FPGA Test Time Reduction Through a Novel Interconnect Testing Scheme

Stuart M^{c}Cracken Zeljko Zilic

Dept. of Electrical and Computer Engineering

McGill University, Montreal, Canada

smccra@macs.ece.mcgill.ca zeljko@macs.ece.mcgill.ca

ABSTRACT

As device densities increase, testing cost is becoming a larger portion of the overall FPGA manufacturing cost. We present an approach to speed up testing FPGA interconnect by reconfiguring it during the test. Simple additions are made to create feedback shift register structures, which considerably reduce the number of test configurations for the switching matrix interconnect. This new testing architecture reduces switching matrix test time by 66% and the diagnosis time by 72%. The additions are transparent to the users both in terms of speed and functionality.

1. INTRODUCTION

FPGAs offer a great reduction in the design development and product release times. To facilitate wider use of FPGAs, it becomes critical to reduce their inherent cost. Current FPGA costs arise from low yields, high production costs and testing costs. Yields can be improved through fault tolerance. Production costs are technology dependent. Testing costs depend on the time required to perform the test set. We propose to reduce this required time.

FPGAs need to be fully tested before they are shipped to customers. Testing incurs significant costs in producing microelectronics devices [17]. In fact, testing cost has been increasing in comparison to the other costs of design and manufacturing [1]. The cost of testing is determined by the length of time required to test each chip. By reducing this time, the cost of testing is also reduced, which ultimately translates into savings for users and manufacturers [17].

Additional savings for FPGA manufacturers come from yield improvements. Yields can be improved by enabling the use of partially faulty devices by fault tolerance. However, to incorporate fault tolerance, the devices need to be not only fully tested, but also diagnosed in order to know what needs repairing. It is thus the diagnosis improvements described in this paper that enable use of fault tolerance, and consequently improve yields. Hence, our scheme can further help in improving yields, and reducing the costs even further.

FPGA'02, February 24-26, 2002, Monterey, California, USA.

In this paper, we propose a scheme to reduce the testing and diagnosis time of an FPGA switch matrix interconnect structure. This reduction in testing time is achieved by realizing several test configurations by a single FPGA bitstream. The scheme only requires the addition of a nonlinear feedback shift register (nl-FSR) into the SRAM cells of the existing interconnect structure.

The paper is organized as follows. The considered FPGA interconnect structure is described in Section 2. The baseline testing approach on which we build our scheme is presented in Section 3. Our new nl-FSR testing and diagnosis scheme is described in Section 4. To realistically assess the utility and the costs of the proposed scheme, we have built and laid out an FPGA that uses our scheme. The analysis of the scheme using the data collected from an implemented circuit is presented in Section 5.

2. INTERCONNECT STRUCTURE

FPGAs are comprised of three major parts: the logic block (LB) array, the interconnect structures, and the I/O blocks. The logic blocks perform the combinational logic, along with registering the data where necessary. The I/O blocks provide the interfaces between FPGA and the external world. Finally, the interconnect structure routes the internal signals between I/Os and logic blocks.

The interconnect structure occupies over 50% of the FPGA area [2]. Thus, it is critical, however costly to fully test it. Although switch matrix testing is the focus of this paper, the principles of our nl-FSR testing scheme can be applied to most of the major FPGA building blocks, as outlined in Section 4.5 and Section 6.

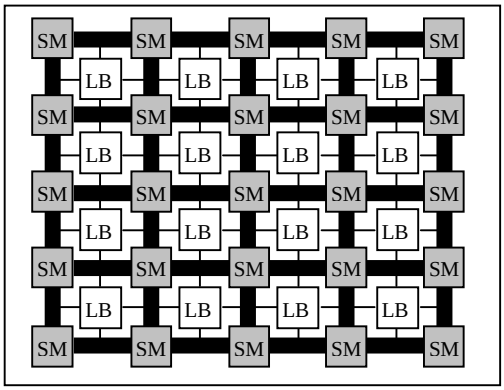
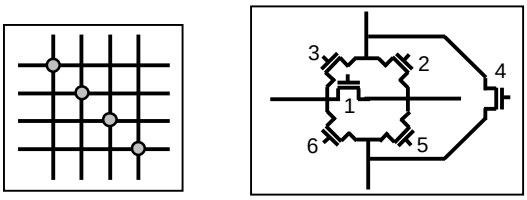


Figure 1 – FPGA Structure Considered

The interconnect structure is comprised of *wire segments* and *switching matrices (SMs)*, which surround the logic blocks, as can be seen in Figure 1. We concentrate on the SM portion of the interconnect. The segmented routing architecture is highly vendor-dependent, while testing of SM blocks, as presented here, is much more universally applicable.

In addition to SM blocks, there are also the connection switches, often referred to as C-modules. They connect the routing structure and the logic blocks, and are considered to be a part of the logic block structure, at least from the testing point of view. This association arises because testing of the connection blocks always requires the use of the LBs [3].

The SMs facilitate flexible data routing throughout the FPGA. The SMs connect wire segments by pass transistors or buffers. Figure 2a shows an example of a SM, where each dot is called a cross-point. Wire intersections without a dot are not connected, unless there is an undesired short. Each cross-point realizes the connection between the North, East, South, and West sides. This connection pattern is achieved with 6 pass transistors, allowing for all the possible connection permutations. A diagram of a cross-point is shown in Figure 2b [4]. The controlling inputs at the transistor gates are stored in SRAM cells, which are set during FPGA configuration.



a) b)

Figure 2 – (a) Switch Matrix and (b) Cross-Point Structure

Much effort has been put into the study of the number and placement of cross-points within a SM [5], [6]. As customary in research [13], a diagonal symmetrical set of cross-points will be used. In other words, the testing will deal with an *N x N* square matrix where a cross-point can be found at each point *(I, J)* where *I=J*. The technique described is equally valid to other cross-point placements.

