Instrumentation Technique for FPGA based Fault Injection Tool

ABDUL RAFAY KHATRI, MANUEL MILDE, ALI HAYEK, JOSEF BÖRCSÖK
Department of Computer Architecture and System Programming
University of Kassel
Wilhelmshöher Allee 71-73, 34121 Kassel
GERMANY
arkhatri@student.uni-kassel.de

Abstract: - This paper presents an overview about FPGA based fault injection tools which are developed by using the instrumentation technique or modification in the original code. The fault injection technique is used to evaluate dependability parameters of computer based embedded systems or safety critical systems, by injecting faults in a system. An observation on the behaviour or of the response for both -fault free and faulty systems- is made and the results are matched. In the instrumentation based technique, faults are injected by adding some additional circuits in the target system known as fault models. Different fault models are explained in this paper, e.g. Stuck at 0/1, Bit flip, Mutant and Saboteur. A variety of FPGA based tools are discussed and a new FPGA based fault injection technique is introduced. The tool that has been developed according to our methodology can work on any simple or complex digital circuits or SoC realized on FPGA. This can be exercised to evaluate the dependability parameters for the target systems under test e.g. fault coverage, fault latency and reliability.

Key-Words: - FPGA, Fault Injection, FPGA based fault injection tools, Instrumentation, Fault models, Fault coverage

1 Introduction
Recently the principle of fault injection has become a widely used technique in order to calculate dependability parameters of embedded and safety critical systems. These systems are based on computers or microprocessor systems which are used mostly in areas where failure leads to a huge problem, such as railway traffic control, aircraft flight schedules and space craft etc. Due to the condensed size of components in electronic systems, it is difficult to guarantee an acceptable degree of operational reliability of a system, hence testing and verification are more important [1][2]. The dependability parameters for embedded systems and computer based systems are verified before they are handed over to the end user. Four main methods exist in which fault injection testing is possible namely:

- Hardware based fault injection
- Software based fault injection
- Simulation based fault injection
- Emulation based fault injection

There are certain merits and demerits of every method stated above. Detailed information about this, is presented by H Ziade et al [3]. Recent development in the field of programmable logic devices i.e. FPGA (Field Programmable Gate Array) has allowed safety critical systems and computer based systems to be realized on it. It provides many properties like re-configurability, higher speed, higher performance and cost effectiveness [4], but FPGA devices are also required testing. For that purpose various FPGA based fault injection tools and techniques are presented in the literature.

The FPGA based fault injection method is the most widely used nowadays because of the FPGA’s features mentioned above. The FPGA based tools cover all benefits of simulation based tools. Furthermore, they are able to overcome the major disadvantages of time consumption and hence increase the speed of the fault injection process. FPGA based fault injection tools can be categorized into two main groups [2].

1) Reconfiguration based technique
2) Instrumentation based technique

The first group takes advantage of the features provided by FPGA, i.e. reconfiguration, partial reconfiguration. In this technique the faults are injected by changing bits in different copies of the bit stream. The major downside of this method is the reconfiguration of the FPGA every time the fault injection experiment is performed, which is the source of time overhead. While in the technique of the second group additional circuits are added or a
modification of the original code is carried out, which contributes in area overhead. This paper is focused on the second approach and presents the latest techniques and tools for FPGA fault injection. The Section 2 describes background or previous work. Section 3 illustrates the basic building block and functioning of any fault injection tools. Section 4 presents the various fault models that are most widely used in this field. Section 5 presents a new proposed approach for the development of instrumentation based FI tool for FPGA systems and devices. Finally, the conclusion of this paper is presented.

2 Background

The system, which is designed to perform some particular task, does not give the output which is desired. Different reasons can be considered for this irregularity in output. Those reasons are called factors of dependability which incorporate fault, error and failure [5]. Most researches have taken place in order to validate and verify the dependability of a target system. More work has been produced for most commonly used dependability evaluation techniques i.e. fault injection techniques.

