

BUILT-IN SELF-TEST OF VIRTEX AND SPARTAN II FPGAS USING PARTIAL RECONFIGURATION



Sachin Dhingra, Srinivas Garimella, Aditya Newalkar and Charles Stroud

Dept. of Electrical and Computer Engineering
 200 Broun Hall, Auburn University, AL 36849-5201
 email: dhingsa/garimsm/newalaa/strouce@auburn.edu

ABSTRACT: Built-In Self-Test (BIST) of Field Programmable Gate Arrays (FPGAs) provides the ability of in-system testing without area or performance penalty to the intended system function. The primary problem is the long testing time and memory storage requirements associated with the large number of BIST configurations typically needed for complete testing of the programmable resources in the FPGA. Some FPGA architectures support the ability to reprogram only the portion of the FPGA that changes from one configuration to the next. Some FPGAs also provide the ability to read partial contents of the configuration memory for data capture and verification purposes. These features can be exploited to provide significant reductions in the amount of memory for BIST configuration storage as well as the total time needed to test of the FPGA.¹

1. INTRODUCTION

Built-In Self-Test (BIST) approaches have been developed for Field Programmable Gate Arrays (FPGAs) where the FPGA logic and routing resources are reprogrammed to allow the FPGA to test itself without the need for external test equipment or dedicated circuitry for BIST [1]. The basic idea is to program some of the Programmable Logic Blocks (PLBs) as Test Pattern Generators (TPGs) and Output Response Analyzers (ORAs) to test the remaining programmable logic and routing resources. If the FPGA is determined to be faulty, the faulty resource(s) can be identified through diagnostic procedures and the intended system function can be reconfigured to avoid the faulty resource(s) for fault-tolerant operation. One of the main drawbacks is the large number of BIST configurations needed to completely test the FPGA in conjunction with the time required to download these BIST configurations into the FPGA and to retrieve the test results at the end of the BIST sequence [2]. This problem can be reduced by exploiting partial reconfiguration capabilities provided in FPGAs such as Xilinx Virtex and Spartan II series FPGAs [3]. In some cases, the total number of BIST configurations required can also be reduced by using more efficient ways of reading the test results.

In this paper, we investigate the impact of partial reconfiguration and partial configuration memory read-back on BIST for Virtex and Spartan II FPGAs. We begin

with an overview of prior work in BIST for FPGAs in Section 2. The impact of partial reconfiguration and partial configuration memory read-back on BIST for PLBs, RAMs, and routing resources is discussed in Sections 3, 4, and 5, respectively. Experimental data is given from implementations of the respective BIST approaches in actual Virtex and Spartan II devices.

2. PRIOR WORK IN BIST FOR FPGAS

The most frequently used approach to testing the programmable logic resources in the FPGA is to program a column (or row) of PLBs to function as two or more identical TPGs (Figure 1) [1] [2] [4]. The TPGs drive pseudo-exhaustive test patterns to alternating columns (or rows) of identically configured programmable logic blocks under test (BUTs). Comparison-based ORAs (Figure 2) are located in columns (rows) between the BUTs such that they monitor the outputs of their adjacent BUTs and latch any mismatches due to faults in the BUTs. The BUTs are repeatedly reconfigured in their various modes of operation until they are completely tested, then the architecture is flipped about the vertical (or horizontal) axis such that during the second test session the PLBs that previously functioned as TPGs and ORAs are now BUTs and vice versa. The BIST architecture can be either row or column based. A row based logic BIST architecture was used in [1] but it was observed in [4] that dedicated carry logic and routing as well as differences in vertical and horizontal routing resources made a column based architecture more effective for some FPGAs.

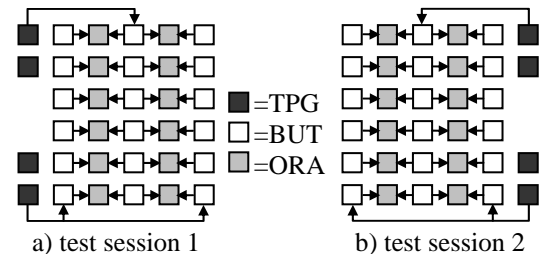


Figure 1. FPGA logic BIST architecture

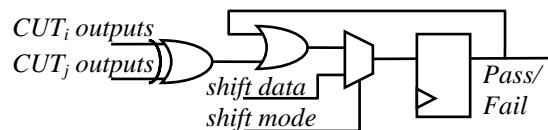


Figure 2. Integrated ORA scan chain

¹ This work was sponsored by the National Security Agency under contract H98230-04-C-1177.