3. SWITCH MATRIX TESTING

3.1 Basics

Switch Matrix testing has been undertaken in several ways. Much research has been done on using the available resources (logic blocks) to perform the roles of test pattern generation and output analysis [7], [8], [18], [20] in built-in self-test (BIST) techniques. Alternative techniques have been developed which do not incorporate BIST pattern generators and analyzers. Instead, multiple FPGA programs, often referred to as *bitstream configurations*, are used [9], [10], [15]. The test vector generation and the output analysis are performed externally, by a standalone tester.

These test methods are all based on the following principle. The SRAM bitstream is first set programmed into an FPGA in a certain *test configuration* that tests a portion of the faults in the given block under the test. Stimuli are applied at the inputs, and the outputs are analyzed. The procedure is repeated until all faults are tested. As downloading a bitstream takes much more time than applying test vectors, the problem becomes that of finding the minimal set of necessary test configurations to detect all faults.

The BIST approach to FPGA manufacturing tests uses internal elements for the test pattern generator (TPG) and output response analyzer (ORA). The BIST approaches are less often used for FPGA manufacturing testing, as the added ORA and TPG circuitry, implemented in FPGA LBs, would leave some LBs and interconnect incompletely tested in a given test configuration. Our scheme relies on the external TPG and ORA sources to avoid these dependencies. Furthermore, non-LB based BIST is possible, which would, however, incur an area penalty involved in including the TPG and ORA, and a time penalty required to test the BIST system.

There are two levels of fault assessment that can be performed: *testing* and *diagnosis*. The first is simply an evaluation process, where a device is deemed to be faulty, or fault free. The second step is to diagnose, i.e. to determine a fault's source. We consider both testing and diagnosis of FPGA interconnects.

3.2 Configurations for Test and Diagnosis

We consider test configurations by Wang and Huang [3] as a baseline testing approach, that will be sped up by adding the nl-FSRs. We first briefly review their findings on the required configurations for both testing and diagnosis.

The authors showed that 11 test configurations are necessary in order to fully test and diagnose the SM structures as in Figure 2. Furthermore, if only testing is desired, only three configurations are required. The resulting 11 test configurations are given in Figure 3. To see why the 11 test configurations are required, consider the transistors that are turned on for each case, as shown in Table 1. For instance, the test configuration a) in Figure 3 turns on the transistors 1, 3 and 5.

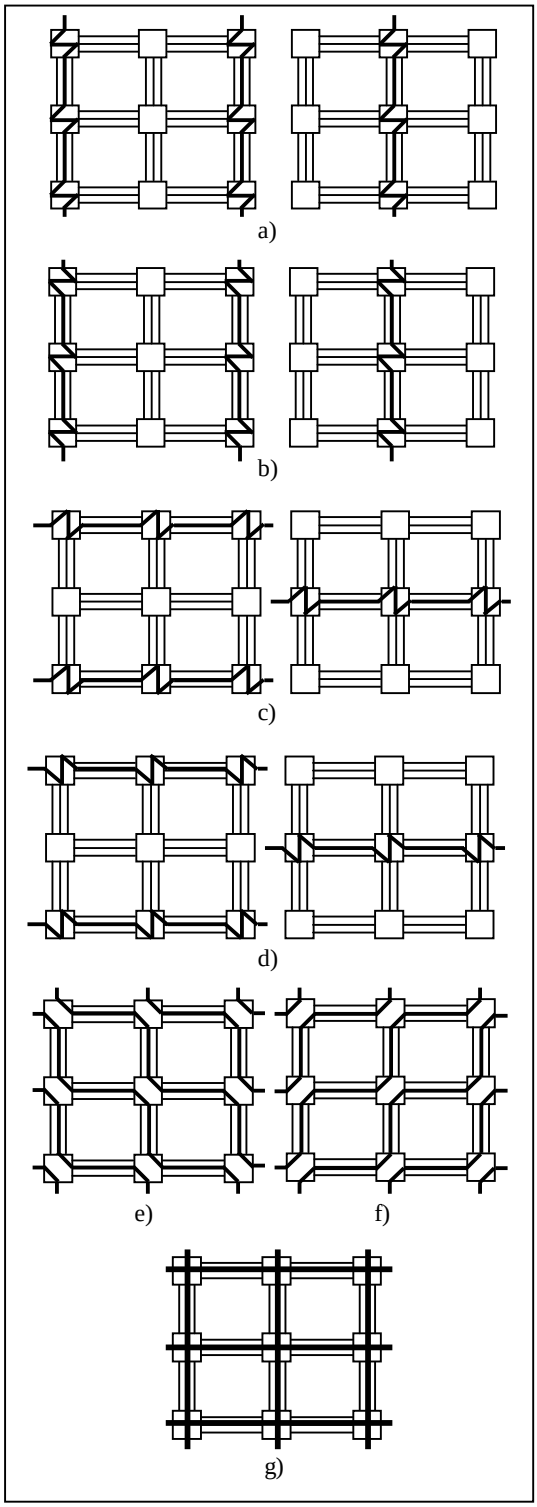


Figure 3 – 11 Test Configurations from [3]

Two consecutive columns cannot be tested concurrently in cases a) through d). Instead, two configurations are required to fully test a cross-point for these pass transistors. This is due to the fact that the horizontal interconnect wires between two adjacent SMs will be driven by any transistors that are turned on. Thus, if two adjacent SMs are both set on, they will both drive the same wire. This can either lead to contention, or to false results.

