consists of DUT and Golden Node (GN), both implemented on the FPGA. GN is fault-free, whereas, fault injector unit injects faults in the DUT and responses are gathered and compared by the result analyser. The fault injector injects faults in a controlled manner, i.e. location and duration. The fault injection manager is used to select faults

ISSN – 1999-4974

to be inserted in the DUT, hence this unit is a very complex and intelligent. There are two modes of operation of the tool, automatic and interactive mode.

7) *DBIT*

The primary purpose of DBIT is independent the process of verification for FPGA-based designs written in VHDL language [28]. DBIT supports the operation of fault profiling and performs fault injection at the coding step of designing. VHDL file is applied as an input to the tool, which is examined for fault injection. The user selects the fault model and targets a line of code for modification of the particular design under test. Next step performs fault profiling, and then a result analyser collects the results and produces a report. The location of faults in the VHDL code of the target system where faults are injected as a mutant. Some locations are given in the sequel,

- signal/variable names;
- constants;
- operators ;
- assignments;
- conditional statements;

This tool is developed for the sole purpose of Independent Verification and Validation (IV & V) of the FPGA designs. It performs fault profiling, fault injection procedures, and the result analysis.

IV. RASP-FIT: PROPOSED FAULT INJECTION TOOL

RASP-FIT stands for "Rechner Architektur und System Programmierung-Fault Injection Tool". The first part of its name is the German name of the department. The RASP-FIT is based on an instrumentation technique and developed in Matlab. FPGA-based designs are described in hardware description languages, mainly Verilog and VHDL. This tool is specially developed for Verilog designs. These Verilog designs can be expressed at different abstraction levels, e.g. gate-level designs, data-flow designs and behavioural level designs [29], [30].

Fig. 3 GUI of the proposed tabbed-based FI tool (RASP-FIT)

FISA Unit: The FISA unit is a fault control unit which is based on demultiplexer logic. 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 Figure 5. The FIS signal has a logic value '1'. The function of the demultiplexer is to route the value of

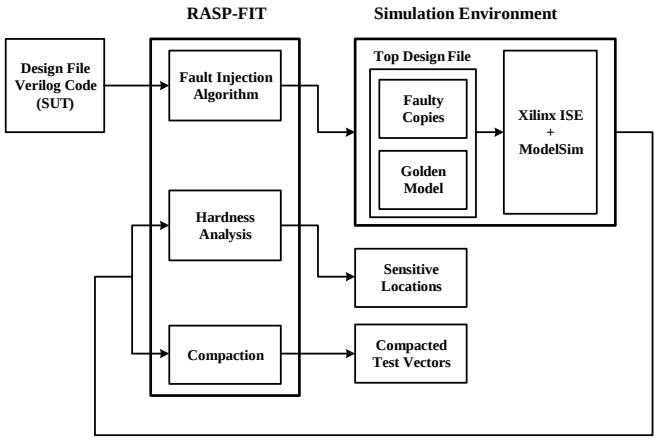

Fig. 4 RASP-FIT and simulation environment [11]

The graphical user interface for the tool is also developed in Matlab. Figure 3 shows the screen-shot of the GUI of the tabbed-based tool, where each tab describes some functionality of the RASP-FIT. One of the significant advantages of this tool is that it is technology independent; one can use it on any FPGA from any vendor as it works on the Verilog code. Secondly, it can test the design at the code level, and the user can obtain compact test vectors and fault coverage. Also, it helps to improve fault tolerance capability and reliability of the FPGA-based design [31], [11], [32].

The RASP-FIT tool takes a synthesizable Verilog design file as an input, and the Verilog code modifier algorithm (aka Fault Injection Algorithm) modifies the design by introducing different fault models. These faults remain inactive until the user activates them by giving signal through Fault Injection, Selection, and Activation (FISA) unit. The methodology along with the whole fault injection environment of RASP-FIT tool is shown in Figure 4.

