Real-Time BIST for Fault Detection in FPGA

Xiaoming Ju
Software Engineering Institute
East China Normal University
Shanghai, China
xmju@sei.ecnu.edu.cn

Jiehao Zhang
Software Engineering Institute
East China Normal University
Shanghai, China

Yizhong Zhang
Software Engineering Institute
East China Normal University
Shanghai, China
yizhong_zhang@ieee.org

***Abstract*—Real-time fault detections are needed in systems which require high reliability. Some methods like online BIST can detect fault on runtime, but it cannot real-time detect the fault. In this article, three online self-checking methods for built-in fault detection will be presented. These detection methods not only can real-time detect the fault, but can also locate the fault where logic blocks have undergone an SEU by comparing their configuration data. The testing result shows that the method proposed in this article have better performance than that of online BIST**

Keywords-fault detection; real-time; BIST; self-checking; FPGA

I. INTRODUCTION

SEU (Single Event Upset) [1] is one of the most common reasons to cause the functional error of an FPGA. In some system which requires high reliability, only improving the reliability of the FPGA themselves is far from enough. It is not only because it is impossible to prevent the SEU, but it is also not equipped with the ability of self-checking as well. To meet the demand of the systems requiring high dependability, fault detection and fault recovery must be fast enough.

In this article we will present fast online self-checking methods for fault detections in FPGA. These models can detect fault of output signal instantly. The feature of identifying whether the output signal is correct is important, because the output signals may not go wrong immediately even if the logic block went error. Additionally, different from other approaches which only alarm the fault, the methods proposed in this article will locate the fault and tell which blocks have went wrong.

In Section II, we present the related works we have done. Next, three models for fault detection and its basic ideas are presented in Section III. The description for testing and simulation are shown in Section IV. Its result and analysis are discussed in Section V. Finally, a conclusion is drawn in Section VI.

II. RELATED WORK

A variety of fault detection methods have been proposed, one of them is TMR [2,3]. TMR consists of three redundancies. Mismatch of the output of the redundancy indicates an error. However, it cannot locate the fault.

A more widely used approach is to use an online BIST [4, 5] (built-in self-test unit). This method uses a special area called STAR [6, 7] (self-testing area). During the test, BIST will copy the block in the STAR to the spare rows, and recover the connection. Test vectors are generated as inputs to the block under test. Faults are detected if the output of the block under test does not meet expectation. However, during the process, the rows under test must be copied to the spare areas to make the FPGA continuing working. The copying could cost a lot of time, during which the fault outside the STAR would fail to be detected. Another problem is that online BIST requires partial configuration technique which is not support widely by FPGA vendors.

Another method uses two columns called Free Columns and Test Columns [8]. This method is almost based on the identical ideas of online BIST. The detection process still costs a considerable time.

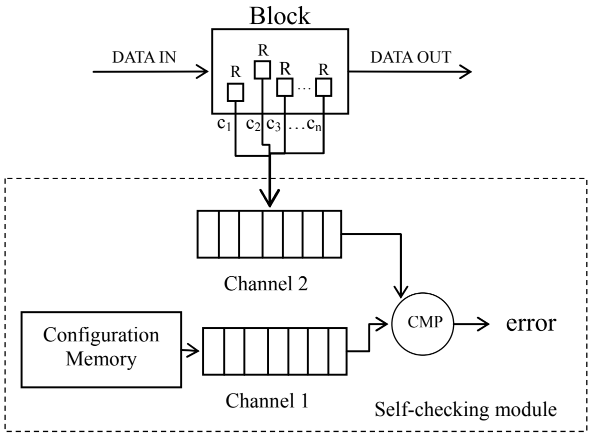
III. PROPOSED METHOD

A. Basic Ideas

Our fault detection model is made of two tunnels inside the FPGA, as showed in Figure 1. Each tunnel can be regarded as a queue. The data that come into Channel 1 are data from a Configuration Memory storing the configuration information of the FPGA, while data into Channel 2 are from the configuration stream of logic blocks at a certain time. Any mismatch between the two channels tells whether the signal is correct and which logic blocks has gone wrong.

The following three models discussed the three different implementations of the channels. Notice that these two channels are always of the same structure in one model, and the detections of fault are always indicated by mismatch of the output of two channels. Model A designs the channels by the structure of pipelines. Model B uses asynchronous FIFO to implement the channels. At last, Model C, the combination of Model A and Model B, is discussed.