Table 1 – List of the Enabled Transistors per Configuration

Case	Transistors	Required for Testing	Required for Diagnosis
A	1 3 5	No	Yes
B	1 2 6	No	Yes
C	3 4 5	No	Yes
D	2 4 6	No	Yes
E	2 6	Yes	Yes
F	3 5	Yes	Yes
G	1 4	Yes	Yes

As can be seen in Table 1, each transistor is tested with 3 configurations. This allows for the exact faulty cross-point and transistor to be located, i.e. diagnosed. On the other hand if only testing is required, with tests e) to g), every SM transistor on the chip can be determined to be either faulty or fault-free.

3.3 Test Procedure

Huang and Wang demonstrate in their paper an approach to test the SMs for both stuck-at value and bridging faults for each test configuration. The test procedure starts by passing opposite polarity values to adjacent inputs. Thus, all the odd numbered lines will have a '1' passing through them and the even numbered lines would have a '0' passing through them. This allows the testing of bridging faults. In the next time step, the polarities are reversed. Thus, all stuck-at value faults have been detected. Equally, by considering both steps, bridging faults have been tested for the interconnect wires.

3.3.1 Test Scheme Advantages and Limitations

One benefit of this method is the fact that the size of the SM does not affect the testing scheme. Nonetheless, this test procedure is dependent on the number and position of cross-points.

In order to test and diagnose the FPGA switch matrix interconnect with this scheme, 11 configurations are required. In each configuration, two input streams are required, one to test for a stuck-at '0' fault and one to test for a stuck-at '1' fault. Together, bridging faults are tested as well. Therefore, the time required to test and diagnose a chip is:

$$t_{diag} = 11 * t_{configuration} + 22 * t_{signal}$$

These 11 configurations allow pinpointing of the specific transistor that is faulty. If such precision is not required, a subset of the tests is used. For instance, in order to test whether the chip SMs are faulty or not 3 tests are required e), f), and g).

All known procedures to test SM interconnects have the same drawback: each test requires downloading of a lengthy configuration bitstream, only to exercise few test vectors. As configuration times increase with the size of the FPGA, while the number of vectors is constant in a given test configuration, the whole testing procedure is dominated by FPGA configuration downloads. Recent advances as to reconfigurability of individual or groups of SRAM bits, such as Xilinx's J-bits, does not help in this case, as all SM SRAM bits need to be reconfigured. The configuration, or reconfiguration, overhead still exists.

The aim of the novel testing scheme presented in this paper is to reduce the number of reconfigurations required in order to test these interconnect structures.

4. NL-FSR TEST TECHNIQUE

As previously discussed, either 3 or 11 configuration steps are required for an FPGA SM interconnect, depending whether we want to test or to diagnose it. This is costly in terms of test time, as repeated configurations incur high time penalties. Therefore, the goal is to reduce the required number of configuration bitstreams while fulfilling all 3 (or 11) test set-ups. This goal is achieved by changing the test configuration during test. For this, some SRAM cells will have to be modified after each test configuration is exercised.

4.1 Proposed Scheme

Our scheme reduces the test and diagnosis time by performing on-the-fly reprogramming to realize several test configurations with a single bitstream. The new test architecture is achieved with no influence on the FPGA functionality from the user's point of view. Furthermore, the design aims to incur the smallest area overhead possible.

Our nl-FSR based scheme adds a hardware overhead to the SRAM cells that control an SM block. Figure 4.a) shows how the nl-FSR is attached to the SM controlling SRAM cells. An FSR series of D flip-flops is added to each SM. The length of the series is the same as for each cross-point, which is to say 6. These are programmed at the same time as cross-point$_{0,0}$ in each SM. The chain of DFF cells, each cell denoted by D, is linked in a shift register manner with a feedback function in combinational logic feeding the input. The SRAM cell outputs are connected to the SM pass transistors, which are inside the SM, and not shown in the diagram. The burden of the additional DFF cells is lessened the more clustering is expanded. The only circuits to be added are the feedback combinational logic gate, the multiplexors to route the right SRAM configuration data, the DFF cells, and the added local routing wires. As explained in the previous section, each cross-point in a switch matrix has the same configuration as the other cross-points therein. Thus, it is only necessary to have one combinational logic function per switching matrix. This reduces the potential area overhead considerably.

Furthermore, the nl-FSR acts like a Finite State Machine (FSM), where each value contained in the DFF cells determines the current and next state. Basically, the nl-FSR is designed to ensure that all test sequences are present at some point in the cycle. Intermittent useless states can arise in between useful testable states, but this is a very little price to pay in terms of time compared to the benefits arising from not having to reconfigure the FPGA.