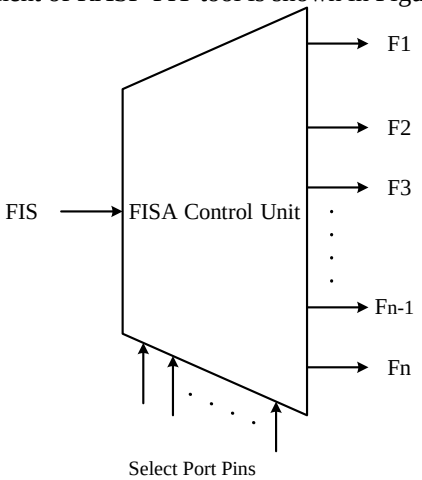

Fig. 5 Proposed DEMUX fault injection model (FISA Control unit) [31]

The FIS signal to the line numbered by select port pins. The fault selection input is activated by test-bench, and it is applied to select and enable all faults sequentially in the SUT. The number of Fault Selection (*FS*) input lines are calculated according to (1),

ISSN – 1999-4974

Gate abstraction level *// Different fault models*	Data−flow abstraction level *// Different fault models*	Behavioural abstraction level *// Different fault models*
module faultModels_GL (a ,b, fi_bf , fi_sa1 , fi_sa0 , c) **input** a ,b, fi_bf , fi_sa1 , fi_sa0 ; **output** c ; **and** and_1 (c , fi_bf ^ a ,b) ; *//Bit−f l i p* **and** and_2 (c , fi_sa1 | a ,b) ; *//Stuck−at 1* **and** and_3 (c ,~ fi_sa0 & a ,b) ; *//Stuck−at 0* **endmodule**	**module** faultModels_DF (a ,b, fi_bf , fi_sa1 , fi_sa0 , c) **input** a ,b, fi_bf , fi_sa1 , fi_sa0 ; **output** c ; **assign** c=((fi_bf ^ a) & b) ; *//Bit−f l i p* **assign** c=((fi_sa1| a) & b) ; *//Stuck−at 1* **assign** c=((~fi_sa0 & a) & b) ; *//Stuck−at 0* **endmodule**	**module** faultModels_behave (a , b, fi_bf , fi_sa1 , fi_sa0 , c) **input** a ,b, fi_bf , fi_sa1 , fi_sa0 ; **output** c ; *//always block blocking and non−blocking* c=((fi_bf ^ a) & b) ; *//Bit−f l i p* c<=((fi_sa1 | a) & b) ; *// Stuck−at 1* **endmodule**

$$FS = [\log_2(N_{copy})] \quad (1)$$

where FS is the number of fault selection port pins and N_{copy} is the number of faults injected per copy, respectively.

The RASP-FIT tool deals with various fault models (for example, bit-flip, stuck-at 1 & 0) for the FI analysis of SUT. It introduces faults in every permissible location in the target system. For these models, XOR, OR and AND gates with inverter are utilized as shown in the code. The RASP-FIT tool performs the following functions at this stage of development,

- Instrumentation of Verilog code (at all abstraction levels)
- Automatic Test Pattern Generation (ATPG) method [31], [30]
- Sensitivity analysis under hardness [11]

V. COMPARISON BETWEEN THE PROPOSED AND STATE-OF-THE-ART TOOLS

Various and fantastic fault injection tools are presented in the last couple of decades in the literature. In this study, FPGA-based fault injection tools are focused on. All tools are useful and designed for specific purposes. Most of the FI tools at the code level of design are developed for VHDL designs, whereas very few fault injection tools are produced for both, i.e. Verilog and VHDL.

Some tools provide more substantial area overhead during the fault injection analysis of the SUT in comparison of other tools, whereas other devices, might increase the number of input port pin in an unexpected or unrealistic count. If the tools require a large number of pins to select and activate individual fault, then it is not a feasible idea for a large design with thousands of failures. FPGA-based fault injection tools come in the category of emulation based technique. In this paper, more focused is put on the instrumentation-based FI tools and techniques.

In this approach, additional circuitry is added to the original design for fault injection analysis, e.g. fault models and the fault control unit are required. Most tools need a host computer (for process data) and target system (implemented on FPGA). The comparison between the proposed tool and the tools available in the literature is carried out based on the different parameters, as given in Table I.

As described earlier, FPGA-based fault injection tools can be developed in various stages of the development flow. The controllability and observability parameters for the injection places in the SUT are decreased as moving further in the development flow to bit-stream generation. These tools are mostly producing statistical results. FI tools, developed at the code level, provide maximum controllability and observability. Therefore, the RASP-FIT tool is developed for fault injection at the code level of the FPGA-based designs. It is also observed that the number of fault models is also limited when one move to the development cycle. For example, in the bit-stream file injection, only the SEU model is used. However, the RASP-FIT tool deals with three fault models.