These models based on two assumptions. One is that the failures of FPGA are caused only by SEU. Because the transistor errors of FPGA usually caused by electronic migration as an aging problem, they are not discussed in this



R : the flip-flop in the Block
C_i : the configuration bit in the Block

Figure 1: Two channels used for fault detection

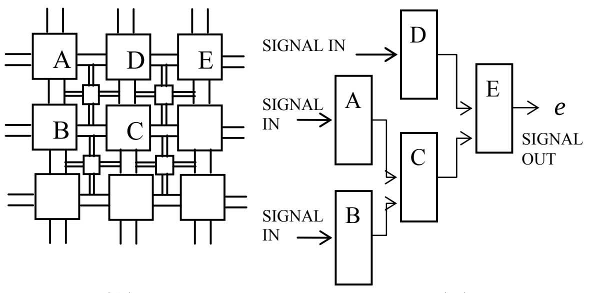


Figure 2: (A) original circuit
(B) Block Tree

paper. A single event upset can caused the flip of some bits in the FPGA. If the SEU happens in the logic blocks, it will cause the function error of the FPGA. An easy method is to compare the current configuration information with the original one in Configuration Memory outside the FPGA. Any mismatch of two data infers a single upset event in logic blocks. That is why two channels are needed for fault detection.

Another assumption is that the flow of the signal in FPGA is deterministic. That is, given by the position of a logic block, we can always trace back to calculate when the output signal went through it. A tree structure called Block Tree is used describe that relationship, as it can clearly describe how the signal is generated. The directed arrows used for connecting two logical blocks show the flow direction of the signal. For example, as showed in Figure 2, it is clear that the output signal e is produced by Block E, whose input signals are output signal of Block D and C, while the input signal of Block C is generated by the output signal of both Block A and Block B.

One of the advantages of the Block Tree is that it can be determined what output signals are still available at the moment when a block is damaged. Consider the case when the Block B is broken. Assume that Block B is broken at time t, and the block in each level will produce a signal at interval T. In this article, the signal at given time τ is donated as e_τ. Then from the Block Tree, it is clear that all the signals in $e_{t \leq \tau \leq t+2T}$ are still valid. And thus, 2T is the period allowed us for fault recovery.

Rule that the closest block from output signal is at the highest level. Then, from Block Tree, a conclusion can be drawn that the lower level the disordered block is at, the more periods it is allowed to repair. It makes difference that it can accurately tell how much time remained for fault recovery.

B. Model A

In Model A, a pipeline is used to implement the channel. S_{e_i} is denoted as the configuration stream of block S when the i^{th} signal when through it. Notice that even though S_{e_i} and S_{e_j} are the same block, its configuration data may be different because they are recorded at different time, which is one of the reason why the same blocks are repeatedly recorded in the pipeline.

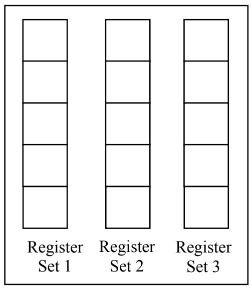
The behaviors of each register set are described in Figure 3: Each register set is responsible for keeping track of the logic blocks at corresponding level. Meanwhile, it also passes its data, recording the status of logic blocks, to the next register set every time a clock signal arrives.

Figure 3(a) shows the structure of the pipeline. Figure 3(b), (c), (d) show the data in register set 1, 2 and 3.

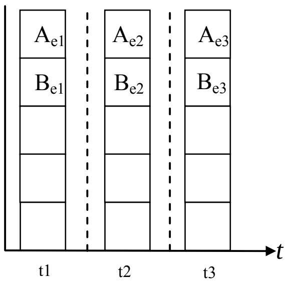
The last register set keeps record all the configuration stream of the same signal that has flowed through. These data are compared with the data in Configuration Memory in the other channel to detect if that signal is correct.

Imagine the whole FPGA as a workshop, and each logic block as a worker. Each register set plays a role as a check point, which records the status of workers at that point. The last register set keeps record of the status of all the workers participating in the production of a certain signal. The data of the last register set flow out of the channel are compared with the data in Configuration Memory in the other channel. A mismatch means the worker was not in mood when produced that signal, which indicated the signal is incorrect.