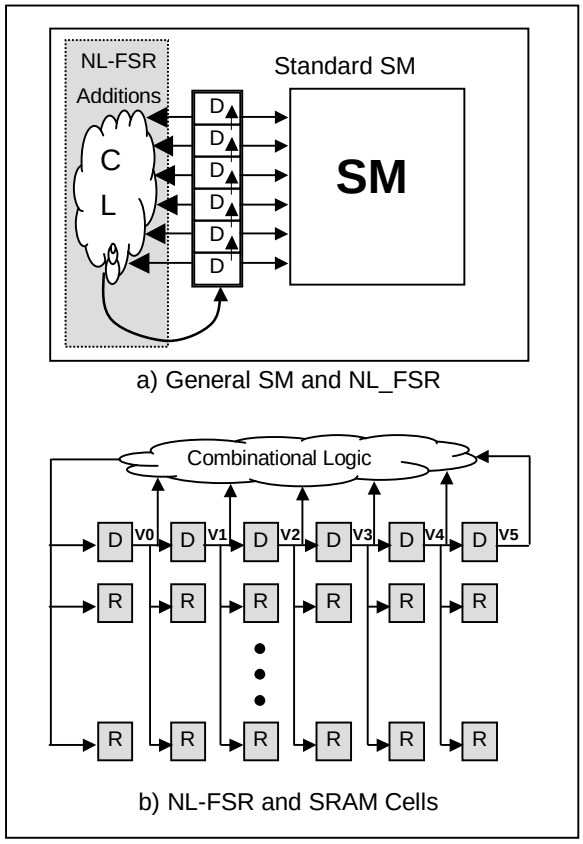


Figure 4 – FSR Switch Matrix Structure

4.2 Switch Matrix Symmetry

We are now presenting the major steps in the design of our testing scheme. The objective is to provide all the test configurations, while minimizing both the number of FPGA reconfigurations and the amount of logic added.

As shown in Figures 3 and 5, test and diagnosis partitions all SMs into four different sets, depending on their row and column number. The first set includes all SMs in an odd column and an odd row ($S_{0,0}$). The second set includes the SMs in an odd row and an even column ($S_{0,1}$). The last two are $S_{1,0}$ and $S_{1,1}$. All switching blocks of the same group have the same settings for all test configurations. For instance, the $S_{0,0}$ SMs all have the same configuration for test cases a) through g), yet different configurations than $S_{0,1}$, $S_{1,1}$,

and $S_{1,0}$ for certain test cases. The same can be said of the other three divisions.

Two distinct configuration groups are visible: those tests where all the switching matrices are the same (e, f, and g), and those where certain configurations are different (a-d). Dealing with cases e), f), and g) simply requires that all the transistors are in the same states. When looking at cases a-d, however, certain switch matrices need to be on whereas others need to be off.

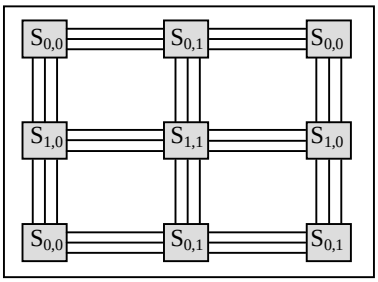


Figure 5 – Switching Matrix Division

Moreover, when considering cases a) through d), it is visible that when a set $S_{x,y}$ (where x and y can be different or the same) is on, set $S_{!x,!y}$ is necessarily off. This allows us to treat, for instance, $S_{x,y}$ with respect to $S_{x,!y}$ (c_1 and d_1) and $S_{!x,y}$ (a_1 and b_1) with no need to integrate $S_{!x,!y}$. And then by dealing with $S_{!x,!y}$ with respect to $S_{x,!y}$ (c_2 and d_2) and $S_{!x,y}$ (a_2 and b_2), all cases will be considered.

Thus, three test sections can be seen: 1) All switching blocks act the same, 2) $S_{x,y}$ is on, while $S_{!x,!y}$ is off, 3) $S_{!x,!y}$ is on, while $S_{x,y}$ is off.

4.3 Feedback State Table Construction

Our scheme uses a three step method. The first step is to analyze the structure of the problem and create the relation graphs describing the test configurations. The second step is to create an appropriate FSR. The final step is to optimize the combinational logic. Sections 4.1 and 4.2 described step 1. Now, we move on to step 2. Finally, Section 4.4 addresses step 3.

The next step in our construction involves creating the appropriate sequence that produces all the test configurations. Generating this sequence can be done in several manners. We first describe our exploration of the traditional feedback structures and show their inadequacy.

4.3.1 Known Solutions: LFSR Use

In order to add the least possible amount of circuitry to achieve the goal, the common traditional strategies were considered first. Adding additional SRAM cells or registers, for instance, would be costly in terms of the area. Therefore, using the existing 6 SRAM cells in the SM presents the optimal choice

One subset of FSRs that have been widely used and have useful properties is the LFSR (Linear FSR). Their combinational logic is entirely based on XOR gates. Furthermore, as they are linear, an algebra can be used in order to create custom sequences, which have the exact required output.

We employed two algorithms in order to create an LFSR with the desired properties. The first is the Berlekamp-Massey [11] algorithm. The second method, which was contrived for this purpose, involves checking the parity of a subset of each literal and seeing if their parity and output are consistent for all elements of the desired sequence. Unfortunately, both blocks resulted in LFSRs with lengths greater than 10, which results in the addition of too many registers.

Another method for creating efficient LFSRs involves reseeding [12]. In this method, a series of LFSR start values are stored in memory and loaded into the LFSR intermittently to supply the correct test vectors for full testing. This is inefficient for FPGAs as a lot of storage is required to hold several seed values in memory. Thus, the area overhead for reseeding due to storing memory locally is far too great. On the other hand, storing the memory globally or externally, requires a lot of configuration time, which is equally impractical.

Thus, linear FSRs and reseeding add to much memory and area overhead. As this is undesirable another solution needs to be found.

4.3.2 The nl-FSR Solution

Our solution uses the elements of reseeding, while not restricting ourselves to linear FSRs. Instead of using the local reseeding memory, we allow a small number of configuration (seed) downloads, as customary to FPGA testing.

Towards this goal, we first create a table where all test cases are handled with no conflicts. This means that no SRAM state can at one instance require a feedback value of '1' and at another time require a feedback value of '0'. This inconsistency is obviously not appropriate to be designed in combinational logic.