Single event upsets are the most recurrent faults in semiconductor devices, in which the value of a particular bit is changed from its original value and this causes some errors. These faults occur mostly in configuration memory, user memory and registers in processors. Some tools are also designed to determine and validate these faults. FITVS is one of them. It stands for Fault Injection Tool for Validating SEEs. This tool is used for grading faults in VLSI and to provide high speed up to 1µs/fault. The faults are injected through modification of the code using the saboteur’s fault model. It does not require FPGA reconfiguration. The process is highly automated with the help of a C program. FITVS does not rely on circuit size and easily intermixes with Verilog and VHDL. Various benchmark circuits were tried out and the results including failure, silent and latent faults are presented by H. Zheng et al [6].

In 2011, L.Reva, V.Kharchenko, et al presented a way to implement a fault injection technique named DBIT [7]. According to them, this proposed tool can be used for fault profiling and injection. The fault injection is accomplished on the VHDL source file and different faults can be injected like faults in signal/variable names, constants, operators, assignment and conditional statements and/or faults in the component’s instantiations. The injection process can be automatic, semi-automatic or manual. During the same year, L.A.B Naviner et al from the institute of the Telecom ParisTech developed a fault injection tool named FIFA (Fault Injection and Fault masking Analysis)[8]. This tool is designed to expedite the process of fault injection. In the FIFA tool, the special saboteurs are inserted in the design to make them faulty and the same golden model is used as fault free. These saboteur models are based on the different fault models like Stuck-at 0/1, single and multiple SEUs. After that, responses are matched and the robustness of digital circuits is examined.

A. Mohammadi et al in 2012 presented an instrumentation based FI technique, which is based on debugging facilities including In-system Memory Content Editor (MCE), In-system Sources and Probes (SAP) for the fault injection purposes [9]. The tool named SCFIT (Shadow Components based Fault Injection Technique) uses commercial tools to implement saboteurs. In the same year J. Grinschgl et al presented their work on fault injection techniques. They implemented the Modular Fault Injector (MFI) concept and used multi-bit faults and saboteur models[10].

In 2013, a FPGA based framework for fault injection analysis was described in order to identify critical variables and most critical registers in a whole microprocessor [11]. The register values are altered to the wrong value by inverting their data. C code was also written to allocate faulty values to different variables in the process and different approximations were used to analyse the critical variables inside the microprocessor. In the same year, a FPGA based method was described, which speeds up the fault injection process by implementing all parts of the tool inside the FPGA [2]. By implementing this method, the communication between the PC and target system is not compulsory as all fault injection components are inside FPGA. This model takes the bit flip fault model into consideration. The faults are injected at various points in the system’s memory and processor’s registers. When activated, the additional circuits insert faults in the system. The results of faulty and fault free systems are compared and critical or sensitive parts are observed with regard to SEU.
TIMA labs France is working on robust circuit architecture and design, verification of SoC and integrated systems. They presented their work on SEU fault injection in VHDL based processors [12]. They introduced a new fault injection technique named Direct Fault Injection (DFI). The crucial point of their strategy is the modification of VHDL code by adding extra circuits i.e. Saboteur fault model. The DFI injects faults in registers that can or cannot be accessed by an instruction set.

3 Fault Injection Environment

From literature survey, it is beheld that fault injection tools are designed with the use of a variety of approaches and methods. Basically two devices are needed to perform this whole operation, a host PC and a target board (FPGA board for emulation). Mostly both devices communicate on serial links for debugging. The host PC contains the software part which is comprised of the fault list manager, the fault list and the result analyser while the target board holds the fault injection manager or the fault injection unit. The fault injection system can be divided into three fundamental modules which are outlined below [2][6][13].

3.1 Fault list manager

The Fault list manager is in charge of generating the list of faults (SEUs) which will be injected in the target system (FPGA board). This handles the types of faults injected in the system.

3.2 Fault injection manager

The main purpose of the Fault injection manager is to control and activate faults from the fault list manager on the target system. In circuit instrumentation, additional circuits are added for faults whose activation is controlled by this block.

3.3. Result Analyser

The task of this module is to comprise the gathering of the results so that a report can be generated during the fault injection experiment. It is beheld that the fundamental operation of any fault injection tool is to inject the fault in the target system and observe the output before and after the activation of faults. Following this, the next both results are compared. Based on that, fault coverage, fault latency and other dependability parameters are examined.