At the beginning of each BIST sequence, the BIST configuration must be downloaded into the FPGA to configure the logic resources in their respective TPG, ORA, or BUT functions along with the routing resources to form the interconnections between TPGs and BUTs as well as between BUTs and ORAs. Full configuration, writing the entire configuration memory, of the FPGA has been used in off-line testing [1] [4]. Partial reconfiguration has been used in on-line testing to program only the portion of the FPGA that is under test at any given time with the remainder of the FPGA continues to perform the intended system function [5]. Dynamic partial reconfiguration via an embedded processor in a SoC was used for logic BIST of the FPGA core of the SoC in [6] with significant reduction in total test time. Due to the regular structure of the logic BIST architecture (Figure 1), partial reconfiguration would require only those portions of the FPGA that change from one BIST configuration to the next to be reconfigured; in logic BIST this would be the BUTs since the TPGs, ORAs and routing would remain the same.

At the end of the BIST sequence, the contents of the ORAs must be read to determine the faulty/fault-free status of the programmable resources being tested. Several approaches for retrieving the ORAs contents have been investigated [7] but two approaches have been most frequently used. In [1], the ORAs incorporate an integrated scan chain to scan out the test results stored in the ORAs (Figure 2). This approach requires additional logic resources for the multiplexer and routing resources for the shift data and control to form the scan chain. In [4], the additional logic and routing resources required for the integrated ORA scan chain was found to be prohibitive in that less resources could be tested in any given BIST configuration such that more test configurations would be required to completely test the FPGA. As a result, full configuration memory read-back was used in [4] to obtain the contents of the ORAs. While this effectively doubled the test time since the configuration memory must be written and read for each BIST configuration, it reduced the total number of BIST configurations needed to test the FPGA. In [8], dynamic partial reconfiguration was used to convert the ORAs to a shift register at the end of the BIST sequence to shift out BIST results without the overhead of the integrated scan chain in the ORA during the actual execution of the BIST itself. In [6], the ORA contents were read, via a reconfigured scan chain, at the end of a set of BIST configurations with no loss in fault detection and very little loss in diagnostic resolution. FPGAs with partial configuration memory read-back capabilities would facilitate reading only those portions of the configuration memory that contain the ORA contents to help reduce the total test time without the expense of additional logic resources and additional BIST configurations [2].

Recent FPGAs incorporate dedicated Random Access Memories (RAMs) to overcome the problem of using many Look-Up Tables (LUTs) that typically can function as small RAMs, to construct large RAMs. BIST for these dedicated RAMs has been addressed in [8] and [9] where PLBs are used to implement the TPG function based on RAM test algorithms and comparison based ORAs. The outputs of the RAMs under test are either compared with other RAMs under test (Figure 3a) or with expected output responses generated by the TPG (Figure 3b). In both approaches, a scan chain was incorporated in the ORA for retrieval of BIST results.

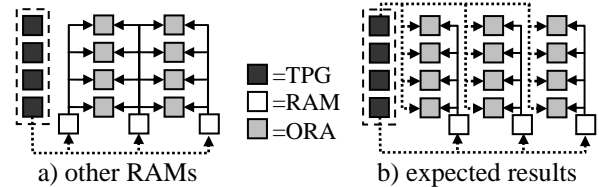


Figure 3. RAM BIST architectures

Two types of routing BIST approaches have proven to be effective in testing the programmable interconnect resources in FPGAs. The first is a comparison-based approach in which the TPG drives exhaustive test patterns over two sets of wires under test that are compared at the other end by a comparison-based ORA [10]. The second approach is parity-based where the TPG sources exhaustive test patterns over a set of wires under test and produces a parity-bit that is sent to the ORA [11]. The ORA generates parity over the data observed on the wires under test and compares the generated parity-bit with that sent by the TPG. While both approaches have been shown to be effective in detecting faults, the comparison-based approach was extended to diagnosis of faults in the programmable interconnect network [10]. The comparison-based approach was used in [4] and [10] while a modified parity approach was used in [7]. Full configuration download of the BIST configurations was used in [4] and [7] and partial configuration was used in the on-line BIST approach in [10]. Configuration memory read-back was used in [4] and an integrated scan chain was used in [10] for retrieval of ORA results. Dynamic partial reconfiguration from an embedded processor was used in [7] to convert the ORAs to a scan chain.