In the appendix, a table containing all the states and the transitions is presented. This table shows that two combinational logic circuits (C_{same} for $S_{0,0}$ and $S_{1,1,}$ and C_{dif} for $S_{0,1}$ and $S_{1,0}$) are required to achieve the feedback function for each set.

Within this table we can see that three seed vectors are required in order to test all 11 cases. Nonetheless, if only a test is needed to determine whether the chip is faulty, only one configuration is required, Group 1. This table shows

how $S_{0,0}$ and $S_{1,1}$ are controlled by the same function (C_{same}), while $S_{0,1}$ and $S_{1,0}$ depend a different function (C_{diff}).

4.4 Feedback Combinational Logic

The values from this table were used to create Berkeley Logic Interchange Format (BLIF) files describing each circuit. SIS [19] was then used in order to find the minimal sum-of-products form with the `script.rugged` and the script for multilevel optimization for area.

Thus, the required feedback logic for $S_{0,0}$ ($S_{1,1}$) is

$$C_{same} = \overline{V0}\cdot\overline{V1}\cdot V2 + \overline{V0}\cdot\overline{V2}\cdot V4 + \overline{V1}\cdot\overline{V3}\cdot V4 + \overline{V1}\cdot V3\cdot V5$$

And for $S_{0,1}$ ($S_{1,0}$),

$$C_{diff} = V1\cdot V4\cdot V5 + V2\cdot V3\cdot\overline{V5} + \overline{V1}\cdot\overline{V2}\cdot V3\cdot V5 + \overline{V1}\cdot V2\cdot\overline{V3}\cdot V5 + \overline{V0}\cdot V1\cdot\overline{V2}\cdot\overline{V5} + \overline{V0}\cdot\overline{V1}\cdot\overline{V3}\cdot V4\cdot\overline{V5}$$

This produces the correct feedback pattern with a limited amount of area overhead. Although the feedback logic gate, C_{diff}, is relatively large, only one such gate is required per switching matrix. As clustering is becoming more and more prevalent, switching matrices are becoming larger and the hardware overhead of the feedback logic is lessened.

This work was only done on a sparsely populated diagonal switch matrix, but is entirely scalable to other cross-point patterns.

4.5 Extrapolation to Testing Irregular SM

Irregular SMs, where the cross-point pattern does not follow the regular pattern as shown in Figure 2a), can also be tested, as shown next. The testing phase of the process is unchanged, such that cases e), f), and g) from Figure 3 can be handled. Likewise, only Group 1 from the table in the Appendix is necessary to test the chip.

This property arises from the fact that given any irregularly patterned SM, the inputs correspond to the outputs in a 1-to-1 fashion for all three test cases. This is shown in Figure 6. The letters e, f and g refer to the test cases from Figure 3 [9]. Test cases e and f have been divided pictorially for clarity but physically occur at the same time. Having a 1-to-1 correspondence is necessary to ensure that all cases are tested where only one input can influence each output. Not having the 1-to-1 correspondences would result in multiple inputs affecting certain outputs. This would make the job of exercising certain paths and not others more complicated and could render certain paths untestable.

In the following paragraph, all transistor numbers refer to those of Figure 2. In case e_1) and e_2), the T2 and T6 pass transistors are on. The inputs enter from the North and West sides, whereas the outputs can be found on the South and East sides. It can be seen that each input corresponds to a particular single output. Furthermore, each pass-transistor (T2 and T6) is exercised once, and no more. For case f_1) and f_2), the same can be said about the T3 and T5 pass transistors, where the North and East sides are inputs, and the South and West sides are outputs. Finally, for case g), the North and West sides are inputs and the South and East sides are outputs. In this case, the T1 and T4 pass transistors are tested. The scheme can be double-checked by observing that all 6 transistors from Figure 2 are exercised for each cross-point.

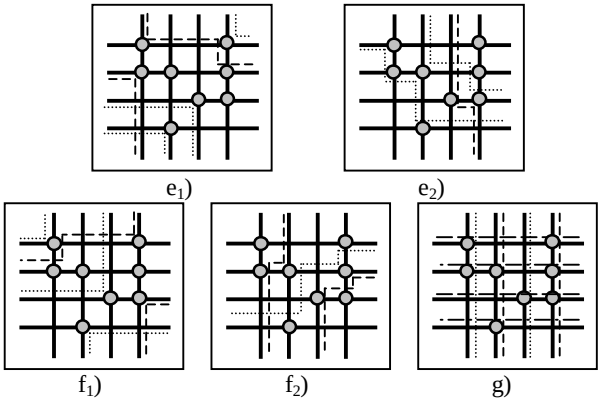


Figure 6: Testing of Irregular SM Patterns

The stated results hold for all potential cross-point placements of an SM. Extrapolating this result to include the entire FPGA routing structure including all SMs is possible, as the entire routing structure can be seen as one large SM. Furthermore, if SMs with differing cross-point placements are found on a same FPGA, the same can be said as, regardless of the SM structure, the inputs have a 1-to-1 correspondence with the outputs. Consequently, the routing structure can be tested with our scheme.

Although we just showed that the testing of the alternative structures is easily handled by our scheme, the diagnosis of irregular cross-point patterns is a more complex issue, which is currently being investigated

5. IMPLEMENTATION AND SCHEME PERFORMANCE

To test the feasibility of our approach, and to accurately assess the hardware cost, an FPGA is being fabricated in TSMC's 0.18µm technology. All elements of our scheme have been designed and added to a homogeneous 4-LUT FPGA architecture. The design has been simulated and laid out using Cadence tools, and the area of all major blocks in FPGA was extracted after the layout and the extracted simulations were completed and the design rule checker (DRC) has passed.

