



BIST for Logic and Memory Resources in Virtex-4 FPGAs

Sachin Dhingra, Daniel Milton, and Charles E. Stroud

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

ABSTRACT: We present a Built-In Self-Test (BIST) approach for testing and diagnosing the programmable logic and memory resources in Xilinx Virtex-4 series Field Programmable Gate Arrays (FPGAs). The resources under test include the programmable logic blocks (PLBs) and block random access memories (RAMs) in all of their modes of operation. The BIST architecture and configurations needed to completely test these resources are presented. The BIST approach exploits architectural and operational features of the Virtex-4 FPGA to reduce the amount of storage required for BIST configuration data as well as the total testing time required for testing the FPGA.¹

1. INTRODUCTION AND BACKGROUND

Built-In Self-Test (BIST) of programmable logic and routing resources in Field Programmable Gate Arrays (FPGAs) has been a topic of research and development for the past decade [1]. The ability to reprogram an FPGA to test itself for fault detection in conjunction with fault diagnosis provides the fundamental step for fault-tolerant operation since the FPGA can be reconfigured to avoid faulty resources. The primary barrier to practical application of this idea has been the large number of BIST and diagnostic configurations required to achieve detection and identification of faulty resources. This barrier is compounded by the time required to download these configurations into the FPGA since the download time represents the major portion of the total testing time [1]. The problem is further confounded by the increase in size and complexity of FPGAs with the addition of specialized cores, including large random access memories (RAMs) and digital signal processors (DSPs) [2].

In this paper, we present a case study of BIST for the Xilinx Virtex-4 series FPGA which exploits architectural and operational features of the FPGA for dynamic partial reconfiguration and configuration memory read-back for BIST and diagnosis of faulty resources based on the BIST results. We begin with a brief overview of the architectural features of the Virtex-4 series FPGAs in Section 2. This is followed by a discussion of the BIST architecture and prior work in BIST for FPGAs, as it relates to this BIST architecture, in Section 3. The

details of BIST configurations for specific logic and memory resources including programmable logic blocks (PLBs) and block RAMs are presented in Sections 4 and 5, respectively. Experimental results are presented in Section 6 and the paper is summarized in Section 7.

2. OVERVIEW OF VIRTEX-4 ARCHITECTURE

The Virtex-4 FPGA is easily the most complex FPGA offered by Xilinx to date. The general architecture is illustrated in Figure 1 and consists of columns of PLBs, block RAMs, and DSPs [3]. The PLBs consist of four slices, each containing two 4-input look-up tables (LUTs) and two flip-flops along with other specialized logic. The LUTs in two of the four slices can also function as small RAMs or shift registers. The block RAMs are 18K-bit dual-port RAMs programmable in a variety of modes of operation including first-in first-out (FIFO) and error correcting code (ECC) modes. The DSPs include an 18x18-bit signed multiplier and a 48-bit adder/subtractor with registers for accumulator operations. The number of rows and columns of PLBs, block RAMs, and DSPs varies with the size and family (LX, SX, or FX) of Virtex-4 FPGAs. The ranges for the total number of PLBs, block RAMs, and DSPs for the various sizes of Virtex-4 FPGAs are included in the legend in Figure 1. The array size for Virtex-4 tend to be large compared to previous FPGAs and these large arrays pose a number of challenges to BIST as will be discussed in the next section. Unlike the example shown in Figure 1, Virtex-4 arrays have more rows than columns with an average ratio of about 2-to-1.

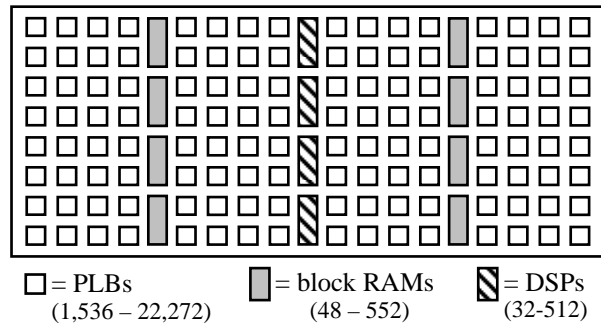


Figure 1. Basic Virtex-4 Architecture

Virtex-4 supports frame addressable write and read access of the configuration memory [4]. This facilitates both dynamic partial reconfiguration of the FPGA via frame addressable write operations and partial configu-

