Fast Fault Emulation for Synchronous Sequential Circuits

Jaan Raik, Peeter Ellervee, Valentin Tihhomirov, Raimund Ubar

Department of Computer Engineering, Tallinn University of Technology, Raja 15, 12618 Tallinn, Estonia,
e-mail: jaan@pld.ttu.ee, phone: +372 620 2252, fax: +372 620 2253

ABSTRACT: *Current paper presents an approach to emulate fault simulation of sequential circuits on FPGA. Fault simulation is an important subtask in test pattern generation and it is frequently used throughout the test generation process. In the paper, we explain the problems associated to fault emulation for sequential circuits. Two alternative approaches are described, which can be considered as trade-offs in terms of required FPGA resources and fault grading accuracy. In addition, an environment for reconfigurable hardware emulation of fault simulation is proposed. Experiments show that it is beneficial to use emulation for circuits/methods that require large numbers of test vectors, e.g. simulation-based test pattern generation or validation.*

Keywords: *Fault emulation, sequential circuits, FPGAs*

1 Introduction

The increasing complexity of modern VLSI circuits has made test generation one of the most complicated and time-consuming problems in the domain of digital design. As the sizes of circuits grow, so do the test costs [1]. Test costs include not only the time and resources spent for testing a circuit but also time and resources spent to generate suitable test vectors. The most important sub-task of any test generation approach is the suitability analysis of a given set of test vectors. There exist many techniques to perform such an analysis. Circuit structure analysis gives good results but it is rather time consuming.

Fault simulation is the most often used way of analysis and there exist many techniques exist to speed up simulation (see, e.g., [2]). Efficient fault simulation algorithms for combinational circuits are known already for decades. However, it is the large sequential designs whose fault grading run times could be measued in years that drive the need faster implementation, e.g. by hardware emulation. At the same time, reconfigurable hardware, e.g., FPGAs, has been found useful as system-modeling environments [3]. This has been made possible by the availability of multi-million-gate FPGAs. For academic purposes, cheaper devices with rather large capacity, e.g., new Spartan devices, can be used.

The availability of large devices allows implementing not only the circuit under test with fault models but also test vector generator and result analysis circuits on a single reconfigurable device. In addition to merely increasing the speed of fault simulation, the idea proposed in this paper can be used for selecting optimal Built-In Self-Test (BIST) structures. To study the possibility of replacing fault simulation with emulation, we first had to solve some essential issues - how to represent logic faults in a synthesizable circuit, how to feed the test vectors into the circuit, and how to read and/or analyze the results of emulation?

Then we created the experimental environment, and finally performed experiments with some benchmark circuits. The experiments showed that for circuits and/or applications that require large numbers of test vectors, it is beneficial to replace simulation with emulation. More work is needed to integrate the hardware part with the software part of the test generation environment.

The paper is organized as follows. Section 2 provides an overview of previous related works. Section 3 presents two alternative approaches to sequential fault emulation, which can be considered as trade-offs in terms of required FPGA resources and fault grading accuracy. The emulation approach implemented in current paper is planned to be used in cooperation with diagnostic software Turbo Tester, described in Section 4. The emulation environment is introduced in Section 5. In Section 6, the results of experiments are presented and Section 7 is dedicated for conclusions.

2 Overview of Related Works

A number of works on fault emulation for combinational circuits has been published in the past. They rely either on fault injection (see, e.g., [4, 5]) or on implementing specific fault simulation algorithms in hardware [6]. Recently, acceleration of combinational circuit fault diagnosis using FPGAs has been proposed in [7]. However, the need for hardware fault emulation has been driven mainly by large sequential designs whose fault grading run-times could extend to several years.

In many of the papers for sequential circuits, faults are injected either by modifying the configuration bitstream while the latter is being loaded into the device [8] or by using partial reconfiguration [9, 10, 11]. This kind of approach is slow due to the run-time overhead required

by multiple reconfigurations. Other options for fault injection are shift-registers and/or decoders (used in this paper). A paper relying on the shift-register based method was presented in [12]. Shift-registers are known to require slightly less hardware overhead than the decoders do. However, in [12] injection codes and test patterns are read remotely from a host PC.

In addition to merely increasing the speed of fault simulation, the idea proposed in current paper can be used for selecting optimal Built-In Self-Test (BIST) structures. In an earlier paper [13] a fault emulation method to be used for evaluating the Circular Self-Test Path (CSTP) type BIST architectures has been presented. Different from the current approach, no fault collapsing was carried out and fault-injecting hardware was inserted to each logic gate of the circuit to be emulated.

In current paper, we propose an efficient FPGA-based fault emulation environment for sequential circuits. The environment has interfaces to the digital test package Turbo Tester [14, 15] developed at Tallinn University of Technology and described in Section 3. The main novelty of the emulation approach lies in implementing MUXes and a decoder for fault injection which, unlike the shift register based injection, allows to insert faults in arbitrary order. This feature is highly useful when applying the presented environment in emulating the test generation process. In addition, we use an on-chip input pattern generator as opposed to loading the simulation stimuli from the host computer. It is also important to note that the time spent for emulator synthesis is included to the experimental results presented in Section 5.

3 Sequential Fault Emulation Approaches

Fault emulation approaches for sequential circuits encounter two major problems that are not present in combinational fault emulation:

1) The sequential fault emulation process takes place as follows. First, faultfree emulation of the circuit is performed. Then, faults are injected one-by-one and for each of them the test stimuli set is emulated. This repetitive emulation requires the circuit to be in an initial state at the beginning of every subsequent emulation of the test set.

2) The environment presented in Section 5 of this paper could be planned to be used with signature analysis (e.g. a MISR) for speeding up the fault response analysis. In that case, all the issues that are related to signature analysis in sequential BIST apply to fault emulation too. In other words, we have to solve the problem of compressing don't care responses to a meaningful signature.

In the following we present two alternative approaches to sequential fault emulation, which contend the above two issues. The alternatives can be considered as trade-offs in terms of required FPGA resources and fault grading accuracy.

3.1 Fault emulation with all registers resettable

It is possible to get rid of both of the problems mentioned above if all the registers in the design are made resettable and the global reset signal is applied at the beginning of the test set. In the general case, not all the registers in sequential designs could be reset. There are two possibilities to solve this issue. First, (and this is the approach used in current paper), it is possible to alter the initial design to contain resettable registers only. This can be considered as a kind of design-for-testability modification. However, this might not be desireable as the circuit's silicon area would increase.

Another option is to modify the circuit only when emulated on the FPGA without modifying the final design itself. The drawback here is that using this approach we cannot compare the results directly to the ones of software fault simulation. However, it is a reasonable approach if fast fault grading is needed with limited FPGA resources available.

3.2 Encoded don't care value approach

An alternative solution for the problem is to rely on the encoded don't care value approach. It is based on coding the three-valued logic: 0, 1, X. To code these values we need two bits and thus, two instances of the emulating circuit. 0 is coded as (1,0), 1 as (0,1) and X as (0,0). (Additionally, it is possible to code the third-state high impedance value Z as (1,1)). Table 1 shows the logic used for evaluating the basic gates AND, OR and inverter. T?he gates have inputs A, B and an output Y. The lower indexes are for indexing the coding bits.

Table 1. Four-valued gate evaluation

	Y_0	Y_1
AND	$A_0 \mid B_0$	$A_1 \& B_1$
OR	$A_0 \& B_0$	$A_1 \mid B_1$
INV	A_1	A_0

Similar approach has been used in many applications, including in the parallel fault simulation method PROOFS [17]. An obvious shortcoming is the duplication of hardware necessary for emulation. The main advantage lies in the fact that the fault emulation results directly match the software simulation.

4 Test Package Turbo Tester

Turbo Tester (TT) is a test software package developed at the Department of Computer Engineering of Tallinn University of Technology [14, 15]. The TT software consists of the following test tools: test generation by different algorithms (deterministic, random and genetic), test set optimization, fault simulation for combinational

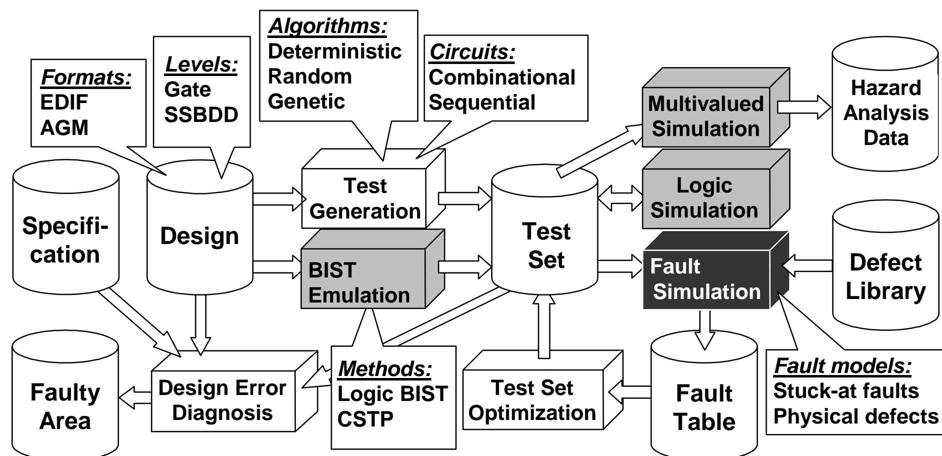

Fig.1. Turbo Tester environment

and sequential circuits, testability analysis and fault diagnosis. It includes test generators, logic and fault simulators, a test optimizer, a module for hazard analysis, BIST architecture simulators, design verification and design error diagnosis tools (see Fig. 1). TT can read the schematic entries of various contemporary VLSI CAD tools that makes it independent of the existing design environment. Turbo Tester versions are available for MS Windows, Linux, and Solaris operating systems.

The main advantage of TT is that different methods and algorithms for various test problems have been implemented and can be investigated separately of each other or working together in different work flows.

Model synthesis. All the tools of TT use Structurally Synthesized BDD (SSBDD) models as an internal model representation. TT includes a design interface generating SSBDD-s in AGM format from EDIF netlists. The set of supported technology libraries can be easily extended.

Test generation. For automatic test pattern generation (ATPG), random, deterministic and genetic test pattern generators (TPG) have been implemented. Mixed TPG strategies based on different methods can also be investigated. Tests can be generated both for combinational and sequential circuits.

Test pattern analysis. There are concurrent and parallel fault simulation methods implemented in the system. In current paper, we have experimented only with "Fault Simulation" part (black in Fig. 1). In the future, also the other simulation-related parts might be considered (gray in Fig. 1).

Test set optimization. The tool is based on static compaction approach, i.e. it minimizes the number of test patterns in the test set without compromising the fault coverage.

Multivalued simulation. In Turbo Tester, multi-valued simulation is applied to model possible hazards that can occur in logic circuits. The dynamic behavior of a logic network during one single transition period can be described by a representative waveform on the output or simply by a corresponding logic value.

Design error diagnosis. After a digital system has been designed according to its specification, it might go through a refinement process in order to be consistent with certain design requirements, e.g., timing specifications. The changes introduced by this step may lead to undesired functional inconsistencies compared to the original design. Such design errors should be identified via verification.

Evaluation of Built-In Self-Test (BIST) quality. The BIST approach is represented by applications for simulating logic BIST and Circular Self-Test Path (CSTP) architectures.

5 Emulation environment

The emulation environment was created keeping in mind that the main purpose was to evaluate the feasibility of replacing fault simulation with emulation. Based on that, the main focus was put on how to implement circuits to be tested on FPGAs. Less attention was paid how to organize data exchange between hardware and TT. For the first series of experiments, we looked at combinational circuits only. Results of experiments with combinational circuits were presented in [5].

For sequential circuits, most of the solutions used for combinational circuits could be exploited. The main modification was an extra loop in the controller because sequential circuits require not a single input combination but a sequence consisting of tens or even hundreds of input combinations. Also, instead of hard-coded test sequence generators and output analyzers, loadable modules were introduced. Before building the

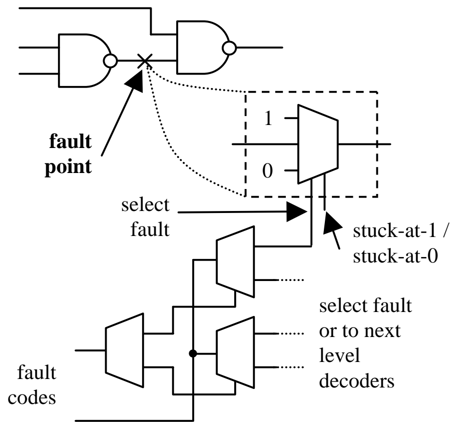

Fig. 2. Fault point insertion with multiplexer

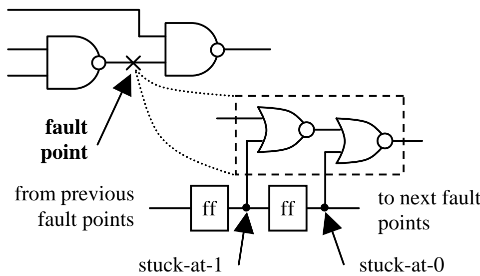

Fig. 3. Fault point insertion with shift-register

experimental environment, we had first to solve how to insert faults, how to generate test vectors, how to analyze output data, and how to automate design flow. The solutions and discussions are presented below.

Fault insertion: The main problem here was how to represent non-logic features - faults - in such a way that they can be synthesized using standard logic synthesis tools. Since most of the analysis is done using stuck-at-one and stuck-at-zero fault models, the solution was obvious - use multiplexers at fault points to introduce logic one or zero, or pass through intact logic value. Also, since a single fault is analyzed at a time typically, decoders were introduced to activate faults (see Fig.2).

The extra multiplexers will increase gate count and will make the circuit slower (typically 5 to 10 times). It is not a problem for smaller circuits but may be too prohibitive for larger designs - the circuit may not fit into target FPGA. A solution is to insert faults selectively. Selection algorithm, essentially fault set partitioning, is a subject of future research.

Compared against shift-register based fault injection approaches (see, e.g., [12] and Fig. 3), the use of multiplexers has both advantages and disadvantages. The main disadvantage is small increase in both area and delay of the circuit. Although the delay increase is only few percents, execution time may increase significantly

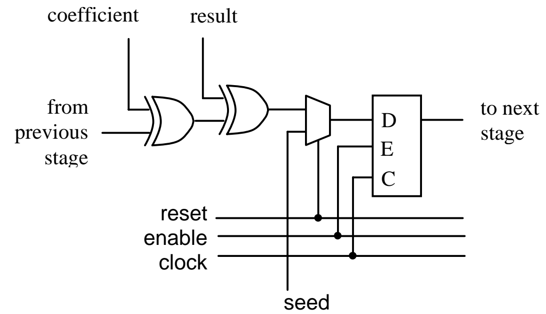

Fig. 4. Single stage of LFSRs

for long test cycles. The main advantage is that any fault can be selected in a single clock cycle, i.e., there is no need to shift the code of a fault into the proper register. Combining both approaches may be the best solution and one direction of future work will go in that direction.

Test vector generation and output data analysis: Here we relied on a well-known solution for BIST - Linear Feedback Shift Register (LFSR) is used both for input vector generation and output correctness analysis (see, e.g., [16]). LFSRs structures are thoroughly studied and their implementation in hardware is very simple. This simplifies data exchange with the software part - only seed and feedback polynomial vectors are needed to get a desired behavior. Output correctness analysis hardware needs first to store the expected output signature and then to report to the software part whether the modeled fault was detected or not. Fig. 4 illustrates a stage of used LFSRs. The input 'coefficient' is used for feedback polynomial. The input 'result' is used only for result analysis and is connected to zero for input vector generation.

Design flow automation was rather easy because of the modular structure of the hardware part. All modules are written in VHDL that allows to parameterize design units (see also Fig. 5):

- CUT - circuit under test, generated by the fault insertion program;
- CUT-pkg and CUT-top - parameters of CUT and wrapper for CUT to interface with the generic test environment, generated by wrapper program;
- Two LFSRs - one for test vector generator and one for output signature calculation (generic VHDL module);
- Three counters - one to count test vectors, one to count test sequences (not used for combinational units), and one to count modeled faults (generic VHDL module);
- Test bench with controller (FSM) to connect all sub-modules, to initialize LFSRs and counters, and to organize data exchange with the external interface; a generic VHDL module; and

- Interface to organize data exchange between the test bench (FPGA) and the software part (PC).

The interface is currently implemented only in part as further studies are needed to define data exchange protocols between hardware and software parts. In future, any suitable FPGA board can be used assuming that supporting interfaces have been developed.

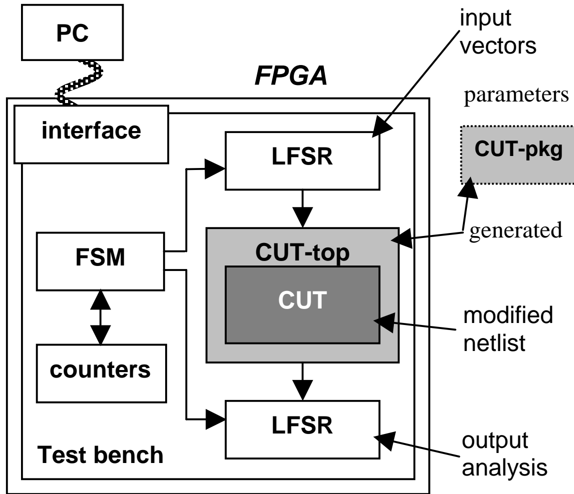

Fig. 5. Emulation environment structure

6 Experimental Results

For experiments, two FPGA boards were used - a relatively cheap XSA-100 board with Spartan2 chip (600 CLBs) and a powerful RC1000-PPE board with VirtexE chip (9600 CLBs). The first one is good for small experiments and to test principles of the environment.

Test circuits were selected from ISCAS'89 and HLSynt'92 benchmark sets to evaluate the speedup when replacing fault simulation with emulation on FPGA. Results of some benchmarks are presented in the paper to illustrate gains and losses of our approach (see Table 2). Columns "# of gates" and "delay" illustrate the parameters of circuits without inserted faults. The "FPGA" columns illustrate the of the whole test bench. Columns "#I", "#O", "#ff", and "#F" represent the number of, outputs, flip-flops, and fault points, respectively. The columns "# of vectors" illustrate the complexity of test. The column "SW" gives the fault simulation time based on a parallel algorithm and " emul" emulation time for the same set of test vectors. Synthesis times have been added for comparison ("synt").

For different benchmarks, the hardware emulation was in average 17.8 (ranging from 6.7 to 53.4) times faster than the software fault simulation. It should be noted that when considering also synthesis times, it might not be useful to replace simulation with emulation, especially for smaller designs. Nevertheless, taking into account that sequential circuits, as opposed to combinational ones, have much longer test sequences, the use of emulation will pay off. Future research will mainly focus on test generation for sequential circuits using genetic algorithms.

7 Conclusions

Current paper proposes two alternative approaches to fault emulation for synchronous sequential circuits: a fully resettable design approach and an encoded don't care approach. An FPGA-based emulation environment implementing the former has been realized with an on-chip test pattern generator (LFSR) and output response compression (MISR). Experiments with HLSynth92 and ISCAS89 benchmarks have been carried out.

The experiments showed that for circuits that require large numbers of test vectors, e.g., sequential circuits, it is beneficial to replace simulation with emulation. Although even for combinational circuits the simulation speedup is significant, there exist rather large penalty caused by synthesis time. Based on that, it can be concluded that the most useful application would be to explore test generation and analysis architectures based on easily reprogrammed structures, e.g., LFSRs. This makes fault emulation very useful to select the best generator/analyzer structures for BIST. Another useful application of fault emulation would be genetic algorithms of test pattern generation where also large numbers of test vectors are analyzed. Future work includes development of more advanced on chip test vector generators and analyzers.

Table 2. Results of experiments

circuit	# of gates	delay [ns]	FPGA		#I	#O	#ff	#F	# of vectors		SW	HW	
			CLBs	MHz					# of seq.	seq. len.	simul	synt	emul
s5378	4933	21.8	2583	10	35	49	179	2517	80	100	26.8”	22’	4.0”
s15850	17.1k	66.8	6125	5	77	150	534	6076	200	200	15.6’	55’	97”
prefetch (32-bit)	1698	20.0	941	25	66	96	128	923	40	400	9.46”	5.1’	1.19”
diff-eq (16-bit)	4562	25.7	4672	5	80	48	115	4789	20	200	87.9”	45’	7.7”
TLC	290	9.5	215	50	3	6	17	196	40	100	2.69”	1.2’	0.05”

Acknowledgments:
This work was supported partly by Estonian Science Foundation grants No 5601 and 5637 and by European projects IST-2001-37592 "E-VIKINGS II" and IST-2000-30193 "REASON".

References

[1] ITRS 2001 roadmap

[2] P. McGeer, K. McMillan, A. Saldanha, A. Sangiovanni-Vincetelli, P. Scaglia, "Fast Discrete Function Evaluation Using Decision Diagrams." In Proc. of ICCAD"95, pp.402-407, Nov. 1995.

[3] "Axis Systems Uses World's Largest FPGAs from Xilinx to Deliver Most Efficient Verification System in the Industry." Xilinx Press Release #0273 - http://www.xilinx.com/

[4] R. Sedaghat-Maman, E. Barke, "A New Approach to Fault Emulation." In Proc. of Rapid System Prototyping, pp.173-179, June 1997.

[5] P. Ellervee, J. Raik, V. Tihhomirov, "Fault Emulation on FPGA: A Feasibility Study." In Proc. of Norchip"03, Riga, Latvia, Nov. 11-12, 2003.

[6] M. Abramovici, P. Menon, "Fault Simulation on Reconfigurable Hardware." In Proc. of FPGAs for Custom Computing Machines, pp.182-190, April 1997.

[7] S.-K. Lu, J.-L. Chen, C.-W. Wu, W.-F. Chang, S.-Y. Huang, "Combinational Circuit Fault Diagnosis Using Logic Emulation." 2003.

[8] M. Alderighi, S. D'Angelo, M. Mancini, G.R. Sechi, "A Fault Injection Tool for SRAM-based FPGAs." In Proc. of 9th IEEE International On-Line Testing Symposium (IOLTS'03), pp.129-133, July 2003.

[9] R. Wieler, Z. Zhang, R. D. McLeod, "Simulating Static and Dynamic Faults in BIST Structures with a FPGA Based Emulator." In Proc. of FPL'94, Springer-Verlag, pp.240-250, Sept. 1994.

[10] K.-T. Cheng, S.-Y. Huang, W.-J. Dai, "Fault emulation: a new approach to fault grading." In Proc. of ICCAD'95, pp.681-686, Nov. 1995.

[11] L. Burgun, F. Reblewski, G. Fenelon, J. Barbier, O. Lepape, "Serial fault emulation." In Proc. DAC'96, pp.801-806, Las Vegas, USA, June 1996.

[12] S.-A. Hwang, J.-H. Hong, C.-W. Wu, "Sequential Circuit Fault Simulation Using Logic Emulation." In IEEE Trans. on CAD of Int. Circuits and Systems, Vol.17, No.8, pp.724-736, Aug. 1998.

[13] R. Wieler, Z. Zhang, R. D. McLeod, "Emulating Static Faults Using a Xilinx Based Emulator." In Proc. of IEEE Symposium on FPGAs for Custom Computing Machines, pp.110-115, April 1995.

[14] G. Jervan, A. Markus, P. Paomets, J. Raik, R. Ubar, "Turbo Tester: A CAD System for Teaching Digital Test." In Microelectronics Education, Kluwer Academic Publishers, pp.287-290, 1998.

[15] "Turbo Tester" home page - URL: http://www.pld.ttu.ee/tt

[16] D.K. Pradhan, C. Liu, K. Chakraborty, "EBIST: A Novel Test Generator with Built in Fault Detection Capability." In Proc. of DATE'03, pp. 224-229, Munich, Germany, March 2003.

[17] T. M. Niermann, W.-T. Cheng, J. H. Patel, "PROOFS: a Fast, Memory Efficient Sequential Circuit Fault Simulator". Proc. DAC, pp. 535-540, 1990.