RESEARCH ARTICLE OPEN ACCESS

High Speed Fault Injection Tool Implemented With Verilog HDL on FPGA for Testing Fault Tolerance Designs

G. Gopinath Reddy*, A. Rajasekhar Yadav**, Y. Mahesh***
*(Department of ECE, CREC, Tirupati)
** (Department of ECE, CREC, Tirupati)
*** (Department of ECE, CREC, Tirupati)

ABSTRACT
This paper presents an FPGA-based fault injection tool, called FITO that supports several synthesizable fault models for dependability analysis of digital systems modeled by Verilog HDL. Using the FITO, experiments can be performed in real-time with good controllability and observability. As a case study, an Open RISC 1200 microprocessor was evaluated using an FPGA circuit. About 4000 permanent, transient, and SEUfaults were injected into this microprocessor. The results show that the FITO tool is more than 79 times faster than a pure simulation-based fault injection with only 2.5% FPGA area overhead.
KEY WORDS : Fault Tolerance Design , Gate level Fault Injection, Emulation Phase.

I. INTRODUCTION

Fault injection is mainly used to evaluate fault-tolerant mechanisms. In the last decade, fault injection has become a popular technique for experimentally determining dependability parameters of a system, such as fault latency, fault propagation and fault coverage [1]. Within the numerous fault injection approaches that have been proposed, there are two classifications for fault injection methods [2] hardware-based fault injection [3], [4], and software-based fault injection [5-11]. Software-based fault injection methods are divided into software-implemented fault injections (SWIFI) and simulation-based fault injections. In the simulation-based fault injection, faults are injected into the simulation model of the circuits using VHDL [1], [7], [8], [9] or Verilog[10], [11] languages. The main advantage of simulation-based fault injection as compared with other fault injection methods is the high observability and controllability [10],[2]. However, simulation-based fault injection methods are too time-consuming [2]. One way to provide good controllability and observability as well as high speed in the fault injection experiments is to use FPGA-based fault injection. An effective FPGA-based fault injection technique should support several properties as below:

1. High controllability and observability,
2. High speed fault injection experiments with the target system running at full speed,
3. Capability of injecting permanent and transient faults,

All FPGA-based fault injection techniques that mentioned above inject faults at synthesizable VHDL models of the systems. Because of the use of Verilog hardware description language in implementation of many digital systems, the lack of FPGA-based fault injection tool which supports this hardware description language can be felt. This paper describes the FPGA-based fault injection tool, called, FITO which support all of the fourth properties as mentioned above and is based on Verilog description of the systems. FITO supports several fault models into RTL and Gate-level abstraction levels of the target system which has been described by the Verilog HDL[2]. For supporting high speed fault injection experiments, the fault injector part of FITO with low area overhead is implemented with synthesized microprocessor core inside the FPGA.

II. FAULT MODELS

Digital circuits which are developed by the hardware design languages have hierarchical modeling and can be implemented by several abstract levels. FITO performs fault injection experiments into the gate level and RTL[3] level of the circuits Verilog models.The fault models which are introduced in gate level are the permanent and transient faults. In addition, bit-flip fault is proposed for the RTL level of the digital circuits. Fault injection process can be done by applying some extra gates and wires to the original design description and modifying the target Verilog model of the system. One of these extra wires is the Fault Injection Signal (FIS) which playing the key role in the fault injection experiments. If a FIS takes the value 1, fault would be activated and if it takes the value 0, the fault would become inactive. For each FIS there would be a path through all levels of hierarchy to its modified circuit. After the modification, the final synthesizable Verilog description will be produced which is suitable to use in emulators. In the rest of the paper the synthesizable modification into the Verilog model of the circuit for supporting each fault model has been described.

II.1. Gate Level Fault Injection

FITO supports permanent and transient fault models by generating the modified Verilog source code of the target system for each fault model. The modified Verilog description of the circuit is synthesizable and can be used for FPGA-based fault injection experiments. For supporting the permanent faults in Verilog design, FITO nominates wires for fault injection and apply the FIS signal with one extra gate. So, by selecting the FIS signal high at fault injection time, the permanent fault into the specified wire will be injected.