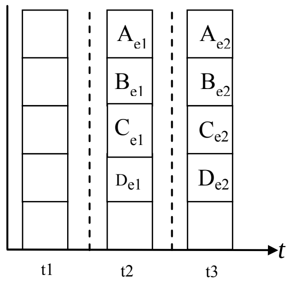
One of the advantages of the model is that the status of all the logic blocks (the workers) and signals are recorded at any time, just like a black box used in aircraft. Another advantage of this model is that it can accurately tell if the output signal is valid even when the blocks went wrong. This feature can also be illustrated using the example of the workers. If the worker is not in mood, the next signal it produced may be wrong, but the previous signal is still valid. This feature allows more signals to output when the logic blocks go failure, which makes sense to the application such as control unit in safety-critical system. The limit is that it



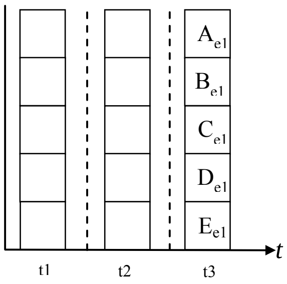
(a) Pipeline



(b) Data in Register Set 1



(c) Data in Register Set 2



(d) Data in Register Set 3

Figure 3: Inner structure of Model A
(The circuit under test is shown in Figure 2)

consumes more FPGA's resources. But it is still suitable for extensive applications which require high reliability without regard to the number of the hardware resources it occupies.

C. *Model B*

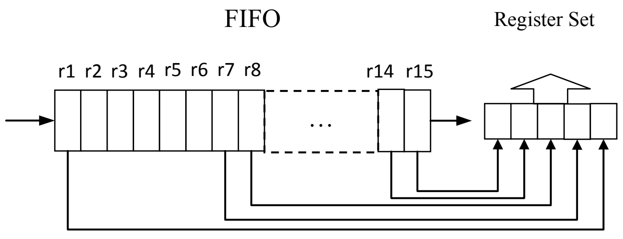
A modified FIFO structure is used to implement channel in Model B. One of the major differences from Model A is that Model B use less hardware resources. Another is the different structure of the channel.

The basic idea is to take the full use of the clock frequency of FPGA. The model uses the PLL (Phase Locked Logic) and the asynchronous FIFO (First In First Out). PLL is an electronic circuit built in FPGA which can provide a faster clock frequency, while an asynchronous FIFO provides feature for connecting two systems of different clock frequency.

A faster clock frequency is used to solve the problem of heavy overhead of Model A. Because the frequency has been raised by 5 times, the depth of each column in a channel is reduced to one.

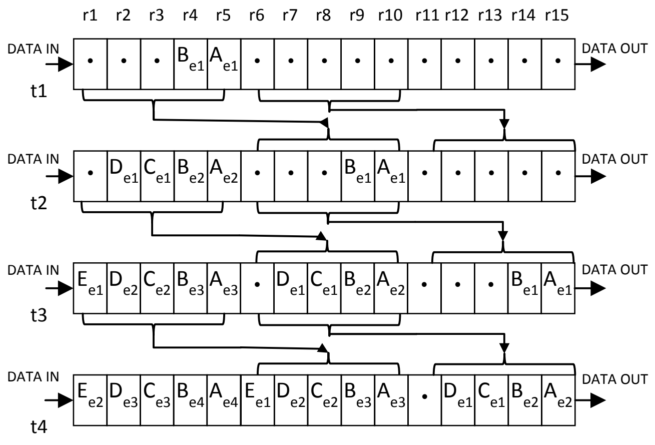
Compared to Model A, the detailed structure of the channel is different. Because it is not allowed by FIFO to insert data into the middle of it, additional register set is used to keep the data of the blocks that the same signal flows.

Figure 4(a) shows the structure of the channel. The entities in a FIFO which are connected to the registry set are responsible for keeping track of a group of blocks which produce the same signal. The registry set is needed because



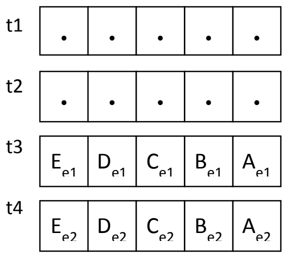
The frequency of FIFO is 5 times as that of the logic blocks.