VI. CONCLUSION

Fault injection is an important technique used to test, fault simulation/emulation applications and evaluate the design characteristics and dependability parameters, such as reliability, safety and fault coverage. Fault injection tools are developed for the testing, verification and validation of the FPGA-based designs. All the FPGA-based FI tools are produced in the specific, focused area of interest and the abstraction level of the development flow. In this paper, the usefulness of all tools is highlighted and studied. FI tools, at the code level, have many advantages such as technology independence, high controllability & observability for fault injection places, implementation and simulation are carried out using any simulator engine. No other specialized hardware is required. The RASP-FIT is proposed and developed to help design and test engineers to perform fault injection analysis for the FPGA based HDL designs at the code level with ease. This tool has been developed specifically for Verilog designs.

ISSN – 1999-4974

TABLE I FPGA-BASED FAULT INJECTION TOOLS AND TECHNIQUES

List No:	Tool Name	Category	Fault Models Used	HDL	Functions Perform	Remarks
1	Flipper	Reconfiguration	SEUs	-	Robustness measurement	Fault injection by heavy ion radiation
2	FT-UNSHADES &-2	Reconfiguration	SEUs	VHDL	Reliability measurement	Fault injection by heavy ion radiation
3	NETFI, NETFI-2	Instrumentation	SET/SEU, Stuck at	-both-	Sensitivity evaluation of soft-errors	Fault injection in net-list
4	FITO	Instrumentation	Stuck-at, Bit-flip	Verilog	Observe responses for error and failures	Provides good Controllability, & Observability
5	FUSE	Instrumentation	Saboteur Injections	VHDL	Speed-up Simulation/Emulation	Simulator Engine dependent
6	DFI	Instrumentation	SEUs in registers	VHDL/Verilog	Salient faults, error timeouts	Platform dependent ASTERICS
7	FIFA	Instrumentation	Bit-flip	VHDL/Verilog (RTL)	Robustness calculations	Single & multiple fault analysis
8	FIDYCO	Instrumentation	SEUs in registers	VHDL	Fault tolerance assessment	Platform dependent ASTERICS
9	DBIT	Instrumentation	fault in coding variable, signals, constant	VHDL	independent verification	Verification and Validation (IV&V)
10	RASP-FIT	Instrumentation	SEUs (bit-flip) Stuck at 1 & 0	Verilog	FIA, Testing, Hardness analysis Compaction, Redundancy	Technology independent

ISSN – 1999-4974

REFERENCES

[1] L. Entrena, "Fast fault injection techniques using FPGAs," in 2013 14th Latin American Test Workshop - LATW. Madrid,Spain: IEEE, Apr 2013, pp. 1–1. [Online]. Available: http://ieeexplore.ieee.org/document/6562680/

[2] M. Solinas, A. Coelho, J. A. Fraire, N. E. Zergainoh, P. A. Ferreyra, and R. Velazco, "Preliminary results of NETFI2: An automatic method for fault injection on HDL-based designs," in 2017 18th IEEE Latin American Test Symposium (LATS). IEEE, Mar 2017, pp. 1–4. [Online]. Available: http://ieeexplore.ieee.org/document/7906748/

[3] A. Benso, M. Rebaudengo, M. Reorda, and P. Civera, "An integrated HW and SW fault injection environment for real-time systems," in Proceedings 1998 IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (Cat. No.98EX223). Austin, TX, USA, USA: IEEE Comput. Soc, 1998, pp. 117–122. [Online]. Available: http://ieeexplore.ieee.org/document/732158/

[4] M. Kooli, F. Kaddachi, G. D. Natale, A. Bosio, P. Benoit, and L. Torres, "Computing reliability: On the differences between software testing and software fault injection techniques," Microprocessors and Microsystems, vol. 50, pp. 102–112, May 2017. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0141933116302575

[5] 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, no. 9-11, pp. 1219–1222, Sep 2010. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0026271410004208

[6] J. T. Flaquer, J. Daveau, L. Naviner and P. Roche, "Fast reliability analysis of combinatorial logic circuits using conditional probabilities," Microelectronics Reliability, vol. 50, no. 9-11, pp. 1215–1218, Sep 2010. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0026271410003318

