Evaluation of the Thor Microprocessor Using Scan-chain-Based and Simulation-Based Fault-Injection

Peter Folkesson, Sven Svensson and Johan Karlsson
Chalmers University of Technology
Göteborg, Sweden

Joakim Ohlsson
SAAB Ericsson Space AB
Göteborg, Sweden

EXTENDED ABSTRACT

Submitted to EWDC-8, 1997

Fault injection has become an established method for testing and evaluating the fault handling capabilities of fault-tolerant and fail-safe systems [1]. Techniques for fault injection fall into two categories.

- fault injection in software simulation models of systems, and
- fault injection in physical systems, i.e. either prototypes or actual systems

These two categories complement each other as they are used in different phases of the design process. The advantage of simulation-based fault injection is that it can be employed early in the design, which facilitates early detection of design faults, which thus reduces the cost for correcting such faults. It also provides a high degree of controllability and observability. The main drawback to simulation-based fault injection is the time overhead involved in simulations, which puts practical limitations on the amount hardware and system activity that can be simulated.

Fault injection in physical systems is important because it tests the actual implementation of fault handling mechanisms. However, techniques for injecting faults in physical systems, such as pin-level fault injection or software implemented fault injection provides limited controllability and observability. Also, these techniques may not be able emulate the effects of all faults since they suffer from a lack of physical reachability.

One way of improving reachability as well as observability and controllability in physical fault injection experiments is to use the built-in testing logic present in many modern VLSI circuits.

We here describe on-going work aiming towards an extensive evaluation of the effectiveness of the error detection mechanisms included in the Thor microprocessor [3]. To this end, we are developing a tool called FIMBUL for fault injection via the microprocessor's test access port (TAP). We are also using a detailed VHDL model of Thor to do fault injection using the MEFISTO-tool [2].

The Thor microprocessor is designed by SAAB Ericsson Space, Göteborg. It is a 32-bit RISC based on a stack-oriented instruction set architecture and is intended for embedded real-time applications with high requirements on reliability. The support for real-time processing and specifically for the Ada language includes: on-chip task handling and scheduling, rendezvous, Ada exceptions and time handling, accurate delays and fast interrupt handling. The error detection support implemented in Thor includes: double error detection/single error correction of memory transfers, parity on address bus, control flow checking, stack limit checks, exceptions and master/slave operation. Thor is equipped with a Test Access Port (TAP) with functionality according to the IEEE standard 1149.1. This port enables access to boundary, internal and debug scan registers that are used to perform fault-injection.

The major goal of this work is to measure the coverage and latency of the error detection mechanisms included in Thor. Another goal is to compare the effectiveness of built-in physical fault injection and simulation-based fault injection. Which of the two methods are most cost-effective? Are the methods really complementary, or could the essential results be obtained using only one of the methods? A third goal is to develop a tool that analyses the hardware usage during program execution so that the injection of faults that do not affect the operation of the microprocessor can be avoided.

The FIMBUL tool

FIMBUL (Fault Injection and Monitoring using BUilt in Logic) is a tool that uses the TAP facilities of the Thor CPU to do fault-injection, i.e. built in fault-injection. Transient faults can be injected into any of the locations accessible by the Boundary and Internal Scan Registers of the Thor CPU using the bit-flip fault model. The points in time when fault-injection should occur can be chosen by setting break-points using the Thor Debug Scan Register. Fault-injection is triggered either when a specific address has been read or written to *n* times or when the PC (Program Counter) has had the same value as a specific address *n* times. The selection of when and where to fault-inject can be made either randomly or non-randomly within the limitations of the Thor hardware.

Figure 1 shows an overview of FIMBUL. In the *set-up phase*, the chosen workload is analysed either manually or by an analysis tool in order to create the experimental control file (ECF). The ECF tells FIMBUL when and where to fault-inject, the number of times fault-injection should occur as well as the timeout value that is used for restarting the experiment if no error occurred after fault-injection.

In the *fault-injection phase*, the ECF is read and interpreted by FIMBUL. The workload is downloaded to the Thor Target System (which consists of a Thor Evaluation Board installed on a Sun workstation). A break-point is then set according to the information given in the ECF and execution of the workload is started. When the break-point has been reached the number of times that is stated in the ECF, the workload stops executing and fault-injection takes place. Fault-injection is made by reading the contents of the Boundary and Internal Scan Registers of the Thor CPU, inverting the

bits stated in the ECF and then writing back the fault-injected scan registers to Thor. The workload then starts executing from where it left off until an error occurs or the timeout value given in the ECF is reached, whichever comes first.

