International Journal of Modern Engineering Research (IJMER)
www.ijmer.com Vol. 3, Issue. 5, Sep - Oct. 2013 pp-2894-2900 ISSN: 2249-6645

High Speed Fault Injection Tool (FITO) Implemented With VHDL on FPGA For Testing Fault Tolerant Designs

M. Reddy Sekhar Reddy, R.Sudheer Babu (Ph.D)
M.Tech (VLSI&ES) GPREC, Kurnool.
Assistant Professor, GPREC,Kurnool.

ABSTRACT—*Fault injection is mainly used to test and evaluate the fault-tolerance based designs. In current VLSI technology fault injection has become a popular technique for experimentally verifying the fault tolerant based designs. There are fundamentally two types of fault injection methods; they are hardware-based fault injection and software-based fault injection. Both have their own limitations and advantages. The FPGA synthesizable fault injection model can give reasonable solution with high speed testing platform and also allows good controllability and observability. Even though a considerable progress has been made in research part of the fault injection algorithms, there is a little progress done in developing a tool for FPGA based fault emulation.*

In this paper an FPGA-based fault injection tool (FITO) that supports several synthesizable fault models of digital systems are implemented using VHDL. Aim is to build real time fault injection mechanism with good controllability and observability. Fault injection will be done by applying some extra gates and wires to the original design description and modifying the target VHDL model of the target system. The design will be validated with state machine based example and applying different types of faults. Analysis will be carried out studying the controllability and observability of the proposed scheme. Comparison will be carried out to estimate the speed wise improvement with respect to software simulation based fault injection method.

Modelsim Xilinx Edition (MXE) will be used for functional simulation and Xilinx ISE tools will be used for synthesis and performance analysis. Spartan-3E FPGA board will be used for on chip verification of the results with Chip scope software running on PC.

Index Terms—*VLSI, FITO, VHDL, fault modelling, Modelsim, Xilinx, GUI, Spartan 3E FPGA kit.*

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 of fault injection methods [2]: 1) hardware-based fault injection and 2) software-based fault injection. 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 or Verilog languages. The main advantage of simulation-based fault injection as compared with other fault injection methods is the high observability and controllability [2]. However, simulation-based fault injection methods are too time-consuming. 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 [3]. 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,
4) minimum area and time overhead into a target system. The techniques which developed for fault grading using emulators proposed in[3]. These techniques are limited to the single stuck at fault model.So they don't have the capability of injecting transient faults and don't support the e third property. In an extension to transient faults is proposed, but the approach is based on reprogramming the FPGA once for each fault. The reconfiguration, even if partial, results in a time overhead.Therefore this technique doesn't support the fourth property.[4]Perform fault injections by using the additional combinational and sequential circuits. Because of using the additional flip-flop for each fault injection location, these techniques introduce too much area overhead and don't support the fourth property. So, these techniques are not sufficient for using on the same chip with the target microprocessor after the fabrication.

The main idea of using fault injector on the same chip with the target microprocessor was proposed in [5]. Because of implementing the most part of the fault injection tool on the target microprocessor, they suffer from the main drawback of high area overhead. This paper describes the FPGA-based fault injection tool, called, FITO1 which support all of the fourth properties as mentioned above and is based on VHDL 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 VHDL. 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. DESIGN OF FITO

In this chapter the FITO design flow is explained in detail. Fault models that are implemented in our project are clearly explained

A. FITO

FITO environment consists of three parts

1) Source code modifier and fault list generator.
2) Fault injection manager
3) Results capturing with FPGA emulation

1) Source code modifier and fault list generator

Source code modifier and fault list generator are the software parts of the FITO. This is implemented with two tools.

- Eclipse editor
- C programs to insert faults on a specific port.

These are located on host (PC) computer. Separate C scripts are developed for inserting each fault. A GUI facilitates invoking the scripts by mouse right click.