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

ration memory readback via frame addressable read operations. An important new feature in the Virtex-4 is the ability to write multiple frame addresses with the same configuration data once the single frame of configuration data has been written to the Frame Data Register. This multiple frame write feature helps to reduce the total time required to reconfigure the FPGA; this is particularly important in the case of large FPGAs and useful in very regular-structured designs that repeat across the array (as is the case with our BIST approach). During configuration memory readback, the contents of memory elements, including PLB flip-flops/latches and RAMs as well as block RAMs, are also read with the contents of the configuration memory. The frames of the configuration memory are fixed in size and are oriented along the columns of the FPGA with each frame associated with the equivalent of a column of 16 PLBs [4]. Furthermore, the 128 flip-flops for all 16 PLBs in the column are located in the same frame such that their contents can be observed by reading a single frame.

A final architectural feature is the ability to route all primary outputs of the PLB through the flip-flops. All of these architecture and operational features of the Virtex-4 also play a major role in the architecture and operation of the BIST approach as will be discussed in the subsequent sections.

3. BIST ARCHITECTURE

The basic BIST architecture is illustrated in Figure 2 and is used for testing all programmable logic and memory resources in the Virtex-4 series FPGA including PLBs, block RAMs, and DSPs. Multiple identically configured test pattern generators (TPGs) supply test patterns to identically configured blocks under test (BUTs) which can be PLBs, RAMs, or DSPs. The outputs of the BUTs are monitored by two output response analyzers (ORAs) and compared with the outputs of two different BUTs. This results in the circular comparison based BIST architecture shown in Figure 2.

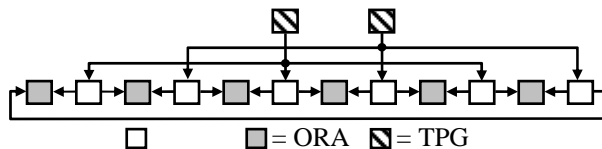


Figure 2. Basic BIST Architecture

This circular comparison architecture was originally developed for testing the 4K-bit block RAMs in Virtex I FPGAs as well as the 18K-bit block RAMs and 18×18-bit multipliers in Virtex II FPGAs [2]. A relatively straight forward diagnostic procedure was developed for this architecture based on the MULTICELLO PLB diagnostic procedure originally developed for an earlier BIST architecture. In that approach, the PLBs along the

edge of the array were monitored by only one ORA [1]. As a result, the diagnostic resolution was the lower along the edges of the array, making unique diagnosis of a set of faulty PLBs difficult and sometimes requiring additional BIST configurations. The MULTICELLO procedure was later extended for use in on-line BIST and PLBs where a sequence of separate test structures with two BUTs and one ORA resulted in a small circular array of four BUTs and four ORAs when superimposed over time [5]. When the diagnostic procedure was extended for use when testing the block RAMs and multipliers in Virtex II, the circular comparison was found to provide an improvement in diagnostic resolution over that of the BIST architecture for which MULTICELLO was originally developed since there are no edges where BUTs are observed by only one ORA [2].

The contents of the ORA flip-flops are obtained via partial configuration memory readback such that no additional logic is required to retrieve the BIST results. Retrieving BIST results via full configuration memory readback was originally used in BIST for Xilinx 4000 series FPGAs [6] to avoid the need for additional logic to form a shift register for scanning out ORA contents at the end of each BIST configuration [1]. Partial configuration memory readback was later used in [7] to reduce the time required to retrieve the contents of the ORAs and, in turn, the overall test time. As a result of the Virtex-4 architecture and the use of partial configuration memory readback, one comparison-based ORA can be implemented using a single LUT and flip-flop such that two independent ORAs can be implemented in each slice. This is illustrated in Figure 3 where each ORA monitors one output from the BUT to the left and to the right of the ORA. Therefore, the total number of slices needed to implement ORAs, N_{ORA} , is given by:

$$N_{ORA} = N_{BUT} \times N_{OUT} \div 2 \quad (1)$$

where N_{BUT} is the total number of BUTs and N_{OUT} is the number of outputs per BUT. A single PLB can implement a total of eight independent ORAs. This fine-grain ORA design also improves the diagnostic resolution by providing the ability to identify the faulty output of the BUT in addition to identifying the faulty BUT itself [2]. The ORAs are physically located in common columns of PLBs in the FPGA in order to minimize the number of frames that must be read when using partial configuration memory readback to obtain the BIST results. With the frame structure of the Virtex-4 and the fact that all flip-flops for a column of 16 PLBs are located in a single frame, the total number of frames, N_{FRAME} , that must be read to obtain the BIST results is given by:

$$N_{FRAME} = (R \div 16) \times (C \div 2) = R \times C \div 32 \quad (2)$$

where R and C are the number of rows and columns, respectively, of PLBs in the array. Using dynamic partial reconfiguration, where the FPGA can be reconfig-

ured without destroying the contents of the ORA flip-flops, the BIST results can be read only once at the end of a sequence of test configurations instead of at the end of each BIST configuration. This can significantly reduce the total number of frames that must be read during the application of BIST. This ORA readback at the end of a set of BIST configurations will not result in loss of fault detection capabilities since the ORA flip-flops will retain failure indications from one BIST configuration to the next. However, it does result in some loss of diagnostic resolution; the faulty PLBs can still be determine but the faulty mode of operation cannot.

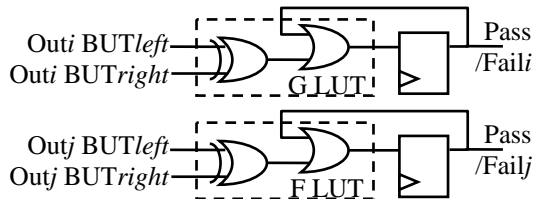


Figure 3. Two ORAs Implemented in One Slice

In the original BIST approaches for FPGAs that consisted of an array of PLBs only, the TPGs were by necessity implemented in PLBs [1][5][6]. A later BIST approach for FPGA cores in System-on-Chip (SoC) devices used an embedded processor core to implement the TPG function to test RAMs in the FPGA core [8]. With the additional cores (RAMs and DSPs) available in the Virtex-4, the TPGs can be implemented using a variety of resources, as summarized in Table 1, depending on the resources, BUTs, being tested. This is particularly important in the case of BIST for the PLBs since the implementation of TPGs using the DSPs frees PLBs for implementing ORAs and the circular comparison BIST architecture.

Table 1. Resources Used for BIST

BUTs	TPGs	ORAs	# Sessions
PLBs	DSPs	PLBs	2
LUT RAMs	DSPs & Block RAMs	PLBs	1
Block RAMs	PLBs	PLBs	1
DSPs	PLBs	PLBs	1

As noted in Table 1, the ORAs are always implemented in PLBs due to the large number of ORAs required (as given by Equation 1). The number of TPGs, on the other hand, is typically much smaller, one or two TPGs in prior BIST approaches for FPGAs. In approaches that used only one TPG, the assumption was made that the resources used to implement the TPG had already been tested and found to be fault-free [2][8]. Another assumption for the use of a single TPG is that only pathological cases existed that would cause a faulty TPG to skip all of the test patterns that would detect faulty BUTs [5]. Two TPGs have been more frequently used since a faulty TPG would apply different test pat-

terns to its BUTs which, in turn, would result in failure indications in all ORAs that observe those BUTs and compare those BUT outputs to those of BUTs driven by the other TPG [1][6]. In most previous cases, the array size has been sufficiently small to facilitate acceptable loading on each TPG. During BIST of PLBs, the number of loads on each TPG, T_{LOAD} , is given by:

$$T_{LOAD} = N_{BUT} = R \times C \times S \div 4 \quad (3)$$

where S is the number of slices per PLB. The previously proposed solution for large arrays has been to divide the array into quadrants with the BIST circuitry, including two TPGs, implemented in each quadrant [1][6][7]. This reduces the loading on each TPG by a factor of 4. The Virtex-4 FPGAs are very large with the largest (the LX200) consisting of 192 rows and 116 columns of PLBs [3]. This would require each TPG to drive 22,272 loads (from Equation 3) and even dividing the array into quadrants, each TPG would have 5,568 loads given four slices per PLBs. Our solution to these large array sizes is to implement BIST circuitry, including two TPGs in each set of four rows of PLBs in the array, as illustrated in Figure 4. The decision to form groups of four rows is based in part on the fact that there are a minimum two DSPs and three block RAMs per group of four rows of PLBs in Virtex-4. As a result, the TPG loading is now given by:

$$T_{LOAD} = 4 \times C \times S \div 4 = C \times S \quad (4)$$

This results in each TPG driving a maximum of 464 loads in the largest Virtex-4 device (LX200) – an order of magnitude improvement. Loading on the TPGs when testing block RAMs and DSPs is not a problem since there are fewer of these cores. Each TPG would drive a maximum of 192 loads when testing these cores.