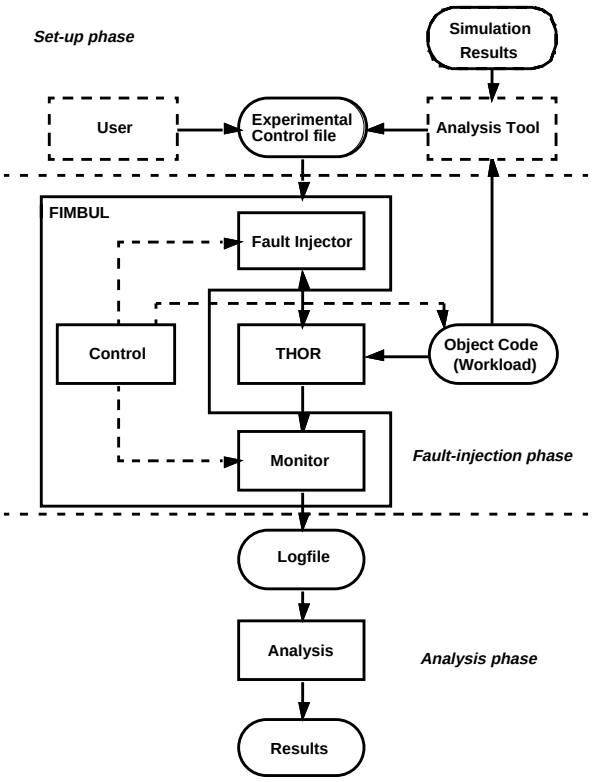

Figure 1. FIMBUL overview

The contents of the Thor scan registers and Thor Evaluation Board memory are then read and logged into a file. Information logged also include when and where faults were injected and whether or not a timeout occurred. The Thor Target System is then reset and another experimental run begins (if stated in the ECF).

In the *analysis phase*, the log file is analysed and results about the effectiveness of the Thor error detection mechanisms are derived (e.g. total error detection coverage and the distribution of errors detected by the various error detection mechanisms).

The MEFISTO tool

MEFISTO (Multi-level Error and Fault Injection Simulation TOol) is a tool for injection of simulated faults into VHDL models on various levels of abstraction. With the tool it is possible to inject a wide variety of faults, e.g. permanent or temporary stuck-at-faults, bit-flips and user-defined functional faults. A fault-injection campaign using MEFISTO can be divided into three phases: *set-up*, *simulation* and *data processing*.

The *set-up phase* involves selecting VHDL objects and applying selected fault-models to the objects. The purpose of this operation is to build up a fault-set. Also in the set-up phase experiments are defined by choosing faults from the fault-set and applying the faults to an activation time.

The *simulation phase* starts with a reference simulation run and then the experiments in the campaign are scheduled and run on a number of workstations. The need for simulation of the time interval from start until a fault is activated is highly reduced by using check-pointing. That is, MEFISTO restores the state of the simulator with the reference check-point having a time stamp that is immediately before the fault activation time.

After the fault-injection campaign is finished the *data processing phase* follows. During this phase the data can be analysed with respect to some property e.g. error detection coverage or error detection latency.

Current status of work

Currently, the FIMBUL tool is able to inject transient bit-flip faults in all locations accessible by the TAP, i.e. at about 3000 internal locations of the Thor CPU. The possibility of implementing injection of permanent faults is under investigation.

The work on simulation based fault-injection has so far resulted in the run of a pilot-campaign. This campaign has generated simulation specific data that will be used to properly set parameters of the MEFISTO-tool. Under preparation is a valid campaign, wherein bit-flip faults will be injected in all memory elements (e.g. flip-flops and registers) of the Thor microprocessor core (about 3500 locations).

A workload that will be executed on Thor in the fault-injection campaigns has also been designed. It represents an application suitable for the processor: a digital state model control algorithm. The workload is based on a repeated program sequence wherein Thor receives state data from a simulation model of the controlled object, calculates control data and closes the control loop by sending the control data to the plant model. By using a simulation model of the object, an almost authentic representation of the state data can be achieved. This will result in an execution of the Thor workload as if the processor had been situated in a real system.

References

[1] Iyer, R.K., “Experimental Evaluation,” Special Issue of Proc. 25th Int. Symp. on Fault-Tolerant Computing (FTCS-25), Pasadena, CA, USA, 1995.

[2] Jenn E., Arlat J., Rimén M., Ohlsson J., Karlsson J., “Fault Injection into VHDL Models: The MEFISTO Tool”, in *Proc. 24th Annual IEEE International Symposium on Fault-Tolerant Computing*, FTCS-24, pp. 66-75, Austin, TX, USA, June 1994.

[3] SAAB Ericsson Space, *Microprocessor Thor, Product Information*, September 1993.