This section presents the impact of integrating this method to an FPGA in terms of the testing speed, hardware overhead,

and delay for both configuration and functionality. Finally, this method is compared to several current methods for testing FPGA interconnects.

5.1 FPGA Implementation

The FPGA that was implemented uses a simple FPGA base, into which the required circuitry for the nl-FSR SM interconnect test circuitry is integrated. Much of the sizing and layout for the core was based on [13]. Furthermore, the architecture is based upon a non-clustered logic block structure with a cross-point placement structure as in Figure 2a). The device acts as a proof of concept and as such all the results in this section were derived from the simulations, functionality and layout of this device. The following figure shows the core layout of the device.

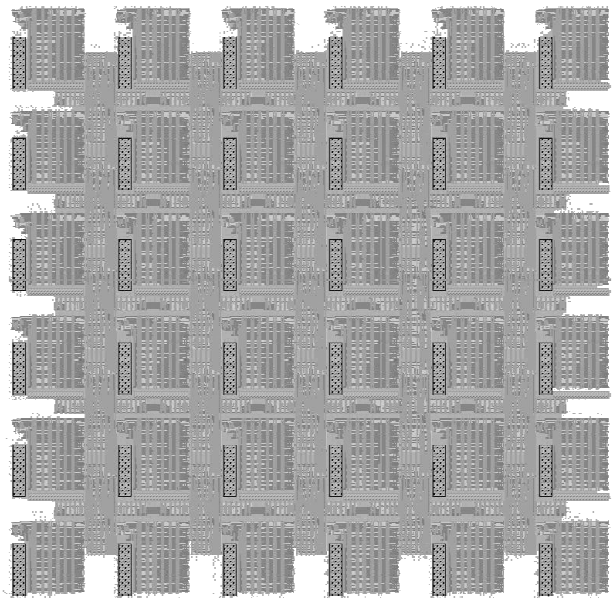


Figure 7 – Layout of NL-FSR SM FPGA Test Core without I/Os

The layout shown is that of a 6x6 square matrix. It does not contain the I/O logic and routing. The I/O routing and logic would be placed in the white (empty) layout area in the peripheral sections. In order to test this circuit I/O blocks are used to scan out the test result data. Furthermore, specialized pins are required to control the testing. This can be achieved through JTAG, which reduces overhead as it is already present in FPGAs.

5.2 Speed Comparison

In order to compare the test techniques it is necessary to compare the time that they take to perform all testing. The traditional test technique requires

$$((6\times\#SM\times\#CPpSM)+(17\times\#LB)+(10\times\#CB))\times 11\,cycles$$

where: #SM is the number of switching matrices

#CPpSM is the number of cross-points per switching matrix

#LB is the number of logic blocks

#CB is the number of connection blocks

Therefore, as an approximation a reasonable number of logic cells is 2000, which implies that there are approximately 2000 switching matrices, and 10000 connection blocks. Equally, 10 cross-points per matrix in a disjoint switching architecture is a reasonable estimate. The resulting number of required clock cycles for testing is 2 794 000 cycles. This is a bare minimum as to the number of configuration cycles required as there are more configuration SRAM cells than have been estimated in this basic system.

On the other hand, the proposed technique needs

$$((6\times\#SM\times\#CPpSM)+(17\times\#LB)+(10\times\#CB))\times 3\,cycles+41\,cycles$$

With the same assumptions as above (#LB = 2000), we obtain a test time of 762 041 cycles. This is a reduction of 72.7%. When the number of logic blocks is large, the 41 cycles in the proposed method become negligible. As such, the improvement becomes a difference between the 3 and 11 configuration passes.

Equally, if only testing is required and the diagnosis is omitted, the savings are 66%. This comes from the improvement from 3 to 1 configurations with the same assumptions as above.

5.3 Area Comparison

The standard test technique does not incur any area penalties, as it uses the chip as is. The proposed method, however, does require extra circuitry. This comes about in the form of feedback logic gates, and a 2x1 multiplexers.

Furthermore, certain test pins are required for the additional signaling to enable the test mode. Nonetheless, current FPGA architectures already include some test pins for similar purposes. Another alternative is to use existing JTAG circuitry that is already found in FPGAs. Thus, this overhead is subsumed by the basic device. Also, the following calculations do not take into consideration any area overhead due to I/O circuit changes to input stimuli and output responses. These area additions, however, can also be minimized.

One complex library gate is required per switching matrix, and one multiplexor is required per cross-point. Let the size of the complex gate be estimated as the size of 3 SRAM cells, and each multiplexor be estimated as the size of one SRAM cell. The increase in area in each switch matrix is estimated as:

$$\frac{3+(1\times\#CPpSM)+(6\times\#CPpSM)+(\#PT/6)}{6\times\#CPpSM+\#PT/6} = 1+\frac{3+(1\times\#CPpSM)}{6\times\#CPpSM+\#PT/6}$$

where: #PT is the number of tri-state and pass transistors in the switch matrix

Therefore, the increase in area within the switch matrix is (3+#CPpSM)/(6X#CPpSM + (#PT/6)). As the number of cross-points per switch matrix increases, the area overhead decreases. As FPGAs are clustering more and more logic, the size of the switch matrices increases to handle the need for routing more signals. Thus, the area burden of the FSR test technique is lessened.