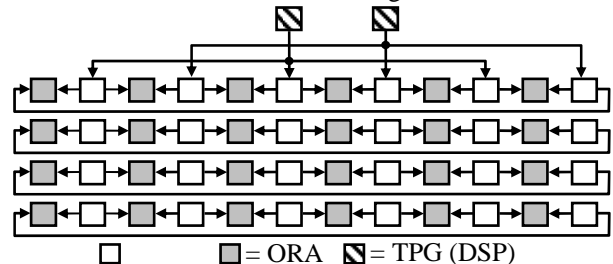


Figure 4. Four Row BIST Structure

The BUTs are repeatedly reconfigured in all of their modes of operation and this set of BIST configurations is referred to as a test session [1]. The number of test sessions required to test each type of BUT is included in Table 1. During each BIST configuration in a test session, only the BUTs are reconfigured while the TPGs and ORAs remain unchanged. This helps to minimize the number of frames that must be written during partial reconfiguration of the BIST configurations. A more important factor in minimizing the number of frames that must be written is to maintain the same TPG-to-

BUT and BUT-to-ORA routing from one BIST configuration to the next. Changes in routing can result in a significant increase in the number of frames that must be written during partial reconfiguration since over 80% of the frames are associated with configuring routing resources. With configuration memories ranging from 4.7 million to 50.8 million bits for the smallest to largest Virtex-4 devices, any increase in the number of frames to be written will have serious implications on the total testing time. Therefore, we attempt to maintain fixed routing from one BIST configuration to the next.

Partial reconfiguration bit files are generated by the BitGen software in the Xilinx ISE tool suite. This software will compare the file to be downloaded with the reference file for the implementation residing in the FPGA. It then generates a file containing the configuration frame data for those frames that change between the two implementations along with the appropriate commands to the configuration registers to perform dynamic partial configuration. In addition, the BitGen software takes advantage of the multiple frame write capability in Virtex-4 by grouping the addresses of all frames that will be written with identical configuration data [4]. During the reconfiguration process, the starting frame address is written followed by the configuration data for that frame. Once the configuration data is loaded in the Frame Data Register, the frame address for each frame to be written with the same configuration data is given followed by the multiple frame data write instruction, without the need to rewrite the frame data register. This is particularly valuable in the case of BIST since we are only changing the configuration of the BUTs and, more importantly, all BUTs are configured identically. The effects of partial reconfiguration and the multiple frame data write feature will be shown in the experiment results of Section 6. We now turn our attention to the details of the BIST for each type of BUT.

4. BIST FOR PLBs

The Virtex-4 PLB consists of four slices of two different types, referred to as SliceM and SliceL [3]. Each slice contains two 4-input LUTs, two flip-flops/latches and additional control logic and carry chain circuitry as illustrated in Figure 5. The difference between SliceM and SliceL is that each LUT in SliceM can also function as a 16-bit shift register or a 16-bit RAM. As a result, there is additional logic in SliceM to perform the shift register and RAM functions which will require additional BIST configurations.

4.1. Testing PLB Logic Resources

The PLBs in the FPGA array are partitioned into alternate columns of BUTs and ORAs and, as a result, two test sessions are required to test all of the PLBs in the array. During the second test session, the positions of

the BUTs and ORAs are swapped. The Virtex-4 PLB architecture allows all primary outputs of the PLB to be routed through the flip-flops such that the logic resources can be tested while maintaining fixed routing for the BUT-to-ORA connections. While this could result in a slight increase in the number of BIST configurations for the PLBs, it reduces the overall testing time due to more efficient partial reconfiguration when the multiple frame data write capability is used.