As can be seen from the discussion above, various combinations of full and partial reconfiguration of the FPGA have been used for BIST. Also various approaches for reading the contents of the ORAs at the end of the BIST sequence have been implemented. However, there has not been a detailed comparative investigation of all combinations of partial reconfiguration and partial configuration memory read-back versus scan chain retrieval of BIST results. Such an investiga-

tion is the focus of this paper. Xilinx Virtex and Spartan II FPGAs are used as the target device in this investigation since they have identical architectures and support both partial reconfiguration and partial configuration memory read-back [3] [12] [13] [14] [15].

3. LOGIC BIST

The architectures of Virtex and Spartan II FPGA families are essentially identical. The PLB consists of two slices each containing two 4-input Look-Up Tables (LUTs) and two flip-flops along with additional carry and control logic [3] [12]. The PLB arrays range from 8×12 (i.e. 8 rows and 12 columns of PLBs) to 64×96 in increments of 4×6 . The LUTs can be also configured as a shift register or small RAMs (to be discussed in the next section). The flip-flops in the PLBs can also be reconfigured as latches. Additional multiplexer logic provided in the PLBs facilitates the formation of function generators capable of more than 4-inputs. The PLBs contain dedicated carry logic for fast arithmetic operations with carry chains oriented in columns of the PLB array [3]. Since the configuration memory is partitioned into frames which are also oriented along columns of the array, a column-based BIST architecture (Figure 1) will be the most efficient for partial reconfiguration as well as partial configuration memory read-back during logic BIST since only the columns with ORAs need to be read at the end of the BIST sequence and only the columns with BUTs need to be reconfigured for the subsequent BIST configuration.

We have developed a set of seven BIST configurations that completely test the logic resources of the PLB excluding the LUT-RAM modes of operation (to be discussed in the next section). Figure 4 shows the individual and cumulative single stuck-at gate-level fault coverage obtained with the seven logic BIST configurations. Therefore, seven reconfigurations of the FPGA are required per test session.

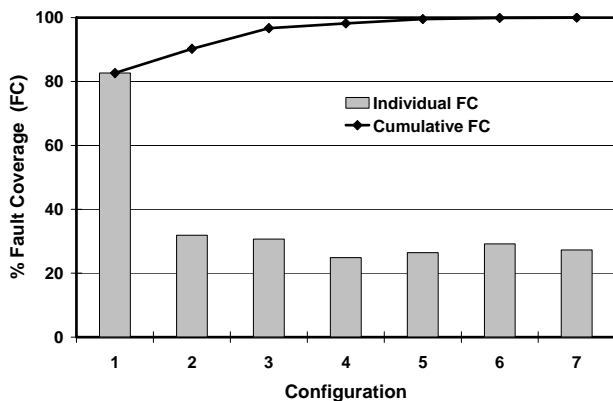


Figure 4. Logic BIST fault coverage

Excluding the arithmetic carry inputs and outputs, there are 12 outputs of each PLB but only eight outputs can

be routed out of the PLB due to limitations in the output multiplexer associated with the PLB. Furthermore, a comparison-based ORA implemented in a single PLB can monitor a maximum of six outputs from its two adjacent BUTs in the logic BIST architecture (Figure 1). In this particular ORA design, three of the 4-input LUTs are used as comparators while the fourth LUT is used in conjunction with its associated flip-flops to logically OR the outputs of the three comparators and the feedback from the flip-flop. Partial configuration memory read-back would be required to retrieve the ORA contents. This ORA requires that the set of seven BIST configurations be applied twice to each BUT while the ORA monitors a different set of six outputs in each case. We refer to each application of the set of BIST configurations to a given set of BUTs as a *slice test set*. When implementing an integrated ORA scan chain (similar to Figure 2), a maximum of only five outputs can be monitored from each adjacent BUT since additional logic resources are needed for the multiplexer that forms the scan chain. Therefore, the set of seven BIST configurations must be applied three times (three slice test sets are required) to completely test the BUTs.

Since the four flip-flops associated with each PLB in a PLB column are mapped into four configuration data frames, potentially there are $M/2-1$ frames, where M is the total number of columns of PLBs that must be read to retrieve the ORA results. However, by restricting the actual flip-flops of the ORAs to the same flip-flop in the same slice, only one frame per PLB column must be read via partial configuration memory read-back. Due to the number of bits per frames partial configuration memory read-back requires about 40 times more clock cycles than that required to retrieve the BIST results using integrated ORA scan chain method. The advantage is that partial configuration memory read-back does not require any additional logic and routing resources and, hence, the extra slice test set can be avoided.

Given that multiple slice test sets must be applied to the BUTs (either two or three depending on the ORA design and its associated results retrieval mechanism), there are multiple approaches to partial reconfiguration of the BUTs. Both slices of the BUTs can be reconfigured during each BIST configuration while the outputs of only one slice is monitored by the ORA which allows two possible scenarios: 1) We can apply the seven BIST configurations to both slices while monitoring the outputs of only one slice and then repeat the seven BIST configurations while monitoring the outputs of the other slice. 2) We can apply the first BIST configuration twice with monitoring the outputs of one slice during the first application and then monitor the outputs of the other slice; this process is repeated for all seven BIST configurations. Since the logic resources in each slice are controlled by multiple configuration memory frames

and a different set of frames for each slice, partial reconfiguration is optimized by reconfiguring only one slice at a time. Therefore, a third scenario would be to reconfigure only the slice under test (for a given slice test set) while maintaining the other slice (not under test) in its original BIST configuration in order to minimize the number of frames that must be reconfigured during each subsequent BIST configuration. A final consideration for partial reconfiguration is that given seven BIST configurations, there may be a particular order that minimizes the number of frames to be reconfigured for the next BIST configuration.

The following methods were investigated in terms of their effect on total test time and the total memory required to store the BIST configurations. While these methods are applicable to RAM and routing BIST, we focus primarily on logic BIST for their introduction.

Method 1: *Full configuration of the FPGA for downloading each BIST configuration and full configuration memory read-back for retrieval of BIST results from the ORAs at the conclusion of each BIST configuration.* This method requires only two slice test sets and, as a result, requires downloading a total of 28 BIST configurations to completely test all PLBs in the FPGA (7 BIST configurations \times 2 slice test sets \times 2 test sessions). This method will represent the normalized total test time and memory storage requirements as well as the baseline for all other comparisons.

Method 2: *Full configuration of the FPGA for downloading each BIST configuration and scan chain retrieval of BIST results from the ORAs at the conclusion of each BIST configuration.* This method uses the integrated ORA scan chain and, as a result, requires three slice test sets for a total of 42 BIST configurations to completely test all PLBs in the FPGA.

Method 3: *Partial reconfiguration for downloading each BIST configuration and scan chain retrieval of BIST results from the ORAs at the conclusion of each BIST configuration.* This method uses the integrated ORA scan chain and, as a result, requires three slice test sets for a total of 42 BIST configurations. In this method, both slices of the BUTs are reconfigured but only one slice under test has its outputs monitored by its associated ORAs for a given slice test set.

Method 4: *Partial reconfiguration for downloading each BIST configuration and scan chain retrieval of BIST results from the ORAs at the conclusion of each BIST configuration.* This method uses the integrated ORA scan chain and, as a result, requires three slice test sets for a total of 42 BIST configurations.

In this method, only the slice under test in the BUTs is reconfigured for the next BIST configuration, that being the slice which has its outputs monitored by the ORAs associated with it.

Method 5: *Optimized order of partial reconfiguration for downloading each BIST configuration and scan chain retrieval of BIST results from the ORAs at the conclusion of each BIST configuration.* This method also requires three slice test sets for a total of 42 BIST configurations. This method is the same as Method 4 with the exception that the order of application of the seven BIST configurations has been optimized such that the minimum number of frames must be reconfigured as we move from one BIST configuration to the next.

Method 6: *Dynamic partial reconfiguration with optimized ordering for downloading each BIST configuration and scan chain retrieval of BIST results from the ORAs at the conclusion of slice test set.* This method also requires three slice test sets for a total of 42 BIST configurations. This method is the same as Method 5 with the exception that the BIST results are retrieved from the ORAs only after all seven BIST configurations have been applied for a given slice test set. This is possible by using dynamic partial reconfiguration which does not corrupt the contents of the flip-flops in the ORA PLBs.

Method 7: *Dynamic partial reconfiguration with optimized ordering for downloading each BIST configuration and partial configuration memory read-back retrieval of BIST results from the ORAs at the conclusion of each BIST configuration.* This method requires one two slice test sets for a total of 28 BIST configurations. This method is the same as Method 5 with the exception that the BIST results are retrieved from the ORAs via partial configuration memory read-back.

Method 8: *Dynamic partial reconfiguration with optimized ordering for downloading each BIST configuration and partial configuration memory read-back retrieval of BIST results from the ORAs at the conclusion of slice test set.* This method also requires one two slice test sets for a total of 28 BIST configurations. This method is the same as Method 6 with the exception that the BIST results are retrieved from the ORAs via partial configuration memory read-back.

These eight methods impact the total test time which includes downloading the BIST configuration, executing the BIST sequence, and retrieval of the BIST results from the ORAs. These eight methods also impact the total memory required to store the BIST configurations.

The Boundary Scan interface (also known as the JTAG interface) was used to access the FPGA. The results include the overhead of all Boundary Scan instructions and operations since partial reconfiguration and partial configuration memory read-back require additional instructions and operations [13][14][15].

The normalized results obtained from actual implementation of the all eight methods on a Spartan II XC2S200 are given in Figure 5. As can be seen in Figure 5, a speed-up in total test time by about a factor of three is obtained by using partial configuration alone (Methods 4, 5, and 6) while the memory required to store the BIST configurations is reduced only by a factor of two. The speed-up in total test time increases to a factor of about five using partial configuration memory read-back while the memory required to store the BIST configurations is reduced by a factor of greater than three. While retrieving BIST results at the end of a slice test set has greater impact on total test time, it should be noted that diagnostic resolution is reduced to the faulty PLB as opposed to the faulty mode of operation in the PLB when results are retrieved after each BIST configuration. It should also be noted that it takes more time to retrieve results when via partial configuration memory read-back as compared to a scan chain.

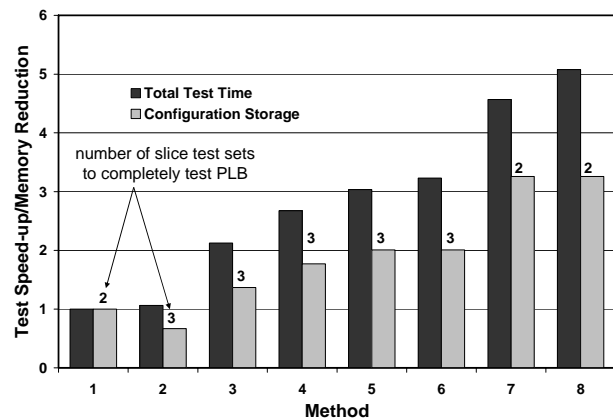


Figure 5. Test time speed-up and reduction in memory storage requirements for different methods

The speed-up in total test time is also a function of the device size. This is illustrated in Table 1 which gives the speed-up for different size FPGAs in the Virtex/Spartan II series using Method 5. As can be seen, a speed-up of about a factor of three is obtained for most devices. The larger speed-up for smaller FPGAs is because the ratio of partial configuration data of PLBs to full configuration data decreases, as the size of FPGA increases due to the fixed number of frames associated with the dedicated *block* RAMs in the Virtex/Spartan II FPGAs. This trend of a slight reduction in test time speed-up as the array size increases holds for all methods that use partial reconfiguration.

Table 1. Test time speed-up versus device size

FPGA	Array size	Speed-up
XC2S15 (smallest)	8 × 12	3.61
XC2S50/XCV50	16 × 24	3.18
XC2S200/XVC200	28 × 42	3.02
XCV1000 (largest)	64 × 96	2.86

4. RAM BIST

There are two kinds of memories in Virtex/Spartan II series FPGAs: 4K-bit dedicated *block* RAMs and distributed RAMs where the LUTs are configured to operate in RAM mode. LUTs in a slice can be configured to operate in three synchronous RAM modes: 16×2 single-port, 32×1 single-port and 16×1 dual-port. The block RAMs can operate in five different sizes in both single-port and dual-port modes: 4K×1-bit to 256×16-bit [3]. The number of BIST configurations required for testing the RAMs and hence number of downloads into FPGA is determined by the number of operational modes.