2) Fault injection manager

Fault injection manger is responsible for performing the real time fault injection. The fault injection manager is implemented in VHDL. The fault injection manager

- VHDL package (dynamically updated by C programs)
- Fault scheduler
- Fault insertion components

The VHDL package is implemented to capture all the constants, type definitions, component declarations and fault injection time for each fault. The package also consists of number of total faults. This VHDL file is automatically updated by C programs every time when a fault is injected in code.The fault scheduler runs multiple counters to schedule each fault with required fault activation time and fault propagation time as per the package. The fault scheduler produces output fault number which is currently being active. This module generates the parallel fault injection signals for every fault. These signals are routed to all fault sites. Fault insertion components are gates with FIS (fault injection signal) control to inject the faults when the FIS is active high. These components instances are automatically made when ever faults are injected.

3) Results capturing with FPGA emulation

This hardware part is implemented on the FPGA board. Result analysis will be carried out with FPGA emulation results and fault list generated by C program. The analysis shall summarize the fault responses for each injected fault. The following Fig shows the fault injection process with FITO.
The golden trace data is ideal trace data (results obtained) without any faults.

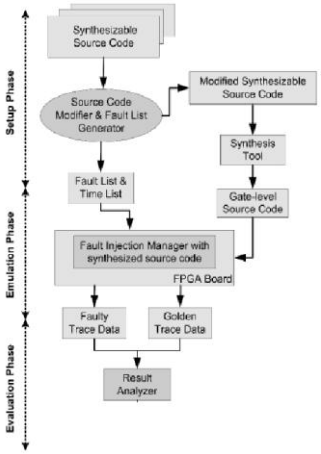

Fig. 1. FITO injection process.

B. Faults Modelled In Our Project

FITO supports the following synthesizable fault models for injecting into any HDL level designs.

- Permanent faults
- Transition faults
- Single event upset faults (or) Bit-flip

Fault injection process can be done by applying some extra gates and wires to the original design description and modifying the target VHDL model of the system. One of these extra wires is the Fault injection system (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 example in the case of Stuck-at-0 fault when the FIS is made 1 then the signal is forced to zero, implementing the fault condition. The below section gives the detailed discussion about injecting the permanent faults.

1) Permanent faults

For supporting the permanent faults in VHDL 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.

For example if the signal name in the original code is X then the modified signal TX will be generated as below. In all the places in the code instead of X, the TX will be replaced.

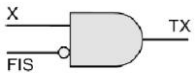

Fig. 2. Synthesizable fault model for stuck-at-0.

Similarly the following code shows the required extra gate and control signal FIS for implementing the stuck-at-1 fault.

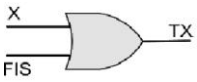

Fig. 3. Synthesizable fault model for stuck-at-1.

For each FIS there would be a path through all levels of hierarchy to its modified circuit. After modification, the final synthesizable VHDL description will be produced which is suitable to use in emulators.

2) Transient faults

The modified circuit that is suitable for transient fault injection is shown in below Fig.

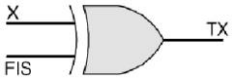

Fig. 4. Synthesizable transient fault model.

For injecting a transient fault, after reaching the fault injection time, the FIS signal will be made 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 replacedwithTX. The fault injection manager is responsible for managing the fault injection experiments, such as loading the timers, setting the FIS for the predetermined time, introducing additional wires and performing the fault injection.

3) Bit flip (or) single event upset (SEU)

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 below Fig.

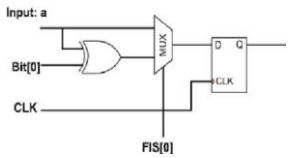

Fig. 5. Synthesizable bit-flip model.

For supporting the bit-flip model, FITO produces the additional signals such as Bit and FIS with one multiplexer. The VHDL synthesizable code for supporting this fault model is shown in above Fig. The inverted input will go to the flip-flop for the next clock when the FIS and bit are "1".

The fault injection manager part of FITO is responsible for setting and resetting the FIS and bit signals.

C. Example Fault Tolerant Design – Redundant ALU Based Fault Tolerant Processor

In this the example fault tolerant design considered for fault injection is explained.The design uses double ALUS

and to achieve fault tolerance in case of soft errors.