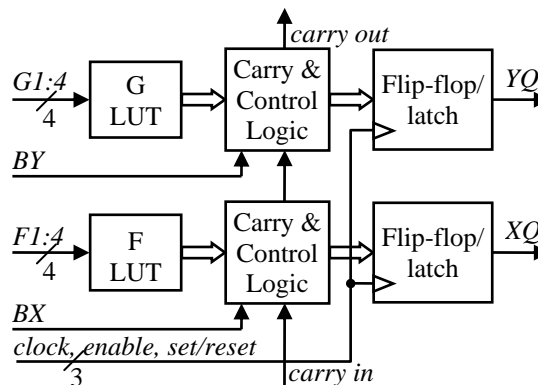


Figure 5. Virtex-4 Slice

Two DSPs per four rows of PLBs are configured in an accumulator mode of operation to function as the TPGs and drive alternate columns of BUTs in those four rows. These accumulators are initialized to all 0s and then a constant value 0x691 is added during each clock [9]. Since there are only twelve inputs to each slice, only the twelve LSBs of the 48-bit accumulator are connected to the slices. While the DSP accumulator-based TPG does not generate pseudo-random test patterns like a Linear Feedback Shift Register (LFSR) with primitive polynomial, the accumulator does produce an exhaustive set of test patterns in with more transitions in each bit position than a counter [9]. Furthermore, the exhaustive set of 4,096 test patterns is generated in 2^{12} clock cycles.

In each test session, a total of twelve BIST configurations are needed to test the logic resources in the SliceM (excluding the LUT RAM modes of operation). The last two configurations in the set of twelve are used to test the LUTs in their shift register mode of operation. A total of ten BIST configurations are needed to test the logic resources in the SliceL. As a result, all four slices in each BUT can be tested concurrently in twelve BIST configurations. Therefore, a total of 24 BIST configurations (2 test sessions of 12 configurations each) are required to test all PLBs in the Virtex-4, independent of the array size. These BIST configurations also test the dedicated routing resources between the slices, such as the carry routing shown in Figure 5. The individual and cumulative fault coverage for the entire PLB (four slices including routing but excluding LUT RAM logic) obtained with the set of twelve BIST configurations are summarized in Figure 6.

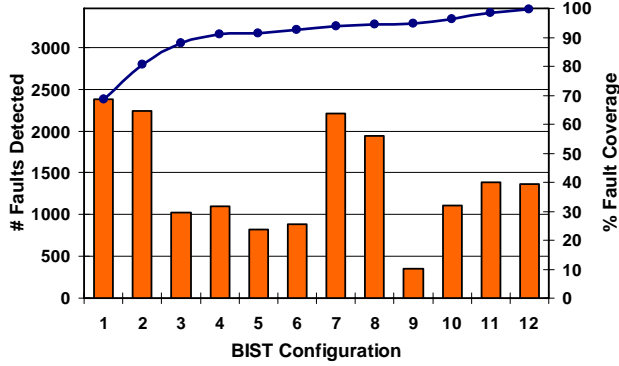


Figure 6. PLB Fault Coverage (4 slices)

After the download of the first BIST configuration only the BUTs are reconfigured to function in different modes of operation using partial reconfiguration. Thus, the first full configuration is followed by nine partial reconfigurations to test the logic resources excluding the shift register mode of operation. These nine configurations can be performed in a particular order to minimize the number of frames to be written. However, we observed that the improvement achieved with optimizing the reconfiguration order in Virtex-4 is negligible compared to the improvement that was achieved in Virtex [7]. This is primarily due to the multiple frame data write feature in Virtex-4.

When testing the PLBs in shift register mode, the X and Y BUT outputs are monitored by the ORAs instead of the XQ and YQ outputs, as was the case in the first ten configurations. This changes the BUT-to-ORA routing which, in turn, forces a full configuration download for one configuration. This full configuration is followed by one partial reconfiguration. Therefore, for each test session, there are two full configurations and ten partial reconfigurations of the FPGA.

Timing analysis of the fastest and slowest BIST configurations, in terms of maximum clock frequency, was performed for various sizes of Virtex-4 FPGAs. The results are summarized in Figure 7 where it can be seen that the maximum BIST clock frequency is generally a function of the number of columns in the device. This is due to the grouping of four rows of PLBs into an independent BIST circuit which eliminates the effect of the number of rows in the FPGA on the maximum BIST clock frequency. However, this was only obtained after breaking the carry chain shown in Figure 5 such that alternating rows of PLBs select the *CARRY-IN* input while the remaining rows select the *BX* input. This selection is swapped in subsequent BIST configurations such that the *CARRY-IN* input (logic and routing) is completely tested without creating a critical timing path. As an example, the maximum clock frequency for a Virtex-4 XC4VLX25-10 went from 40 MHz to 140 MHz with this modification to the carry chain testing.