The architecture used for testing the LUT RAMs is similar to the one used for logic BIST (Figure 1). However, VHDL was used for designing the RAM BIST architecture with placement of the TPGs, ORAs, and RAMs under test controlled via a constraint file. Since different RAM test algorithms were required in each of the RAM BIST configurations as shown in Table 2, the VHDL design flow resulted in a different routing structure (TPG to RAMs under test routing as well as RAMs under test to ORA routing) for each of the BIST configurations. As a result, partial reconfiguration does not result in any significant savings in test time over full reconfiguration since the majority of the frames must be reconfigured. A total of three BIST configurations per test session are required to test the LUT RAMs in the PLBs. Since there are at most four outputs from the PLBs with RAMs under test that must be monitored by the adjacent ORAs, the ORA design can include an integrated scan chain without any impact on the number of BIST configurations that must be applied. In this case, scan chain retrieval of the ORA results is more efficient than partial configuration memory read-back.

In order to take advantage of partial reconfiguration, RAM BIST routing must be controlled to be similar for all RAM BIST configurations. Another approach that can be adopted to decrease test time when utilizing partial reconfiguration is to add redundant circuitry, taking into account all modes of testing, so that transition to next BIST configuration is smoother requiring reconfiguration of a small number of frames. This approach however assumes that there are sufficient logic and routing resources available for adding this redundant circuitry. This approach cannot be applied for LUT RAMs because of lack of sufficient logic resources for implementing redundancy. This approach can, however, be

used for testing block RAMs. Seven BIST configurations are required for completely testing the block RAMs (five single-port and two dual-port test algorithms) [9]. March LR algorithm [17] is used for testing in single-port modes and March s2pf and March d2pf algorithms [18] are used for testing in dual-port mode. While BIST execution times for the different RAM BIST configurations are in the order of tens of microseconds to hundreds of microseconds, the download time for each of the configurations is in the order of a few seconds.

Table 2. March algorithms for RAM BIST

Testing Resource (mode)	March Test	BIST Time
Block RAM (512x16 single-port)	March LR (with BDS) [17]	170 μ s
Block RAM (1024x8 single-port)	March LR (w/o BDS) [17]	84 μ s
Block RAM (512x16 dual-port)	March s2pf [18]	34 μ s
Block RAM (512x16 dual-port)	March d2pf [18]	31 μ s
LUT RAM (16 x2 single-port)	March Y [2]	13 μ s
LUT RAM (32 x1 single-port)	March Y [2]	25 μ s
LUT RAM (16 x1 dual-port)	March DPR [4]	27 μ s

The BIST execution times excluding the download time but including the time taken to execute different march algorithms and time taken to retrieve the BIST results are listed in Table 2. Four of the five single-port tests use the same march algorithm except that each of these configurations traverse a different address space and use different data widths. If an extra address register is added which specifies the maximum address up to which the march sequences have to be generated, all four RAM modes can be tested with single full reconfiguration of the FPGA. First, the single-port RAM BIST configuration which tests in 512x8-bit mode is downloaded and after running the BIST and retrieving the BIST results, partial reconfiguration can be used to test the 1Kx4-bit mode. The TPG has to be reconfigured to count up to 1024 instead of 512 by modifying the contents of the address register. The data width need not be changed as the extra data bits that are generated by the TPG do not affect the operation of the RAMs. Also since the ORA associated with each RAM now compares 4 bits instead of 8 bits, half the bits have to be ignored when retrieving the BIST results. Since the remaining modes and their associated RAM test algorithms are dissimilar, the total number of full reconfigurations to test the block RAMs is four for a test time speed-up of less than a factor of two.

5. ROUTING BIST

The Virtex/Spartan II programmable interconnect architecture consists of local and global routing resources consisting of various length wire segments and programmable interconnect points (PIPs). Four different types of PIPs are found in the routing architecture: break-point PIPs, cross-point PIPs, multiplexer PIPs and switch-box PIPs. A group of switch-box PIPs is referred to as a switch matrix that provides connectivity between the adjacent and non-adjacent PLBs (Figure 6). There are a total of 248 switch-box PIPs in the switch matrix. There are 70 multiplexer PIPs which vary in number of inputs from 4 to 24 for a total of 856 PIPs. Associated with each PLB are a total of 96 Single lines that span 1 PLB, a set of 48 buffered Hex lines that span 6 PLBs, and 12, Long lines that span the width and the length of the PLB array [3]. Table 2 summarizes the total number of routing BIST configurations required to test each resource targeting specific faults.