The area overhead for the whole chip is much less than the overhead for the switching matrix. Estimates for FPGA sizings state that approximately 50% to 90% of FPGAs are dedicated to interconnect resources, and that 50% to 70% of interconnect resources are dedicated to wires. Furthermore, chip overhead such as pins and drivers add a lot overhead; approximately half the area can be dedicated to this. If we take the averages, we see that approximately 28% of the chip is dedicated to switching matrices. Thus, the increase in area due to the FSR addition is approximately (not including pin and driver overhead):

$$\frac{0.28\times(3+\#CPpSM)}{6\times\#CPpSM+\#PT/6} = \frac{0.28\times(3+40)}{(6\times40+(40\times6)/6)}$$

If we assume a clustering level of 4 like in the Xilinx Virtex Chips, from [13] we see that the number of cross-points per switch matrix can approximately be 40. Therefore, the overall area overhead in adding the FSR testing technique is 2.2%. This relatively small addition can reduce the test time by almost 73%. Note that as the size of the clusters grows the percentage depends solely on the size of the multiplexer.

In our FPGA implementation, the area overhead is 4.5%. However, this is a basic FPGA without clustering, which would inevitably reduce the overhead considerably.

5.4 Delay Overhead

There is no inherent functional delay added by including the nl-FSR testing scheme as the changes do not affect the critical paths of the configured circuits. The nl-FSR is only active in a test mode. They will, however, potentially slow down the configuration process if the added multiplexers fall into the longest configuration path. Nonetheless, as configuration is performed at low frequency and the bottlenecks are in the configuration logic itself, this delay is not bound to reduce the configuration speed.

5.5 Comparison to Other Techniques

As seen from the results, the gains in terms of testing speed are very visible. Both BIST-like [7,8] and scan method [3, 15] approaches to testing require considerably more bit-stream reconfigurations than this FSR approach. None of the methods influence the functionality of the FPGA when not under test, which is crucial for an acceptable testing solution.

The drawback to this FSR approach is the hardware overhead required to achieve the functionality. The percentages (2% to 4.5%), however, are not too large compared with the benefits to be a deterrent for integrating the FSR approach.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we presented the testing and diagnosis scheme for FPGA switching modules that can be used to reduce the FPGA manufacturing costs. The addition of a nl-FSR to FPGA switch matrix interconnect testing is beneficial as it reduces the testing time required by over 70%. Furthermore, the nl-FSR system is transparent to the end user and does not slow down the implemented designs. Certain hardware overhead is incurred, but as logic block clustering continues to be integrated into FPGAs this overhead is kept low. Finally, this scheme is fully exportable to the testing of FPGA I/O and logic blocks, resulting in greatly improved FPGA testing times. Although testing is considered here, in non-BIST type scenarios the same proposed scheme can be applied to BIST approaches. In that case, all area and time overheads from section 3.1 would apply.

As all FPGA blocks are based on SRAM controlling cells in SRAM FPGAs, this scheme is equally applicable to testing I/O blocks, and logic blocks (along with the connection modules). All that is required is to find the set of test vectors required, find the logic to achieve the pattern, and create an FSR with the appropriate feedback function. Testing is then be done in the same manner as testing demonstrated for the interconnect structure.

The ability to quickly diagnose SMs allows us to develop, in the future, the practical fault tolerant schemes that would improve FPGA yields. The overall reduction in the production cost by such a scheme would be of benefit to both FPGA manufacturers and users.

7. ACKNOWLEDGEMENTS

We would like to thank the Canadian Microelectronics Corporation (CMC) for providing services and funding to produce a custom designed FPGA device. We also thank Altera Toronto Technology Center for their support. Finally, the reviewer's comments were gratefully appreciated and hopefully addressed.

8. REFERENCES

[1] W.-T. Cheng. *Current status and future trend on CAD tools for VLSI testing*. Proceedings of the Ninth Asian Test Symposium, 2000. (ATS 2000). Page(s): 10–11

[2] S. Dutt, and F. Hanchek. *Methodologies for tolerating cell and interconnect faults in FPGAs.* IEEE Transactions on Computers, Volume: 47 Issue: 1, Jan. 1998. Page(s): 15–33

[3] S.-J. Wang, and C.-N. Huang. *Testing and diagnosis of interconnect structures in FPGAs.* Proceedings of the Seventh Asian Test Symposium, 1998. ATS '98. Page(s): 283–287

[4] H.-J. Lee, and M.J. Flynn. *High-speed interconnect schemes for a pipelined FPGA.* IEEE Proceedings of Computers and Digital Techniques, Volume: 147 Issue: 3, May 2000. Page(s): 195–202

[5] G. Lemieux, P. Leventis, and D. Lewis. *Generating Highly-Routable Sparse Crossbars for PLDs.* Proceedings of the ACM/SIGDA international symposium on Field programmable gate arrays 2000. FPGA '00. 2000. Page(s): 155-164

[6] W. Guo, and A.Y. Oruç. *Regular sparse crossbar concentrators.* IEEE Transactions on Computers, Volume: 47 Issue: 3, March 1998. Page(s): 363–368

[7] I. Harris, and R. Tessier. *Diagnosis of interconnect faults in cluster-based FPGA architectures.* ICCAD-2000. IEEE/ACM International Conference on Computer Aided Design, 2000. Page(s): 472 –475

[8] X. Sun, J. Xu, B. Chan, and P. Trouborst. *Novel technique for built-in self-test of FPGA interconnects.* Proceedings of the International Test Conference, 2000. Page(s): 795 –803

[9] M. Renovell, J.M. Portal, J. Figuera, and Y. Zorian. *Testing the interconnect of RAM-based FPGAs.* IEEE Design & Test of Computers, Volume: 15 Issue: 1, Jan.-March 1998. Page(s): 45 –50

[10] H. Michinishi, T. Yokohira, T. Okamoto, T. Inoue, and H. Fujiwara. *A test methodology for interconnect structures of LUT-based FPGAs.* Proceedings of the Fifth Asian Test Symposium, 1996. Page(s): 68 –74

[11] A. Menezes, P. van Oorschot, and S. Vanstone. Handbook of Applied Cryptography, CRC Press, 1996.