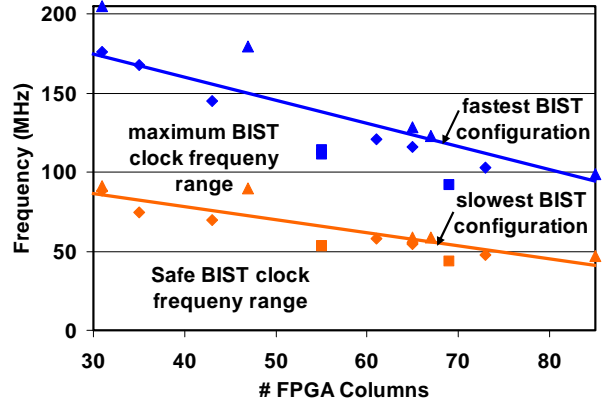


Figure 7. Maximum Clock Frequency for Fastest and Slowest BIST Configurations vs. Device Size

4.2. Testing LUT RAMs

The BIST circuitry for testing the LUT RAMs is illustrated in Figure 8. The LUT RAMs can be tested in a single test session since only half of the slices (SliceM) of the PLB contain LUTs which can operate as RAMs. As a result, the other half of the slices (SliceL) can be used for ORAs. Each TPG is constructed from a block RAM (used as a ROM to store test patterns) and a DSP (used as a counter to address the block RAM). We perform a circular comparison with all LUT RAMs under test being monitored by two ORAs and compared with two different LUT RAMs under test. An alternative approach is to compare the LUT RAM outputs with expected results stored in the TPG block RAM.

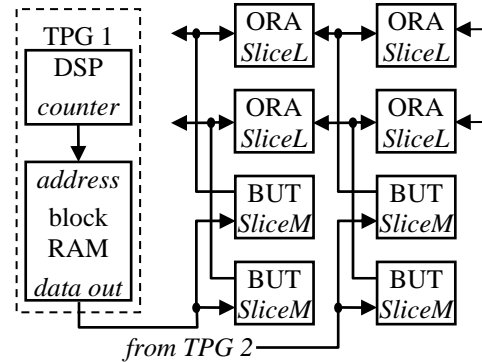


Figure 8. LUT RAM BIST Structure

For every four rows of PLBs there are two DSPs which are used to sequence through the addresses of two block RAMs in the same set of four rows. The test patterns read from each of the block RAMs are supplied to the LUT RAMs in alternating columns of PLBs in that set of four rows to limit the loading on the TPGs in the large Virtex 4 FPGAs. The LUT RAM BIST configurations and the RAM test algorithm used in each configuration are summarized in Table 2. In addition, the number of test patterns needed for each test is given which corresponds to the number of address locations needed

in the block RAM. The block RAM is configured in the 1K×18-bit mode with the test vectors for each BIST configuration loaded in the block RAM during partial reconfiguration of the BUTs. The LUT RAMs can also operate in a 16×4 single-port mode but this mode does not need to be tested since the circuitry is tested by the other three BIST configurations, as illustrated by the individual and cumulative fault coverage for the three BIST configurations given in Table 2. The March DPR test is the same algorithm that was originally developed for testing the dual-port RAM mode in the LUT RAMs of the Xilinx 4000 series FPGA [6].

Table 2. BIST Configurations for LUT RAMs

Configuration	1	2	3
RAM Mode	64×1 single-port	32×1 single-port	16×2 dual-port
Test Algorithm	March Y	March Y	March DPR
# Test Patterns	512	256	624
Individual FC	94.5%	77.3%	76.4%
Cumulative FC	94.5%	96.4%	100%

5. BIST FOR BLOCK RAMS

All of the block RAMs can be tested concurrently even with PLBs used to implement the TPG and ORA functions. There are sufficient PLBs in all families and sizes of Virtex-4 FPGAs to facilitate testing the block RAMs concurrently. A total of ten BIST configurations, summarized in Table 3, are required to test the 18K-bit block RAMs in Virtex-4. Table 3 includes the RAM sizes for each configuration in terms of the number of address locations (*A*) and data width (*D*) associated with each RAM mode of operation as well as the test algorithm used and the number of clock cycles required for execution of the BIST sequence. The number of slices in Virtex-4 needed to implement the TPG and ORA functions is included at the bottom of the table. Note that BDS indicates the application of background data sequences for detecting pattern sensitivity and coupling faults for word oriented memories [10]. Further details of the RAM BIST configurations will be discussed in Section 5.2.