A soft error is also a signal or datum which is wrong, but is not assumed to imply such a mistake or breakage. After observing a soft error, there is no implication that the system is any less reliable than before. If detected, a soft error may be corrected by rewriting correct data in place of erroneous data. Highly reliable systems use error correction to correct soft errors on the fly. However, in many systems, it may be impossible to determine the correct data, or even to discover that an error is present at all. In addition, before the correction can occur, the system may have crashed, in which case the recovery procedure must include a reboot.

1) Proposed fault tolerant processor architecture:

The proposed architecture is double ALU based fault tolerant for handling soft errors. The Fig. 6. describes the detailed architecture of fault tolerant processor [6].

Definition - Fault-tolerant describes a computer system or component designed so that, in the event that a component fails, a backup component or procedure can Immediately take its place with no loss of service. Fault tolerance can be provided with software, or embedded in hardware, or provided by some combination.

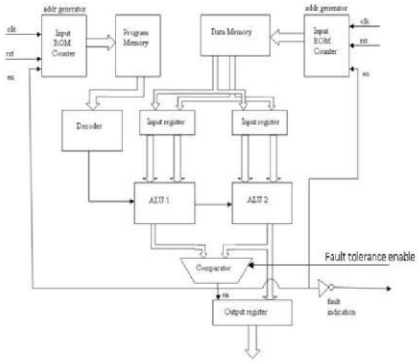

Fig. 6. Architecture of fault tolerant processor.

At a hardware level, fault tolerance is achieved by duplexing each hardware component. Disks are mirrored. Multiple processors are "lock-stepped" together and their outputs are compared for correctness. When an anomaly occurs, the faulty component is determined and taken out of service, but the machine continues to function as usual.

The same technique is used in our architecture by duplexing the ALU unit and comparing the results. An enable signal is provided by the comparator which is used to iterate the function and produce fault free results. The address generator part of the block diagram access the data and is stored in two different input registers and are processed separately by different arithmetic and logic units to achieve redundancy. The enable signal also indicates the occurrence of faulty signal. When it is assured that the fault has not occurred, the output is taken from the output register.

III. IMPLEMENTATION OF FITO MODULES

In this chapter the VHDL modules implemented for FITO are explained in detail. As explained in chapter 2 the Fault injection manger is responsible for performing the real time fault injection. The fault injection manager is implemented in VHDL. The fault injection manager

- VHDL package (dynamically updated by C programs)
- Fault scheduler
- Fault injection components

A. FITO - Package

The VHDL package is implemented to capture all the constants, type definitions, component declarations and fault injection time for each fault. The package also consists of number of total faults. This VHDL file is automatically updated by C programs every time when a fault is injected in code.

The maximum number of faults are taken to be 63 and based on that the other constants are defined. However there is no limitation of the maximum number of faults that can be inserted. Depending on the requirement one has to the constant's values in this package .Another constant is defined sto give the number of injected faults. This constant value is updated by C program every time when a new fault is inserted. FIS_vec_type defines a bus of size equal to number of faults. This bus goes through all modules such that any module can use the control lines for fault injection. It may appear that by routing 64 length wider bus to all small and big modules of design under test we are consuming high number of FPGA routing resources. But the synthesis tool can optimize the resources by only routing the lines which are used in this module. Constant by name Fault injection signal (FIS) high duration indicates the number of clock cycles for which fault will be injected in the design. FIS duration constant tells the time allotted for each fault. That is even after removing the fault we can wait for output to capture before enabling the next fault. This will be useful in cases where the fault propagation time is high. For every fault these two constants are settable in the GUI.

The constant fault_type defines the type of fault as per the below table. For each fault that is injected used will choose this option on GUI.

TABLE I: TYPES OF FAULTS.

Constant value	Fault type
0	Stuck at 0
1	Stuck at 1
2	Transient
3	Bit flip

Package also holds several component declarations. So that in all module if this package is declared then fault injection only requires to give component instantiation (no need to declare the components)

1) Fault scheduler

The fault scheduler runs multiple counters to schedule each fault with required fault activation time and fault propagation time as per the constants in FITO_package. The fault scheduler produces output fault number which is currently being active. This module generates the parallel fault injection signals for every fault. These signals are routed to all fault sites.