In FPGA based fault injection techniques and tools, faults can be injected in many points as shown in Figure 1[14]. In order to develop the FPGA based tool, there are different abstraction levels to program a FPGA e.g. gate level, data flow and RTL level. The procedure to program a FPGA consists of certain steps of programming from higher abstraction level VHDL or Verilog program to bitstream generation. Different tools and techniques have been developed for various points of designed modification as shown in Figure 1. For example: Rapidsmith is a JAVA API tool that works on the *.ncd (native circuit descriptive) file generated after the place and route process. This tool converts the *.ncd file into a XDL (Xilinx Design Language) file, modifies the design and converts it back to the *.ncd file for further processing[15]. Another example is JBits, which is also a JAVA API. It works on the FPGA bit stream and modifies it [16].

In this paper RTL and gate abstraction levels are picked out. The code can be modified in order to inject faults in the target circuit/system. The
Contributions in this regard are presented in the background section.

4 Fault Models

From the detailed study, it is beheld that various classes of faults exist, namely permanent, transient and intermittent faults. A fault is basically a deviation from its planned function in a hard- or software. Some fault models are widely used in fault injection processes, e.g. stuck at model, bit flip model, saboteur and mutant model. These models are used to validate the dependability of fault tolerant critical systems. These faults are injected in the synthesizable design or in the code to attain reliability and dependability analysis.

4.1 Stuck at fault model

In this fault model, a logic value (0/1) is allocated to a signal line in the logic circuit. Two faults per line can occur, these are stuck at 1 (sa-1) or stuck at 0 (sa-0) at the input or the output of a logic gate [17]. The simplest form of this model is an OR gate with one input fault injection stuck at signal (fi_sa). When this signal becomes ‘1’, the output of this gate will be locked to ‘1’, no matter what the value of other input/s will be. Hence the output of that line in circuit is stuck at ‘1’. Similarly, for stuck at 0, the simplest model is an AND gate with the inverted fi_sa signal. Figures 2 and 3 show the stuck at 1 and 0 models in their simplest form respectively [18].

When the instrumentation based technique is used in FPGA based tools, these stuck at fault models are inserted in the code. A FPGA can be programmed using HDL languages VHDL and Verilog. In this paper, the Verilog language is considered.

At behavioural level, this fault model can be described as,

```
module stuck_at_1 (A,B,fi_sa,C);
  input A, B, fi_sa;
  output C;
  assign C = (A | fi_sa) & B;
endmodule
```

The code example shows that whenever fi_sa gets the value 1, then value of A is stuck at 1 which leads to an output of C=B, hence a fault is injected. This stuck at model is used to find dependability parameters like fault coverage, fault latency and fault propagation.

4.2 Bit flip fault model

The bit flip fault model is also widely used in order to calculate SEU (Single Event Upset) soft errors. A SEU occurs when a bit is changed from ‘0’ logic to ‘1’ logic and vice versa. This model can be implemented in different ways. In [18] the bit flip model is designed by using a XOR gate with the one fi_bf input, which inverts the bit when activated. The simplest model is shown in Figure 4.

```
module bit_flip (A, fi_bf, C);
  input A, fi_bf;
  output C;
  assign C = A ^ fi_bf;
endmodule
```

Fig. 4 Bit flip model (XOR gate) [18]

If the output is required to be stored for further processing a FF must be added to store the bit value. Hence the model given above is deployed as shown in Figure 5 [18],

```
input K, fi_bf; output Q;
 FF K Q
```

Fig. 5 Bit flip model utilization in circuit [18]

In some other literature, this multiplexer is not used and the output is directly fed to the D-FF, as shown in Figure 6 [19].
The bit flip model in Verilog HDL can be represented as:

```
Module bitflip (a,b,fi_bf,c)
  input a,b,fi_bf;
  output c;
  assign c = fi_bf ^ (a & b);
endmodule
```

When the fi_bf (Fault injection_bitflip) signal is low (inactive), the combinational output is assigned to C, which is the correct result and when the fi_bf signal is high, the value changes to its opposite state.