Figure 1 shows the Verilog source code modification for supporting stuck-at fault models. FITO uses one timer for determining the fault injection time. It also uses another timer for finishing the fault injection experiment (workload execution). After reaching the fault injection time, the FIS signal will be high and another timer starts to count. As shown in figure 1 wire TX is the additional wire which is applied to the original design and the every wire namely X will be replaced by TX.In addition, FITO can generate synthesizable modified Verilog source code of the target system for supporting transient faults. The modified circuit that is suitable for transient fault injection is shown in figure 2. After reaching the fault injection time, the FIS signal will be high and the timer which have been loaded with the duration of the transient fault injection start to count. Therefore, the FIS will be high (at logic 1) for the specified duration of time. As similar to the permanent fault, the additional wire (TX) will be used and each wire, namely X will be replaced with TX. Note, the fault injector part of FITO which is called Fault Injection Manager.

II.2. RTL Level Fault Injection

The fault model that is used by FITO at this level is bit-flip (or Single Event Upset). SEUs are the random events and may flip the content of the memory element at unpredictable times. FITO generate modified circuit for each memory element that is specified for fault injection.The modified circuit for supporting bit-flip fault model is shown in figure3.

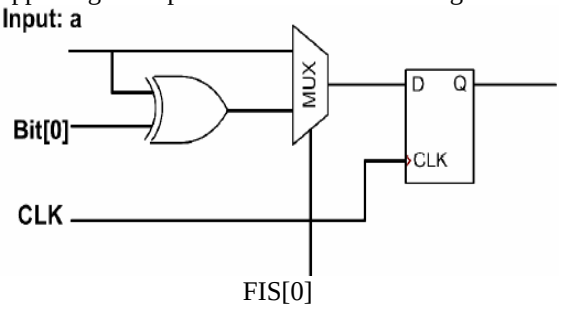

Figure 1. Synthesizable bit-Flip fault model

For supporting the bit-flip fault model, FITO produces the additional signals such as Bit and FIS with one multiplexer. The Verilog synthesizable code for supporting this fault model is shown in figure 3. The inverted input will be goes to the flip-flop for the next clock that FIS and Bit are 1. FIS indicates the target register and the Bit will be high for the target register's bit. The fault injection manger part of FITO is responsible for setting and resetting the FIS and Bit signals.

III. THE FITO ENVIRONMENT

FITO is made of three main parts that every part is used in different fault injection phases. These parts are:

1. Source Code Modifier & Fault List Generator
2. Fault Injection Manager
3. Result Analyzer

Source Code Modifier & Fault List Generator and Result Analyzer are the software parts of the FITO which are located on the host computer. On the other hand, Fault Injection Manager is responsible for performing the real-time fault injection. This hardware part is implemented on the FPGA board.

The fault injection process with FITO has been shown in Figure 4. As shown in this figure, each FITO's part that were mentioned before are used in different phases of the fault injection process. In the rest of the paper, each fault injection phases and the main work of each FITO's part in these phases will be described in more details.

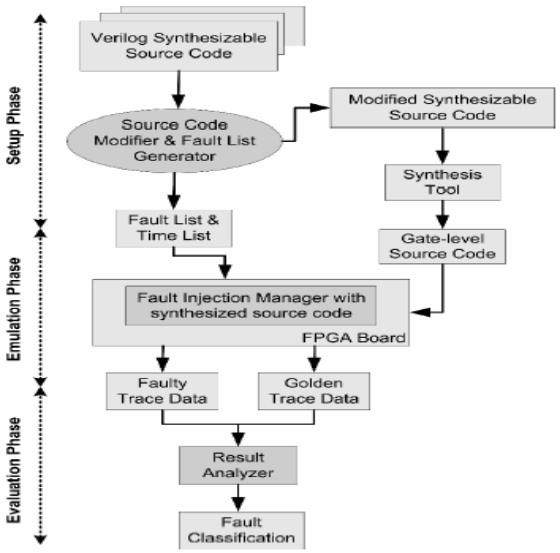

Figure 2. Fault injection process with FITO

III.1. The Setup Phase

The main objectives of this phase are achieving modified Verilog source codes of the original model that is synthesizable and generating correspond fault list for each fault injection experiments.

In setup phase the Verilog models have been given to the FITO. First, by selecting all or some of the considered fault models, the Source Code Modifier processes the Verilog model of the system. After user specifies the main module, a source navigator shows

the wires and registers to user. After selecting the fault injection properties and the observation points, FITO generates the corresponding fault list, time list and the synthesizable modified source code. The synthesizable modified source code has additional flip-flops for each observation points.

Each time list indicates the time for triggering each fault injection experiment and the fault list is used for indicating the fault injection location. A typical fault list is described in figure 5. As shown in figure 5, the first bit of fault list is used for performing the fault injection experiment. In addition, two bits and eight bits are the inputs to decoder A and B. Outputs of decoder A and B are Bit[3:0] , FIS[255:0] which together indicate the bit position of the target register for bit-flip fault injection. The FIS[255:0] without Bit[3:0] are used for supporting permanent and transient fault models.

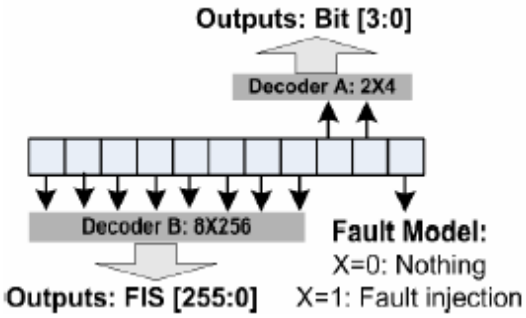

Figure 3 . Fault list format

Modified source code contains fault injection manager with modified circuit. So, the target system is suitable for fault injection experiments. Decoder A and B are the main parts of the fault injection manager.

After this step, the modified source code must synthesize with some synthesis tool and the gate level source code which is suitable for programming the FPGA will be produced. By using the gate level source code the FPGA will be programmed.

III.2. The Emulation Phase

In the emulation phase, modified codes created by the previous phase are emulated. After emulating each experiment, the information of the observation points will be sent through the serial port. So, each experiment will have one trace file. Each trace file is created with the observation data points of each experiment. Results of this phase are providing 1) one fault free trace file and 2) faulty trace files which are generated by performing faulty experiments. During this phase, the Result analyzer part of FITO must be run from the user. This part sends each fault list and time list of the fault injection experiment to the fault injection manager. Then, the fault injection manager sends the contents of the observation points to the result analyzer. At the start of the fault injection experiments, the fault injection manager reset the first bit of fault list for creating the golden trace file. Then, each fault list and time list is sent to the FPGA board. After the fault injection the contents of the observation points are sent to the host computer for analyzing the system behavior.

III.3. The Evaluation Phase

The main objective of this phase is the fault tolerance parameter estimation. It is done by result analyzer software part of FITO that is located on the host computer. Result analyzer estimates the dependability parameters by tracing differences between golden run and faulty trace files. Some facilities were developed for user interactions and for required fault tolerant parameter determination.

IV. EXPERIMENTAL RESULTS

We developed the fault injection using the Altera DSP development board, equipped with Strati EP1S25F780C FPGA. An OpenRISC 1200 has been used as benchmark for FITO evaluation. The main reason for using OpenRISC 1200 is that it has synthesizable Verilog Description and intended for embedded systems, automotive, portable computer environments. In the experiments, two common workload programs are considered [10]. The matrix multiplication and the bubble sort. The workloads are coded in C and are compiled with GNU gcc compiler. So, after this step, the suitable code for the OpenRISC 1200 microprocessor will be generated. After this step we connected instruction and data memory to the processor with the workload which is loaded into the instruction memory.

.Table 1: Available and consumed FPGA resources (EP1S25F780C5)

	#	%
Total Available LEs in the FPGA	25660	100
LEs used by the OpenRISC 1200	4769	18.58
LEs used by the OpenRISC 1200 + FI	5401	21.04

The faults are injected in different parts of the CPU modules of the OpenRISC 1200 core consisting of control unit, the genPC unit, the Instruction Fetch unit and the ALU unit. The total runtime of the matrix multiplication and bubble sort were 990 and 5890 clocks. In this experiment total 4000 permanent and transient faults injected at 100 random locations. For each location of the every fault, experiments were carried out 20 times with uniform distribution during the running of the each workload. The fault duration for transient faults were one clock period. The OpenRISC 1200 microprocessor emulated using 80 MHZ clock. The observation points are the address bus, data bus and the register file.

Table 2 shows the speed-ups. As shown in table 2, the resulted speed-up is workload dependent. This is because bubble sort workload generates

more signal event than matrix multiplication.