Fig. 7. Fault Scheduler

2) Fault injection components

Fault injection components are gates with FIS (fault injection signal) control to inject the faults when the FIS is active high. These components instances are automatically made in the selected module when ever faults are injected. Since all these components are declared in package fault injection need not add component declaration. Hence the fault insertion becomes easy to implement only the following steps.

- Generate code to declare a signal of the same size of the port on which fault need to be injected.
- Add the corresponding fault injection component instance connecting the port signal, FIS control line and output signal.
- Replace all the port signal instances with the declared new signal.

3) Random bit generator for bit flip fault

A random bit generator for bit flip fault is implemented with a Gaussian random variable generated through a Look up table. A Look up table with 127 values is taken and is used to randomly flip the bits in memory when the bit flip fault is activated.

SIMULATION RESULTS

Simulations results of Fault tolerant processor without injection of faults and with injection of faults are shown in this section.

The top level test bench module implements Fault tolerant processor with out injecting faults and after injection of faults with necessary test inputs. The following Fig shows the simulation results obtained by simulating the test bench.

A. Fault_Tolerant_Processor (Without Injecting Faults)

Fault tolerant processor with out injecting faults is implemented with VHDL and simulation results for this are shown below.

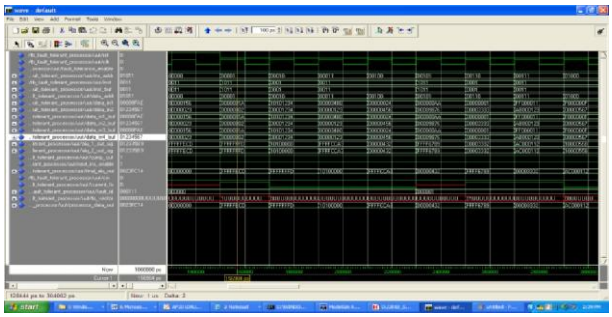

Fig. 8. The fault tolerant processor without injecting any faults.

From ins_address 00000 to 01000 the alu operations are shown in above.

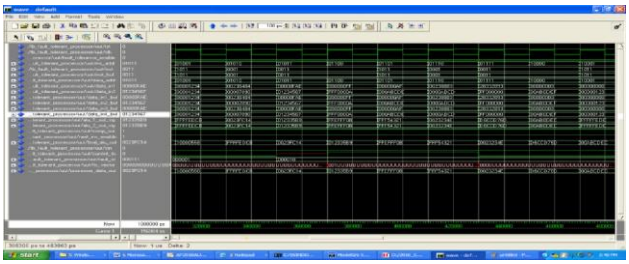

Fig. 9. The fault tolerant processor without injecting any faults (continued).

From inst_address 01001 to 10001 the alu operations which are performed are shown in above.

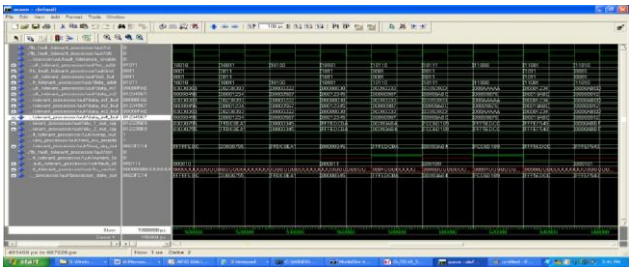

Fig. 10. The fault tolerant processor without injecting any faults (continued).

From ins_address 10010 to 11010 the alu operations which are performed are shown in above.

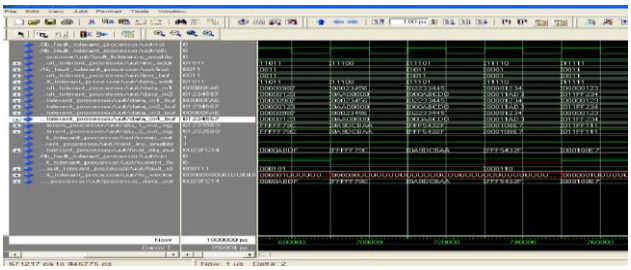

Fig. 11. The fault tolerant processor without injecting any faults (contined)

The ins_address from 01011 to 11111 the alu operations which are performed are shown in above.