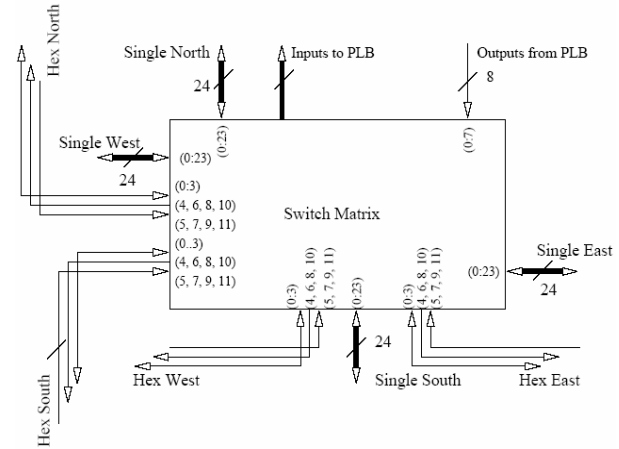


Figure 6. Routing Resources of Virtex FPGA

A comparison-based approach similar to [4] and [10] was used to implement the routing BIST for this investigation. This will detect stuck-at faults as well as opens and shorts at in the wire segments or the programmable interconnect network. The PIPs are also tested for stuck-off and stuck-on faults. Stuck-on PIP faults are sensitized by applying opposite logic patterns at both ends of the PIP while having different ORAs monitoring both ends of the PIP [4][10].

Table 3. Summary of routing BIST configurations

Resource	#/PLB	Faults Detected	Configs
Single lines	96	stuck-at, shorts and opens	24
Hex lines	48	stuck-at, shorts and opens	12
Long lines	12	stuck-at, shorts and opens	2
SwBox PIPs	248	stuck-on and stuck-off	20
MUX PIPs	856	stuck-on and stuck-off	225

Routing BIST configurations were developed using the Xilinx bitstream manipulation tool, JBits [16]. The par-

tial reconfiguration file is generated from a design netlist file of the current routing BIST test phase and full configuration bitstream of previous routing BIST configuration. Thus, the generated partial configuration bitstream contains the data frames differing between the current configuration of the FPGA and the next configuration to be downloaded. For the first routing BIST configuration, a full reconfiguration of the FPGA is required while, for subsequent BIST configurations, only partial reconfiguration with respect to the preceding BIST configuration is required. The regularity of the routing BIST architecture results in a reduction of the total amount of configuration data by as much as a factor of four for some routing BIST configurations when using partial reconfiguration. It should be noted, however, that this does not include the overhead of additional Boundary Scan instructions and operations associated with partial reconfiguration. Also, it should be noted that due to the changes in TPG and ORA locations as well as the change in routing resource under test, the reduction in the amount of configuration data that changes and can be as low as a factor of two between many consecutive routing BIST configurations. Therefore, the overall reduction in configuration data that must be downloaded via partial reconfiguration for all routing BIST configurations will be somewhere between a factor of two and four and can be expected to be closer to a factor of two than four. This ambiguity is due to the fact that only a portion of the 283 routing BIST configurations have actually been implemented at this point in time.

The use of partial configuration memory read-back is more critical in routing BIST than in logic or RAM BIST since the additional logic required for the integrated scan chain reduces the number of wires under test during any given configuration and, hence, increases the total number of routing BIST configurations. Furthermore, the additional routing resources required for the scan chain and shift control often conflict with the wires under test, making the development of the routing BIST configurations more difficult and more likely to result in a larger number of routing BIST configurations. The number of BIST configurations given in Table 2 is based on partial configuration memory read-back and, as a result, the number of routing BIST configurations required when using the integrated ORA scan chain would be higher.

6. SUMMARY AND CONCLUSIONS

A speed-up in total test time can be achieved by using partial reconfiguration and partial configuration read-back. Partial reconfiguration requires that we store only the differences between two consecutive BIST configurations and, therefore, reduces the total memory required to store the BIST configuration. Proper ordering

of the sequence of BIST configurations allows partial reconfiguration to achieve the optimal results in both test time speed-up and reduction in memory storage requirement. Therefore, it would be useful to have a procedure to help determine the optimal order rather than trying all combinations of sequences as was done in our logic BIST investigation. Partial configuration memory read-back requires more time to retrieve BIST results from the ORAs than the integrated ORA scan chain approach but eliminates the need for additional BIST configurations due to logic and routing resource restrictions imposed by the integrated ORA scan chain.