[7] F. Kastensmidt, L. Carro, and R. Reis, Fault-Tolerance Techniques for SRAM-based FPGAs. Boston, MA: Springer US, 2006, vol. 32. [Online]. Available: http://link.springer.com/ 10.1007/978-0-387-31069-5

[8] P. Civera, L. Macchiarulo, M. Rebaudengo, M. Sonza Reorda, and M. Violante, "Exploiting FPGA-based techniques for fault injection campaigns on VLSI circuits," in Proceedings 2001 IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems. San Francisco, CA, USA, USA: IEEE Comput. Soc, 2001, pp. 250–258. [Online]. Available: http://ieeexplore.ieee.org/document/966777/

[9] P. Civera, L. Macchiarulo, M. Rebaudengo, M. S. Reorda, and M. Violante, "An FPGA-based approach for speeding-up fault injection campaigns on safety-critical circuits," Journal of Electronic Testing: Theory and Applications (JETTA), vol. 18, no. 3, pp. 261–271, 2002.

[10] M. Reorda, L. Sterpone, M. Violante, M. Portela-Garcia, C. Lopez-Ongil, and L. Entrena, "Fault Injection-based Reliability Evaluation of SoPCs," in Eleventh IEEE European Test Symposium (ETS'06), vol. 2006. Southampton, UK: IEEE, 2006, pp. 75–82. [Online]. Available: http://ieeexplore.ieee.org/document/1628157/

[11] A. R. Khatri, A. Hayek, and J. Borcsok, Applied Reconfigurable Computing, ser. Lecture Notes in Computer Science, V. Bonato, C. Bouganis, and M. Gorgon, Eds. Cham: Springer International Publishing, 2016, vol. 9625. [Online]. Available: http://link.springer.com/10.1007/978-3-319-30481-6

[12] P. Graham, B. Nelson, and B. Hutchings, "Instrumenting Bitstreams for Debugging FPGA Circuits," in Proceedings of the 9th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'01), Rohnert Park, CA, USA, 2001, pp. 41 – 50. [Online]. Available: https://ieeexplore.ieee.org/document/1420900/

[13] F. Serrano, J. A. Clemente, and H. Mecha, "A Methodology to Emulate Single Event Upsets in Flip-Flops Using FPGAs through Partial Reconfiguration and Instrumentation," IEEE Transactions on Nuclear Science, vol. 62, no. 4, pp. 1617–1624, Aug 2015. [Online]. Available: http://ieeexplore.ieee.org/ document/7181741/

[14] M. Alderighi, F. Casini, S. D'Angelo, M. Mancini, D. M. Codinachs, S. Pastore, C. Poivey, G. R. Sechi, G. Sorrenti, and R. Weigand, "Experimental Validation of Fault Injection Analyses by the FLIPPER Tool," IEEE Transactions on Nuclear Science, vol. 57, no. 4, pp. 2129–2134, Aug 2010. [Online]. Available: http://ieeexplore.ieee.org/document/5550298/

[15] M. Alderighi, F. Casini, S. D'Angelo, S. Pastore, G. Sechi, and R. Weigand, "Evaluation of Single Event Upset Mitigation Schemes for SRAM based FPGAs using the FLIPPER Fault Injection Platform," in 22nd IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems (DFT 2007). Rome, Italy: IEEE, Sep 2007, pp. 105–113. [Online]. Available: http://ieeexplore.ieee.org/document/4358378/

[16] F. Kastensmidt and P. Rech, FPGAs and Parallel Architectures for Aerospace Applications, F. Kastensmidt and P. Rech, Eds. Cham: Springer International Publishing, 2016. [Online]. Available: http://link.springer.com/10.1007/978-3-319-14352-1

[17] J. Nápoles, H. Guzmán, M. Aguirre, J. N. Tombs, F. Muñoz, V. Baena, A. Torralba, and L. G. Franquelo, "Radiation Environment Emulation for VLSI Designs: A Low Cost Platform based on Xilinx FPGA's," in 2007 IEEE International Symposium on Industrial Electronics. Vigo, Spain: IEEE, 2007, pp. C1–C1. [Online]. Available: http://ieeexplore.ieee. org/document/4374556/

[18] S. Cuenca-Asensi, A. Martinez-Alvarez, F. Restrepo-Calle, F. R. Palomo, H. Guzman-Miranda, and M. A. Aguirre, "A Novel Co-Design Approach for Soft Errors Mitigation in Embedded Systems," IEEE Transactions on Nuclear Science, vol. 58, no. 3, pp. 1059–1065, Jun 2011. [Online]. Available: http://ieeexplore.ieee.org/document/5746555/