B. Simulation Results of Fault tolerant Processor after Faults Have Been Injected in to the Processor are Shown Below

1) Fault tolerant processor after injection of faults

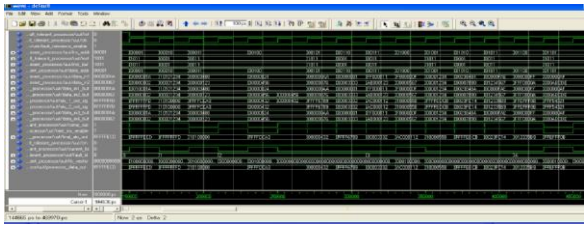

Fig. 12. Simulation results after injection of faults.

In this we injected 11faults from 0th to 10^{th} hence the fault id that is shown above is from 0 to 11.Fault id is represented presently which fault is scheduled. Current_FIS represents on each fault id how much time the current fault should be injected i.e., activated.

The 1^{st} 2^{nd} 3^{rd} 4^{th} faults are injected and results are shown in above.

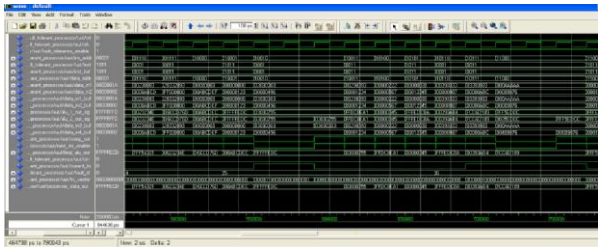

Fig. 13. Simulation results after injection the faults (contd.).

The results of 5th and 6th faults are shown in above.

The results of 7th 8th 9th and 10th fault responses and output are shown below

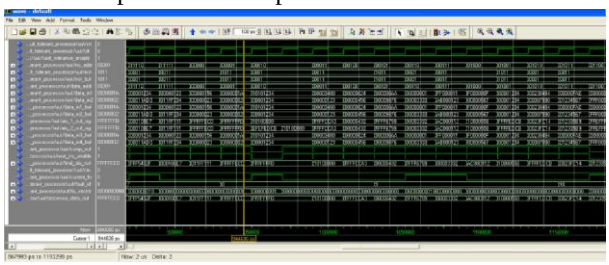

Fig. 14. Simulation Results after injection the faults(cont)

IV. CONCLUSION

The project implements the FPGA based fault called FITO for evaluating the digital systems modelled by VHDL.Fault injection with FITO is done by applying some extra gates and wires to the original design description and modifying the target VHDL model of the target system. FITO support some properties such as high speed good controllability and observability and low area overhead.We have taken a example of fault tolerant design and evaluated on FPGA and faults have been injected in to this processor and proved that really this fault tolerance feature is implemented and also proved that FITO is more faster than simulation based fault injection.

This paper is for real time application for testing the fault tolerant designs and several other fields of VLSI testing.

ACKNOWLEDGEMENT

We would like to thank the Faculty of G Pulla Reddy Engineeering College,Kurnool for their support in the Implementation of High Speed Fault Injection Tool (FITO) Implemented with VHDL on FPGA for Testing Fault Tolerant Designs

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 compa-rsion 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] K.-T. Cheng, S.-Y. Huang, and W.-J. Dai, "Fault emulation:A new methodology for fault grading," *Trans. on the IEEE Computer-Aided Design of Integrated Circuits and Systems*, Oct. 1999, pp. 1487-1495.

[4] P. Civera, L. Macchiarulo, M. Rebadengo, M. S. Reorda, and M. A. Violante, "Exploiting FPGA for accelerating fault injection experiments," *Proc. of the International On-Line Testing Workshop*, 2001, pp. 9-13.

[5] B. Rahbaran, A. Steininger, and T. Handl, "Built-in fault injection hardware – The FIDYCO example," *Proc. of the IEEE International Workshop on Electronic Design, Test and applications.*

[6] "Fault tolerant Architecture Manual." [Online]. Available: http://www.opencores.org