[12] S. Hellebrand, J. Rajski, S. Tarnick, S. Venkataraman, and B. Courtois. *Built-in test for circuits with scan based on reseeding of multiple-polynomial linear feedback shift register.* IEEE Transactions on Computers, Volume: 44 Issue: 2, Feb. 1995. Page(s): 223 –233

[13] V. Betz, J. Rose, and A. Marquadt, *Architecture and CAD for Deep-Submicron FPGAs*. Kluwer Academic Publishers, Boston, 1999.

[14] Xilinx, Inc. *The Programmable Logic Data Book 2000.*

[15] Y. Yu, J. Xu, W.K. Huang, and F. Lombardi. *Diagnosing single faults for interconnects in SRAM based FPGAs*. Proceedings of the Asia and South Pacific Design Automation Conference, 1999. Page(s): 283 -286 vol.1

[16] C.-F. Wu, and C.-W. Wu. *Testing interconnects of dynamic reconfigurable FPGAs*. Proceedings of the Asia and South Pacific Design Automation Conference, 1999. Page(s): 279 -282 vol.1

[17] Y. Zorian. *Emerging trends in VLSI test and diagnosis.* Proceedings of the IEEE Computer Society Workshop on VLSI, 2000. Page(s): 21-27

[18] C. Stroud, S. Wijesuriya, C. Hamilton, and M. Abramovici. *Built-in self-test of FPGA interconnect.* Proceedings of the International Test Conference, 1998. Page(s): 404 –411

[19] E.M. Sentovich, K.J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P.R. Stephan, R.K. Brayton and A.L. Sangiovanni-Vincentelli. *SIS: A System for Sequential Circuit Synthesis.* University of California at Berkeley, Memorandum No. UCB/ERL M92/41, Electronics Research Laboratory, College of Engineering, University of California, Berkeley, CA 94720, May 1992.

[20] I.G. Harris, and R. Tessier. *Interconnect testing in cluster-based FPGA architectures*. Proceedings Design automation Conference, 2000. Page(s): 49 –54

APPENDIX

Table A.1: Bit sequence for Switching Matrices

	$S_{0,0}$	Feedback $S_{0,0}$	$S_{1,0}$	Feedback $S_{1,0}$	$S_{0,1}$	Feedback $S_{0,1}$	$S_{1,1}$	Feedback $S_{1,1}$
1	001001	1	001001	1	001001	1	001001	1
	100100	0	100100	0	100100	0	100100	0
	010010		010010		010010		010010	
2	010110	1	010110	1	001000	0	000000	0
	101011	1	101011	1	000100	0	000000	0
	110101	0	110101	0	000010	1	000000	0
	011010	0	011010	0	100001	0	000000	0
	001101	1	001101	0	010000	1	000000	0
	100110	0	000110	0	101000	0	000000	0
	010011	1	000011	0	010100	1	000000	0
	101001	0	000001	0	101010	0	000000	0
	010100	0	000000	0	010101	0	000000	0
	001010	1	000000	0	001010	1	000000	0
	100101	1	000000	0	100101	1	000000	0
	110010	0	000000	0	110010	0	000000	0
	011001	0	000000	0	011001	0	000000	0
	001100	1	000000	0	001100	1	000000	0
	100110	0	000000	0	100110	0	000000	0
	010011	1	000000	0	010011	1	000000	0
	101001		000000		101001		000000	
3	000000	0	001000	0	010110	1	010110	1
	000000	0	000100	0	101011	1	101011	1
	000000	0	000010	1	110101	0	110101	0
	000000	0	100001	0	011010	0	011010	0
	000000	0	010000	1	001101	0	001101	1
	000000	0	101000	0	000110	0	100110	0
	000000	0	010100	1	000011	0	010011	1
	000000	0	101010	0	000001	0	101001	0
	000000	0	010101	0	000000	0	010100	0
	000000	0	001010	1	000000	0	001010	1
	000000	0	100101	1	000000	0	100101	1
	000000	0	110010	0	000000	0	110010	0
	000000	0	011001	0	000000	0	011001	0
	000000	0	001100	1	000000	0	001100	1
	000000	0	100110	0	000000	0	100110	0
	000000	0	010011	1	000000	0	010011	1
	000000		101001		000000		101001	

This appendix contains the data that was used to create the nl-FSR for interconnect SMs. The sequence of values that satisfies the constraints for an SM can be seen in Table A.1. The transistor order in the table is as follows: H, NE, NW, V, SW, SE (or 123456 when referring to Figure 2b). In the table, the shift that takes place is a shift right, where the least significant bit (LSB) is lost and the new bit is shifted into the MSB bit. The shaded squares indicate those that satisfy a test configuration. As can be seen, $S_{0,0}$ and $S_{1,1}$ have the same feedback output for the same inputs. Also, $S_{0,1}$ and $S_{1,0}$ have the same feedback output for the same inputs. Thus, the FSR logic for $S_{0,0}$ and $S_{1,1}$ will be the same, as will that for $S_{0,1}$ and $S_{1,0}$.