4.3 Mutant fault model
In the Mutant technique, a component is replaced completely by another component. While in an inactive phase this behaves as the original circuit component and when it is activated it injects faults. Mutants can be created by replacing a logical function with a conditional if-else and case statement in HDL code [20]. Here a simple example is given to illustrate the mutant process in a code.

```
if (~ fi_mutant)
c= a+b;
else
c= a- b;
end
```

4.4 Saboteur fault model
In saboteur techniques, a special HDL component is added to the original circuit. It remains inactive during the normal operation, but when it is activated it injects faults during the fault injection process. The basic and simplest saboteur model could be the stuck at model. It can be a single logic gate or complex combinational or sequential logic circuit [21].

There are some more techniques e.g. scan chain, BIST can also be used in FPGA based fault injection tools. Above methods target FPGA devices and FPGA cores in SoC implementation. Dependability analysis, fault tolerance and verification can be carried out by injecting physical faults and emulating them on FPGAs [22]. A few more approaches to emulate faults in FPGA based systems are Memory Stub logic, Bus Stub Logic, Mask logic and Fault Injection Bus [6].

5 Proposed Approach
From literature survey it is beheld that no fault injection tool covers all types of fault models. Tools that cover multiple models do not give the user the opportunity to select a particular one. In addition, some of the tools are technology dependent and some are very time consuming in order to perform the experiment. Keeping these points in mind, we are proposing the new FPGA based fault injection method using the instrumentation technique. In this technique four fault models are proposed. All faults are injected in the code during the design phase. The block diagram for our proposed approach is shown in Figure 9. The operation of each block is described below.

5.1 Target system (SUT)
The system under test (SUT) can be any synthesizable combinational, sequential circuit, SoC or any processor based system. This module consists of one fault free model (now and onwards called as Golden model) and three copies of the golden models, in which faults are injected using instrumentation. All modules are triggered with the same inputs and control pins from other blocks. Faults can be injected in primary inputs, internal lines and primary outputs of each module in the target circuit. The output responses from all faulty systems are compared with the fault free in order to detect the faults.

5.2 Fault injection controlling unit (FICU)
This unit will reside on the FPGA chip with the target system on board. Faults injected might be bit flip, stuck at 0/1, mutants and saboteurs. This unit will decide which fault model will be activated. The user of this tool is also able to control the injection (automatic or manual) by the Automatic Fault Selection (AFS) switch. This unit activates the fault selection and activation unit (FSAU).
5.3 Fault activation & selection unit (FASU)
This unit will also reside with the target system on the FPGA board. When a particular fault model is activated, this unit will activate the faults one by one for every input present at the input lines. Three faults will be injected and checked at a time and the responses are compared so that the faults can be detected.

5.4 Memory unit (MU)
We introduce a separate memory unit to store different observation and check points. This unit will also be implemented on the FPGA board. The idea is to store the different output values from the fault free and faulty systems. The stored data can be used for further calculations by the fault analyser unit available on the host pc. After execution, the stored information will be transferred to the host PC via UART port.

![Fig. 9 Block diagram of proposed FI tool](image)

5.5 Result/fault Analyser
This unit will be implemented on the host PC, and will be written in C, MATLAB or some other language to analyse the data transferred through the Memory Unit for further analysis. This unit receives the inputs which cause deviation in the outputs and the detected faults. It will determine silent faults, latent faults and detected faults in order to calculate the fault coverage. The FC can be calculated as,

\[
\text{Fault Coverage} = \frac{F_D}{F_T} \times 100\% \tag{1}
\]

Where,
- \(F_D\) = Number of faults detected
- \(F_T\) = Total number of faults injected/considered

6 Conclusion
In this paper, we conferred on a detail study about the FPGA based fault injection tools and techniques. The use of a FPGA enables fault injection tools to be developed with a variety of techniques. Instrumentation based techniques are focused throughout the paper and tools and techniques that have been developed so far are illustrated. The purpose and objective of fault injection tools is to inject faults and match the faulty response with the fault free. In instrumentation based techniques additional circuits are added in order to inject faults. The most commonly used fault models are presented. Finally, we proposed a new technique to automate the fault injection process and it is shown how all fault models can be applied for SUT. The fault models are added to the target system and inject faults, when activated. Numbers of faults for each fault model are injected. The faults are activated one at a time with same fault selection lines. Hence no extra input lines are required. Three faults can be processed at a time, which speeds up the process. The fault injection tool needs to be developed for analysing each fault model and calculate different dependability parameters.

References:


