POWER ANALYSIS AND DIAGNOSIS OF FAULTS IN VLSI CIRCUITS

B.K.V. Prasad\textsuperscript{1}, Dr. T. Madhu\textsuperscript{2}

ECE Dept.
\textsuperscript{1}ASRCOE, \textsuperscript{2}SIET
Tanuku, \textsuperscript{2}Narasapur, India
\textsuperscript{1}baditakali@gmail.com, \textsuperscript{2}tennetimadhu@yahoo.com.

Dr. P. Satish Kumar\textsuperscript{3}, Dr. B. S. Charles\textsuperscript{4}, Dr. S. Ravi\textsuperscript{5}

ECE Dept.
\textsuperscript{1}CVRCOE, \textsuperscript{2}SSCET, \textsuperscript{3}Dr. MGR University
Hyderabad, \textsuperscript{4}Kurnool, \textsuperscript{5}Chennai, India
\textsuperscript{3}psatish_1968@rediffmail.com, \textsuperscript{5}ravi.mls@yahoo.co.in,
\textsuperscript{4}bstephencharles@rediffmail.com.

Abstract- FPGA based Fault injection and Fault tolerance techniques are used to evaluate and validate the reliability of VLSI circuits. This approach combines the efficiency of hardware based techniques and the flexibility of simulation based techniques. The system efficiency and robustness increases as the reconfiguration of FPGA is not needed for each fault experiment. Fault injection is performed using commercial VHDL simulation tools such as static and dynamic methods. Instrumented 2 bit multiplier circuits were employed to evaluate the fault injection technique. The power analysis results provided for fault free, stuck-at-0 and stuck-at-1 faults in digital circuits validate the point that faulty circuits dissipate more and hence draw more power.

Keyword- FPGA, Fault Identification, Power Dissipation, Stuck-at-faults, VHDL.

I. INTRODUCTION

In recent years, there has been a rapid increase in the use of FPGA technology in areas such as railway traffic control, aircraft traffic control and telecommunications. This trend has led to a growing interest in the validation of the fault tolerance properties and evaluation of their reliability. Simulation-based Fault Injection is very effective in allowing early and detailed analysis of designed systems. The FPGA technology can be effectively exploited not only for rapid prototyping and for small-volume productions, but also in performing Fault Injection experiments. In FPGA, a portion of the device resources is set aside to perform the fault handling. Fault diagnosis includes both fault detection and fault location. An internal fault monitor is run continuously without disturbing device functionality. This internal fault monitor conserves limited-pin resources and avoids the relatively slow process of transferring information off-chip through the pins as shown in Figure.1. This approach allows for rapid detection of both hard and soft faults. The resources needed to perform fault testing can be kept to minimum by using fault-scanning methodology[1].

Only a small section of FPGA is tested at a time, but testing can scan across the FPGA assuring that the entire FPGA will be tested eventually. In FPGA, faults mainly occur in interconnects. These programmable interconnects contain horizontal and vertical single and double length lines intersecting at a box called programmable switch box. Each switch matrix consists of programmable pass transistors and is used to establish connection between the lines. All the interconnections are composed of metal segments with programmable switch points and switching matrices to implement the desired routing. Various faults that occur in FPGA interconnects are

\textbf{Stuck-at-fault}: It assumes that a fault in logic gate results in one of its inputs or the output being fixed to either logic 0 or logic 1. The faulty circuit behaves as a combinational circuit.

\textbf{Bridging fault}: A bridging fault occurs when two leads in a logic network are connected accidentally and “wired logic” is performed at that connection.

\textbf{Stuck-open fault}: These faults are a peculiarity of CMOS digital integrated circuits. The faulty circuit behaves as a sequential circuit [4].
II. CIRCUIT TRANSFORMATIONS IN FAULT INJECTION

The method implements an effective way to trigger the occurrence of faults at the injection time and supporting the observation of the faulty behavior. The architecture is based on adding to the original circuit a register named mask chain which stores the binary information about the flip-flops affected by the fault and ad-hoc logic for performing the fault injection. The signal inject controls the fault injection. The fault injection master asserts it to force the occurrence of the selected bit-flips.

A. Mask Chain:

Each bit in FFs is associated to a bit in Mask Chain, which is a register with parallel- and serial-load capabilities. The load and mode signals control the operation of the register. The register operates in the initialization phase, as a shift-register, and loads a bit stream coming over the scan_in signal. The bit stream contains a 1 in the positions corresponding to the flip flop(s) where the fault must be injected, 0s elsewhere. In the injection time (i.e., when the Fault Injection Master asserts the inject signal), each bit in Mask Chain set to 1 produces a bit-flip, i.e., the corresponding FFs module loads the complement of the value coming from the Combinational Circuitry. The Mask Chain may load the content of the FFs module. The state of the circuit can then be read out by operating the module as a shift register. This feature may be exploited at the end of each experiment to better classify fault effects[2].

B. M:

It is the combinational logic in charge of possibly complementing the output of the Combinational Circuitry which is loaded into the FFs module. The behavior of M depends on the contents of the Mask Chain module and on the inject signal.

C. L:

It is a corrector circuit which is in charge of correcting the error output coming from the flip-flop’s module.

The combinational circuit output can be made to identify the required number of logic circuits. For example if the output of the circuit is 8 bit, then 8 logic circuits are needed. Multiplexer and d-flip-flop combine, act as mask chain which was controlled by load and select (sel) signals. Inject signal controls the fault injection process. The gate level description of the instrumented Flip-flop is shown “Fig. 3”. XOR gate acts as inverter in order to invert the output coming from combinational circuit. The Comparator and XOR gate combinely acts as corrector circuit which corrects the error output coming from FF’s module. The instrumented circuit for 2*2 multiplier is shown in “Fig.4”.

For example by considering “Fig.5” such as 2 * 2 multiplier as combinational circuit. The 4 bit output namely as p3 p2 p1 p0.
D. Operation:
Initially Scan_in, load, clock signals are set to ‘1’ in order to activate FF’s. The Inject signal is set to ‘0’ as no error is introduced into the circuit. When we introduce error in the circuit then set inject to ‘1’. Before the process of injection select scan_in or previous FF’s module output by using select (sel) signal. Let us assume that process is started from the starting bit itself and then 1st scan_in selected is ‘1’, inject signal is ‘1’ so at the AND gate output is ‘1’. The operation of Xor gate combinational output gets inverted and stored in FF’s module. The Comparator compares actual combinational output and error output from FF’s module. The output is ‘1’ if they are different else ‘0’ if they are same. Finally Xor gate corrects the error output coming from FF’s module. The same process will be continued in the next stages, but select the previous flip-flops output by keeping select line as ‘1’. By observing the comparator output, the flip-flops that are affected should be identified. If comparator output is ‘0’ then the flip-flop is not affected by the fault else flip-flop is affected by the fault.

III. RESULTS
A 2*2 multiplier along with instrumented description had been simulated and synthesized. The hardware is tested with FPGA spartran 3. The results are shown in Table I. The delay of the circuit decreased from 10.834 ns to 10.818 ns. The faults were diagnosed. In the multiplier circuit, the errors are injected, corrected and finally matched with the original circuit. Similarly testing is done for 4*4 and 8*8 multiplier, respectively and the results were shown in Table II and Table III respectively. The power dissipation values are calculated for the full adder and the min(x, y) circuits. It is observed that the faulty circuits consume more power than the fault-free circuits as dissipation is more in the former.

<table>
<thead>
<tr>
<th>CONTENTS</th>
<th>2*2 mul without instrumented circuit</th>
<th>2*2 mul with instrumented circuit</th>
<th>DESCRIPTION</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delay (nsec)</td>
<td>10.834</td>
<td>10.818</td>
<td>Speed</td>
<td>0.147%</td>
</tr>
<tr>
<td>Memory usage (K bytes)</td>
<td>52708</td>
<td>53732</td>
<td>Memory</td>
<td>1.9%</td>
</tr>
<tr>
<td>Area</td>
<td>5%</td>
<td>22%</td>
<td>Overhead</td>
<td>77%</td>
</tr>
</tbody>
</table>
From the above tables, it is evident that the execution time of instrumented circuit is faster than the non instrumented circuit. Although the area is a constraint, the instrumented circuits can be implemented in real-time applications where the speed is a main criterion.
Further, it is evident from the full adder and min(x,y) circuits that the power dissipation is more in Stuck-at-1 fault than at Stuck at 0.

IV. CONCLUSION

This paper is an environment for performing Fault Injection campaigns that is based on the adoption of an FPGA device for emulating the system under analysis. A major novelty in the proposed approach lies in the circuit transformations adopted to inject faults in the circuit emulated through the FPGA. These transformations introduce an area overhead that ranges from 19% to 42% for the considered multiplier circuits, but allow a significant reduction in the time required to perform the Fault Injection experiments even with respect to the most effective simulation-based techniques. By exploiting a prototypical version of the described Fault Injection environment, we were able to evaluate its effectiveness with respect to state-of-the-art alternative approaches. Speed-up figures up to 60 have been recorded when our technique is compared with a state-of-the-art gate-level approach. The power dissipation is least when SA0 fault is in the circuit, medium for fault free and highest for SA1 fault. This type of analysis will help in designing a more reliable low power efficient circuitry for automated industry.

REFERENCES