(a) The structure of the channel


r1 r2 r3 r4 r5 r6 r7 r8 r9 r10 r11 r12 r13 r14 r15
DATA IN
DATA OUT
t1
• • • B_{e1} A_{e1} • • • • • • • • • •
t2
• D_{e1} C_{e1} B_{e2} A_{e2} • • • B_{e1} A_{e1} • • • • •
t3
E_{e1} D_{e2} C_{e2} B_{e3} A_{e3} • D_{e1} C_{e1} B_{e2} A_{e2} • • • B_{e1} A_{e1}
t4
E_{e2} D_{e3} C_{e3} B_{e4} A_{e4} E_{e1} D_{e2} C_{e2} B_{e3} A_{e3} • D_{e1} C_{e1} B_{e2} A_{e2}

' • ' represents empty data in FIFO

(b) The process of fulfilling the FIFO


t1
• • • • •
t2
• • • • •
t3
E_{e1} D_{e1} C_{e1} B_{e1} A_{e1}
t4
E_{e2} D_{e2} C_{e2} B_{e2} A_{e2}

(c) The content of register set

Figure 4: Inner structure of Model B
(The circuit under test is shown in Figure 2)

the blocks needed to compare are not in continuous position in FIFO. The content of the register set is exactly the same as the last register set in Model A, which is used for comparing with the configuration data in Configuration Memory.

Figure 4(b) shows how the data fulfill the pipeline. Initially at time t_1, Block A_{e1} and B_{e1} are enqueued into FIFO with 3 empty data one by one. Because data flow into the channel at 5 times the frequency of the logic blocks, it seems that 5 data are moving together. At time t_3, data are fulfilled in the FIFO. Then A_{e1}, B_{e1}, C_{e1}, D_{e1} and E_{e1} are ready to compare in the register set.

Figure 4(c) shows the content of the register set. In time t_1 and t_2 before FIFO is fulfilled, the register is empty. After FIFO is fulfilled, the content of the register then become the configuration data of the blocks that the same signal flowed through. For example, the register set at t_3 keeps track of the status of blocks that signal e_1 has flowed through and the register set at t_4 keeps track of the status of blocks that signal e_2 has flowed through.

The advantage of the Model B is obvious. It reduces the large overhead of the Model A without adding the time for fault detection. Nowadays, Cyclone III FPGA PLLs have well-support the frequency multiplication, and can reach the multiplication of the integer from 1 to 512. However, the FPGA frequency has its limit so that it cannot be raised too much. So Model B is more applicable in some small-scale circuits.

D. Model C

Both methods mentioned above can detect the fault signal instantly. However, they either require plenty of hardware resources or the higher clock frequency. In this section, we try to combine the advantages of both models to solve the heavy overhead.

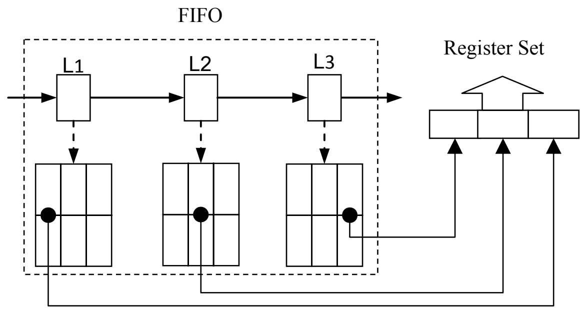
This model reduces the depth of the columns by p times and raises the frequency by p times compared to the Model A. The value of parameter p is important. If well selected, both the depth of FIFO and clock frequency will be acceptable.

Consider the FPGA circuits whose Block Tree has 3 levels (Figure 2). Originally, followed by the Model A, the depth of the pipeline is 6. If reducing the depth by 2 times, and raising the frequency by the same times, the result will be the pipeline of depth 2 and the clock frequency of 3 times (Figure 5). And thus both the depth and frequency are acceptable to current FPGA.

The inner structure of this method is illustrated in Figure 5.Because the basic structure we use is still a FIFO, a register set is needed to record a group of blocks producing the same signal. The structure of the FIFO is similar to that in Model B. The difference is the depth of the FIFO. Actually, Model B is a specialization of Model C, in which the depth of FIFO is one.