[19] W. Mansour and R. Velazco, "SEU Fault-Injection in VHDLBased Processors: A Case Study," Journal of Electronic Testing, vol. 29, no. 1, pp. 87–94, Feb 2013. [Online]. Available: http://link.springer.com/10.1007/s10836-013-5351-6

[20] ——, "An Automated SEU Fault-Injection Method and Tool for HDL-Based Designs," IEEE Transactions on Nuclear Science, vol. 60, no. 4, pp. 2728–2733, Aug 2013. [Online]. Available: http://ieeexplore.ieee.org/document/6555963/

[21] W. Mansour, M. A. Aguirre, H. Guzman-Miranda, J. Barrientos, and R. Velazco, "Two complementary approaches for studying the effects of SEUs on HDL-based designs," in 2014 IEEE 20th International On-Line Testing Symposium (IOLTS). IEEE, Jul 2014, pp. 220–221. [Online]. Available: http://ieeexplore.ieee.org/document/6873702/

[22] M. Shokrolah-Shirazi and S. G. Miremadi, "FPGA-Based Fault Injection into Synthesizable Verilog HDL Models," in 2008 Second International Conference on Secure System Integration and Reliability Improvement. Yokohama: IEEE, Jul 2008, pp. 143–149. [Online]. Available: http://ieeexplore.ieee.org/ document/4579806/

ISSN – 1999-4974

[23] M. Jeitler and J. Lech, "Speeding up Fault Injection for Asynchronous Logic by FPGA-Based Emulation," in 2009 International Conference on Reconfigurable Computing and FPGAs. Quintana Roo: IEEE, Dec 2009, pp. 65–70. [Online]. Available: http://ieeexplore.ieee.org/document/5382029/

[24] M. Jeitler, M. Delvai, and S. Reichor, "FuSE - a hardware accelerated HDL fault injection tool," in 2009 5th Southern Conference on Programmable Logic (SPL). Sao Carlos: IEEE, Apr 2009, pp. 89–94. [Online]. Available: http://ieeexplore.ieee. org/document/4914906/

[25] D. de Andres, J. Ruiz, D. Gil, and P. Gil, "Fault Emulation for Dependability Evaluation of VLSI Systems," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 16, no. 4, pp. 422–431, Apr 2008. [Online]. Available: http://ieeexplore.ieee.org/document/4469913/

[26] L. Naviner, J.-F. Naviner, G. dos Santos, E. Marques, and N. Paiva, "FIFA: A fault-injection–fault-analysis-based tool for reliability assessment at RTL level," Microelectronics Reliability, vol. 51, no. 9-11, pp. 1459–1463, Sep 2011. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0026271411002162

[27] B. Rahbaran, A. Steininger, and T. Handl, "Built-in Fault Injection in Hardware - The FIDYCO Example," in Second IEEE International Workshop on Electronic Design, Test and Applications. Perth, WA, Australia: IEEE, 2004, pp. 327– 327. [Online]. Available: http://ieeexplore.ieee.org/document/1409860/

[28] L. Reva, V. Kulanov, and V. Kharchenko, "Design fault injection-based technique and tool for FPGA projects verification," in 2011 9th East-West Design & Test Symposium (EWDTS). Sevastopol: IEEE, Sep 2011, pp. 191–195. [Online]. Available: http://ieeexplore.ieee.org/document/6116608/

[29] A. R. Khatri, A. Hayek, and J. Borcsok, "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. [Online]. Available: http://thesai.org/Publications/ViewPaper?Volume=9&Issue=10&Code=ijacsa&SerialNo=4

[30] 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: IEEE, Aug 2018, pp. 544–551. [Online]. Available: https://ieeexplore.ieee.org/document/8511946/

[31] A. R. Khatri, A. Hayek, and J. Borcsok, "ATPG method with a hybrid compaction technique for combinational digital systems," in 2016 SAI Computing Conference (SAI). London, UK: IEEE, Jul 2016, pp. 924–930. [Online]. Available: http://ieeexplore.ieee.org/document/7556091/

[32] 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. [Online]. Available: http://dx.doi.org/10.14569/IJACSA.2018.091201

 ISSN – 1999-4974