While only eight BIST configurations were required to test the block RAMs in Virtex II FPGAs [2], Virtex-4 block RAMs provide additional modes of operation including a number of FIFO and ECC modes which incorporate Hamming circuitry for ECC applications [3]. For Virtex-4, the total set of BIST configurations are partitioned into two test sessions. The first session (configurations 1 through 5) tests the block RAMs in single and dual port modes while the second session (configurations 6 through 10) focuses on testing the block RAMs in the FIFO modes of operation using the algorithm described in [11].

Partial reconfiguration is used to decrease the amount time needed to test the block RAMs. At the beginning of each test session, the initial download configures the TPGs, ORAs, routing resources, and the initial block RAM configuration for the test session. Subsequent reconfigurations only modify the configuration of the block RAMs. During the ten BIST configurations, other programmable modes of operation can be tested, including active edges clocks, active levels of clock enables and resets, registered inputs and outputs, and the various write/read options.

Table 3. BIST Configurations for Block RAMs

BIST Con-fig	Test Algorithm	Address Locations (<i>A</i>)	Data Width (<i>D</i>)	V-4 Clock Cycles	V2P Clock Cycles
1	March LR w/ BDS	512	36	58× <i>A</i>	58× <i>A</i>
2	MATS+	8K	2	2×5× <i>A</i>	2×14× <i>A</i>
3		16K	1	2×5× <i>A</i>	2×14× <i>A</i>
4	March s2pf-	512	36	14× <i>A</i>	14× <i>A</i>
5	March d2pf	512	36	9× <i>A</i>	9× <i>A</i>
6	FIFO	4K	4	6× <i>A</i>	14× <i>A</i>
7		2K	9	6× <i>A</i>	14× <i>A</i>
8		1K	18	6× <i>A</i>	14× <i>A</i>
9		512	36	6× <i>A</i>	-
10	FIFO ECC	512	64	3× <i>A</i>	-
TPG slice count = 608 ORA slice count = 72× <i>N</i> where <i>N</i> = # block RAMs				Total 334,848	Total 829,952 [2]

5.1 Block RAM BIST Architecture

The block RAM BIST architecture is dependent on the overall architecture of the Virtex-4. The entire architecture is outlined in Figure 9. The BIST architecture must be compatible with all sizes and families of Virtex-4. There are at least four columns of PLBs on either side of the center column of the array. The smallest device contains 64 rows which yields a total of 1024 slices available per TPG which is more than enough to place and route the 608 slices required for each TPG (from Table 3). The two TPGs drive alternating rows of block RAMs. After each column of block RAMs (as we move out from the center) there is a minimum of four columns of PLBs before encountering the next column of block RAMs. Each block RAM spans four rows of PLBs, thus enabling a 4×4 array of PLBs to implement ORAs. At most, there are 72 ORAs needed to monitor the block RAM outputs which equates to nine PLBs per block RAM for implementing ORAs. Adjacent block RAMs in the same column are monitored by neighboring sets of ORAs. The remaining connections at the tops and bottoms of the columns needed to complete the ORA circular comparison are shown in Figure 9.