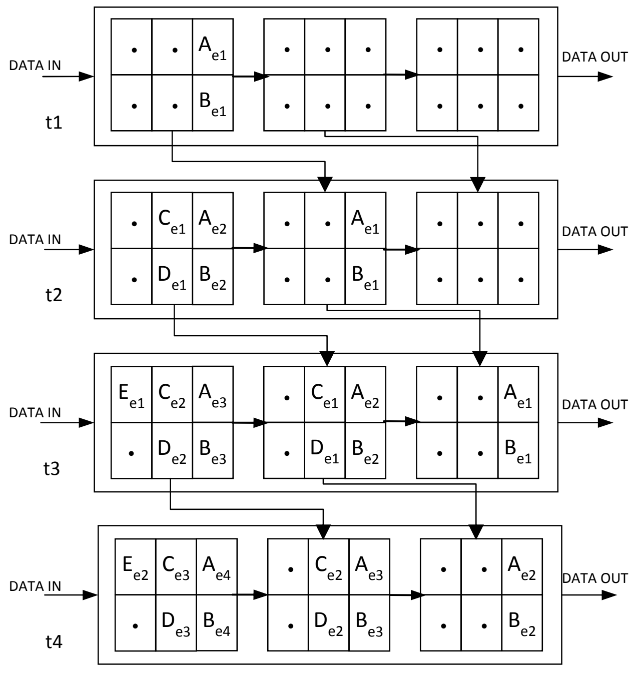
Figure 5(a) shows the structure of the channel. The entities in FIFO are divided into three groups. When a system clock arrives, one group will move to the position of the next group as illustrated in Figure 5(b). The register set keeps record of the configuration stream of the blocks that the same signal has flowed through.

Figure 5(c) shows the content of the register set. In time t_1 and t_2 before FIFO is fulfilled, the register is empty. After FIFO is fulfilled, the content of the register then become the configuration data of the blocks that the same signal flowed through. Then, the data in last register are compared with the data in Configuration Memory to detect if the signal is correct.



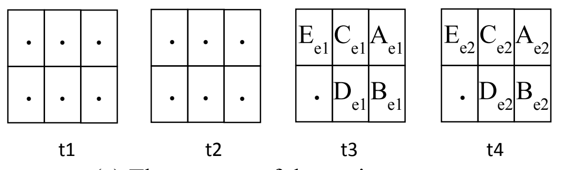
The frequency of FIFO is 3 times as that of the logic blocks.

(a) Structure of the channel



'•' represents empty data in FIFO

(b) The process of fulfilling the FIFO



(c) The content of the register set

Figure 5: Inner structure of Model C
(The circuit under test is shown in Figure 2)

The advantage of Model C is that we can strike a balance between the huge hardware resources and limited clock frequency. It can be applied to some highly reliable systems which have restriction on the hardware resources.

Table 1: Device utilization and performance summary

Model / Type	Model A		Model B		Model C	
Device Utilization Summary						
	Used	Utilization	Used	Utilization	Used	Utilization
Number of Slice Flip Flops	1,821	25%	1,666	23%	1,838	25%
Number of 4 input LUTs	1,538	21%	1,521	21%	1,513	21%
Number of occupied Slices	1,658	46%	1,507	42%	1,629	45%
Number of Slices containing only related logic	1,658	100%	1,507	100%	1,629	100%
Total Number of 4 input LUTs	1,538	21%	1,521	21%	1,513	21%
Number used as logic	1,474		1,345		1,449	
Number used for Dual Port RAMs	64		64		64	
Performance Summary						
Throughput(MB)	1,576.9		1,576.9		4,731.8	
Clock Frequency(ns)	80		70		80	
Clock frequency for FIFO(ns)			10		20	

Table 2: The comparison of Real-time BIST to Online BIST

	Runtime	Real-time	Mean hit rate of detecting SEU		
			8×8	16×16	32×32
Real-time BIST	Yes	Yes	100%	100%	100%
Online BIST	Yes	No	46.9%	38.7%	32.8%

IV. TESTING AND SIMULATION

The environment of the simulation is under the Xilinx Chip XC3S400-4PQ208.

The first test is to compare the performances of three models mainly in device utilization and its throughput. The circuit under test is a 4×4 configurable blocks. The clock interval of Model A, Model B and Model C is 80ns, 70ns and 80ns. The clock frequency of FIFO in Model B and C is 10ns and 20ns

In the second test, we simulate the method of online BIST, one of the most popular methods for fault detection, and then compare it with the method we proposed in this article.

The rows under test in online BIST are called STARs (self-testing area). The blocks in STARs are tested by testing vector generated by TPG (Test Patten Generator). When blocks in STARs finished checking, the STAR moves to the next rows detecting rows in new STAR. The time to detect the whole circuit is related to the scale of the circuit because of its mechanism.

In the test, different scales of the circuits are used to test both of two methods. This simulation uses mean hit rate of detecting SEU to measure the performance. Mean hit rate is the probability of successfully detecting an SEU when SEU occurs. Because the SEU is a soft error which may disappear before the detection has reached the location of the fault, the probability of finding an SEU that has occurred ought to be related to the scale of circuit under test.