Table 2: The Resulted Speed-ups

Workload	Simulation Time (sec)	Emulation Time (sec)	Speed-up
Matrix Multiplication	4605	51	90
Bubble Sort	13770	199	69

The fault propagation results, fault models for each module and the number of fault injection points have been shown in table 2.

As shown in table 2, different fault models are considered for each module of the Open RISC 1200 microprocessor. The Control Unit plays the key role in controlling the pipeline registers of the microprocessor. So, the transient fault model for the internal wires of this module was considered. The pc register which is the most important register of the system for controlling the flow of the workload is considered for bit-flip fault injection. So, the bit- flip fault model was considered for the Genpc unit that involves pc register.

V. C0MPARISON WITH FPGA-BASED FAULT INJECTION TOOLS

For estimating the main properties of FITO that were mentioned in section 1, a comparison between FITO and other fault injection tools is needed. FITO provides controllability over 255 wires and registers of the target microprocessor which is sufficient for having the control over the important wires and registers of the target microprocessor. Because of using the combinational logics (two decoders) and compacted fault and time lists the area overhead of FITO is very lower than the FIDYCO and FIFA and it uses one flip-flop for every fault injection location. The minimum 22% area overhead has been reported for FIFA tool.

VI. CONCLUSION

This paper described the FPGA-based fault injection tool, called, FITO for evaluating the digital systems modeled by Verilog HDL. Fault injection with FITO is done by applying some extra gates and wires to the original design description and modifying the target Verilog model of the target system. FITO support some properties such as high speed, good controllability, good observability and low area overhead. As a case study, an OpenRISC 1200 have been evaluated on the EP1S25F780C FPGA and 4000 faults have been injected into this microprocessor. The effects of faults have been classified into control flow errors, data errors and failures activated. Results show that the FITO is more than 79 times faster than simulation-based fault injections with only 2.5% FPGA overhead.

REFERENCES

[1] V. Sieh, O. Tschache, and F. Balbach, "VERIFY: evaluation of reliability using VHDL-models with embedded fault description," *Proc. of the International Symposium on Fault-Tolerant Computing,* Jun. 1997, pp. 32-36.

[2] P. Folkesson, S. Sevensson, and J. Karlsson, "A Comparsion of Simulation Based and Scan Chain Implemented Fault Injection," *Proc. of the Annual International Symposium on Fault-Tolerant Computing,* Jun. 1998, pp. 284-293.

[3] H. Madeira, P. Joao, and G. Silva, "RIFLE: A General Purpose Pin-level Fault Injector," *Proc. of the European Dependable Computing Conference,* 1994, pp. 199-216.

[4] J. Arlat, M. Aguera, L. Amat, Y. Crouzet, J. C. Fabre, J. C. Laprie, E. Martines, and D. Powell, "Fault Injection for Dependability Validation - A Methodology and Some Applications," *Trans, on the IEEE Software Engineering,*^. 1990, pp. 166-182.

[5] J. Carreira, H. Madeira, and J. G. Silva, "Xception: A Technique for the Experimental Evaluation of Dependability in Modern Computers," *Trans, on the IEEE Software Engineering, Feb.* 1998, pp. 125-136.

[6] Z. Segall, and T. Lin, "FIAT: Fault Injection Based Automated Testing Environment," *Proc. of the International Symposium on Fault-Tolerant Computing,* Jun. 1988, pp. 102-107.

[7] E. Jenn, J. Arlat, M. Rimen, J. Ohlsson, and J. Karlsson, "Fault Injection into VHDL Models: The MEFISTO Tool," *Proc. of the International Symposium on Fault-Tolerant Computing,* pp. 336-344, Jun. 1994.

[8] T. A. Delong, B. W. Johnson, and J. A. Profeta, Iii, "A fault injection technique for VHDL behavioral-level models," *Proc. of the IEEE Design & Test of Computers,* 1996, pp. 24-33.

[9] J. Bou, P. Petition, and Y. Crouzet, "MEFISTO-L: A VHDL-based fault injection tool for the experimental assessment of fault tolerant," *Proc. of the International Symposium on Fault-Tolerant Computing,* Jun. 1998, pp. 168-173.

[10] H. R. Zarandi, G. Miremadi, and A. R. Ejlali, "Fault Injection into Verilog Models for Dependability Evaluation of Digital Systems," *Proc. of the International Symposium on Parallel and Distributed Computing,* Oct. 2003, pp. 281-287.