The test time speed-up for Virtex and Spartan II FPGAs obtained with partial reconfiguration and partial configuration memory read-back is most pronounced in logic BIST where the speed-up in test time was shown to be a factor of five. A test time speed-up of factor of two is obtained for RAM BIST and at least a factor of two speed-up is expected for routing BIST. Given that there are a total of 324 BIST configurations (28 for logic BIST, 13 for RAM BIST, and 283 for routing BIST), we can expect a total test time speed-up by a factor of somewhere between two and three and a reduction in the total memory requirements to store the BIST configurations by about the same amount. These results are significant for both manufacturing and in-system testing of FPGAs.

REFERENCES

- [1] M. Abramovici and C. Stroud, "BIST-Based Test and Diagnosis of FPGA Logic Blocks," *IEEE Trans. on VLSI Systems*, Vol. 9, No. 1, pp. 159-172, 2001
- [2] C. Stroud, *A Designer's Guide to Built-In Self-Test*, Kluwer Academic Publishers, Boston MA, 2002
- [3] —, "Virtex Field Programmable Gate Arrays," Product Specification DS003-1, Xilinx, Inc., 2001
- [4] C.E. Stroud, K.N. Leach and T.A. Slaughter, "BIST for Xilinx 4000 and Spartan Series FPGAs: A Case Study," in *Proc. of International Test Conf.*, Vol. 1, pp. 1258-1268, 2003
- [5] M. Abramovici, C. Stroud and J. Emmert, "On-Line Built-In Self-Test and Diagnosis of FPGA Logic Resources," *IEEE Trans. on VLSI Systems*, Vol. 12, No. 12, pp. 1284-1294, 2004
- [6] C. Hamilton, G. Gibson, S. Wijesuriya, and C. Stroud "Enhanced BIST-Based Diagnosis of FPGA via Boundary Scan Access," *Proc. IEEE VLSI Test Symp.*, pp 413-418, 1999
- [7] C. Stroud, J. Sunwoo, S. Garimella, and J. Harris, "Built-In Self-Test for System-on-Chip: A Case Study," *Proc. IEEE International Test Conf.*, pp. 837-846, 2004
- [8] C. Stroud, S. Garimella, and J. Sunwoo, "On-Chip BIST-Based Diagnosis of Embedded Programma-

- ble Logic Cores in System-on-Chip Devices,” *Proc. ISCA International Conf. on Computers and Their Applications*, pp. 308-313, 2005
- [9] S. Garimella and C. Stroud, “A System for Automated Built-In Self-Test of Embedded Memory Cores in System-on-Chip,” *Proc. IEEE Southeastern Symp. on System Theory*, pp. 50-54, 2005
- [10] C. Stroud, J. Nall, M. Lashinsky, and M. Abramovici, “BIST-Based Diagnosis of FPGA Interconnect,” *Proc. IEEE International Test Conf.*, pp. 618-627, 2002
- [11] X. Sun, J. Xu, B. Chan and P. Trouborst, “Novel Technique for BIST of FPGA Interconnects,” *Proc. IEEE International Test Conf.*, pp. 795-803, 2000
- [12] __, “Spartan II 2.5V FPGA Family: Complete Data Sheet,” Product Specification DS001, Xilinx, Inc., 2001.
- [13] __, “Two flows for Partial Reconfiguration: module Based and Difference Based,” Application note: XAPP290, Xilinx, Inc., 2004
- [14] __, “Virtex Series Configuration Architecture User Guide,” Application note: XAPP151, Xilinx, Inc., 2003
- [15] __, “Virtex Series FPGA Configuration and Read-back,” Application note: XAPP138, Xilinx, Inc., 2002
- [16] __, “Xilinx JBits SDK Version 2.8 for Virtex,” Xilinx Inc., 2001
- [17] A. Van de Goor, G. Gaydadjiev, V. Jarmolik and V. Mikitjuk, “March LR: A Test for Realistic Linked Faults,” *Proc. IEEE VLSI Test Symp.*, pp. 272-280, 1996
- [18] S. Hamdioui and A. Van de Goor, “Efficient Tests for Realistic Faults in Dual-Port SRAMs”, *IEEE Trans. on Computers*, vol. 51, no. 5, pp. 460-473, 2002