V. TESTING RESULT AND ANALYSIZE

In the first test, both resource utilization and performance of three models are listed in Table 1. In the test, all three models can instantly check if the output signal is correct. However, the throughput of three models is significantly different. As indicated from Table 1, Model C reaches the largest throughput (4730.8MB). It is almost three times as that of Model A and Model B (1576.9MB).

The table also shows the device utilization of the FPGA. For hardware resources, Model A uses the most (1658 occupied slices) while Model B uses the least (1507 occupied slices). For the frequency of FIFO, Model B uses the highest (The FIFO is 7 times the frequency of system's clock frequency). In the testing, we found Model C strikes the balance between the hardware resources and the frequency in FIFO. (1629 occupied slices; 4 times the frequency of system's clock frequency) In addition, it also achieves the largest throughput.

In the second test, the comparison results are listed in Table 2. As seen from Table 2, though our method uses more resources than online BIST, it has advantages over online BIST in fault detection on SEU in many aspects: Firstly, though real-time BIST and online BIST can detect fault on runtime, online BIST cannot real-time detect fault because only rows in STAR are tested. Then, for online BIST, the mean hit rate of detecting an SEU decreases with the increase in the scale of the circuit under test (drop from 46.9% to32.8%). The hit rate is closely related to the scale of the circuit, because BIST have to continuously moving STARs before all the blocks are tested. On the contrary, our methods keep the hit rate of 100%, which means the hit rate won't be affected by the scale of the circuit.

VI. Conclusion

The simulation shows that all three models can instantly detecting whether the output signal is correct. Because the device utilization and the throughput are different, they can be applied to different application. Model A requires lots of hardware resources, so it is more acceptable to systems which have enough hardware resources. Model B need raising more clock frequency in FPGA, which can be applied in small-scale applications. Model C has larger throughout and less resources than Model A, so it is applicable to systems which allows raising the frequency and have constraint of hardware resources.

The advantage of our models is that it can detect the fault and locate them very fast. These features are important in some systems requiring high dependability. Although online BIST can locate the fault, some of the SEU may be missed due to its low speed for fault scanning. In addition, the mean hit rate of detecting an SEU in online BIST drops with the increase in the scale of the circuit. The advantage of our method is that it can keep the hit rate of 100% regardless of the circuit's scale. Another feature of real-time BIST is that it can real-time detect if the output signal is correct.

AKNOWLEDGMENT

This paper is Supported by The Shanghai university knowledge service platform(No.ZF1213) and East China Normal University Shanghai Key Laboratory of Trustworthy Computing （No.07dz22304201109）

References

[1] N. Bidokhti., "SEU concept to reality (allocation, prediction, mitigation)", Reliability and Maintainability Symposium (RAMS), Proceedings – Annual, pp. 1–5, Jan 2010.

[2] S. Mitra, and E. J. McCluskey, "Word-voter: a new voter design for triple modular redundant systems", VLSI Test Symposium, 2000. Proceedings. 18th IEEE, pp. 465–470.

[3] K. Matsumoto, M. Uehara, and H. Mori., "Stateful tmr for transient faults", World Automation Congress (WAC), 2010, pp. 1–6.

[4] M. Abramovici, and C. Strond, "BIST-based test and diagnosis of FPGA logic blocks", IEEE Transaction on Very Large Scale Integration (VLSI) Systems, vol. 9, issue.1, pp. 159–172.

[5] Chun-Lung Hsu, "Built-in Self-Test Design for Fault Detection and Fault Diagnosis in SRAM-Based FPGA", IEEE Transactions on Instrumentation and Measurement, vol.2. issue.7, pp. 2300–2315.

[6] M. Abramovici, C. Strond, C. Hamilton, S. Wijesuriya, and V. Verma, "Using roving STARs for on-line testing and diagnosis of FPGAs in fault-tolerant applications", Test Conference, 1999. Proceedings. International, pp. 973– 982

[7] M. Abramovici, J. M. Emmert, and C. E. Stroud, "Roving STARs: an integrated approach to on-line testing, diagnosis, and fault tolerance for FPGAs in adaptive computing systems", Proceedings Third NASA/DoD Workshop on Evolvable Hardware, 2001, pp. 73–92.

[8] N. R. Shnidman, W. H. Mangione-Smith, M. Potkonjak, "Fault scanner for reconfigurable logic", Advanced Research in VLSI, 1997. Proceedings. 17th Conference , pp. 238–255, Sep 1997.