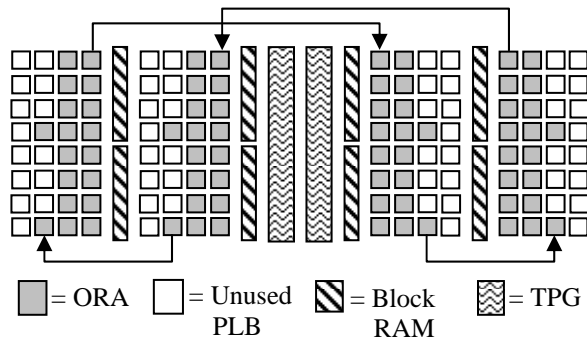


Figure 9. Block RAM BIST Architecture

5.2 Test Algorithms

Virtex-4 block RAMs share many similarities with the block RAMs in Virtex II, but the Virtex-4 block RAMs add more functionality with built-in FIFO and ECC support in certain configurations. As such, many of the testing algorithms previously used for Virtex-II [2] are applied to Virtex-4 block RAMs. For example, the March LR algorithm with BDS is used to verify that the RAM matrix (containing the memory cells) is fault-free. March LR has been shown to be a superior test to March C-, while only increasing the test complexity marginally [12]. A procedure is given in [10] for converting March LR to a word-oriented test by incorporating BDS. The two widest memory configurations (512×36-bit and 1K×18-bit) have the ability of writing byte-wise to the memory locations. After March LR with BDS testing, the RAM matrix is assumed to be fault-free if the ORAs did not detect any mismatch during the circular comparison. As a result, BDS is applied only during the first BIST configuration since once tested there is no need to repeat pattern sensitivity and coupling fault tests in the RAM matrix. The next test algorithm, MATS+ [13], is used to test for address decoder faults in the programmable address decoder. MATS+ is the most efficient test known to detect all address decoder faults. This helps to reduce the overall test time associated with the block RAMs, as can be seen at the bottom of Table 3 where the total test time, in terms of BIST clock cycles, is compared with that for similar block RAMs in Virtex II using the BIST approach described in [2]. In that BIST approach, the March LR algorithm was used to test all single-port RAM modes of operation and, as a result, required about 2.5 times more BIST execution clock cycles compared to what we have currently developed for Virtex-4.

Dual-port testing is achieved by using March s2pf- and March d2pf [14]. This approach was also used in [2]. Virtex-4 block RAMs also support cascading two adjacent block RAMs (either above or below) together to create a 32K×1-bit RAM configuration. Functional testing will serve to test the resources in this mode since the cascading is achieved by configuring two block

RAMs as 16K×1-bit RAMs and manipulating multiplexers to direct information as needed. This implementation does not need to be tested by a thorough March test; only a functional test is needed to examine the expected data flow in the cascading circuitry. It should be noted, however, that the entire device must be reconfigured for this single test so that the ORAs can be configured to monitor every other block RAM.

The FIFO test algorithm described in [11] begins with an initially empty FIFO which is written with 1s and then read until empty. Testing with opposite logic values is also done in a similar fashion, and the programmable almost full and almost empty flags are tested as well by repeatedly reconfiguring the almost full flag to a higher value and then writing more data. Likewise, the almost empty flag must be reconfigured during the emptying of the FIFO.

The Virtex-4 block RAMs also can be configured to a 512×64-bit ECC RAM or FIFO by using two adjacent block RAMs [3]. Hamming code is generated for the incoming data and stored in the RAM using the four parity bit locations in each RAM. When an address location is read, the Hamming code is regenerated for the outgoing data and compared with the stored Hamming bits from the write operation. Single-bit errors are corrected at the RAM output while non-correctable double-bit errors are flagged. Testing the Hamming code generation and bit error correction circuitry presents the interesting problem of testing for faults in a fault-tolerant circuit. Fortunately, Virtex-4 block RAMs can be initialized at configuration time and we can exploit this feature by initializing memory locations with data so that the ECC circuit will correct single-bit correctable errors as well as detect and flag non-correctable errors during an initial read cycle of all the memory location in the RAM. This will test the Hamming code generation as well as the error detection and correction circuitry at the output of the RAM. The Hamming code generation circuitry at the input of the RAM can be tested by simply writing a sequence of 64 data words in which we walk a 1 through a field of 0s and then reading those address locations while monitoring the Hamming error flags at the output; if the Hamming code was generated correctly there will be no errors detected.

5.3 Test Pattern Generation

To maximize the test time speed-up with partial reconfiguration between RAM BIST configurations, there needs to be as few differences as possible between each test configuration. One way to achieve this is to have a static TPG that remains constant for multiple block RAM BIST configurations. By making the TPG static, the only changes between BIST configurations will be the block RAMs. Likewise, the ORAs will monitor outputs of the block RAMs even if they are not active

during a given configuration. For example, if the RAM is configured as a 1K×18-bit, then there would be 36 ORAs monitoring both ports. If the RAM is configured as a 16K×1-bit, then only 2 ORAs are needed. The remaining 34 ORAs in this example are still present and their routing unchanged, however, their values are a “don’t care” since they may not observe valid data. Since all block RAMs are configured identically, the inactive outputs should behave identically under fault-free conditions.

The current TPG design incorporates a state machine that is able to generate several RAM test algorithms sequentially and by user request. Previously 1024 slices was determined to be a limit for a column height TPG. The slice count for our TPG is 608 slices. This TGP is able to generate tests for March LR with BDS, MATS+, March s2pf-, and March d2pf. Figure 10 shows the slice utilization for this TPG and the corresponding ORAs (from Equation 1). Clearly, from this figure one can observe that RAM BIST does not utilize an appreciable amount of PLBs from its implementation. Two more TPGs will also be needed to generate tests for FIFO and ECC functionality, and a simple TPG will be developed to test cascaded block RAMs.

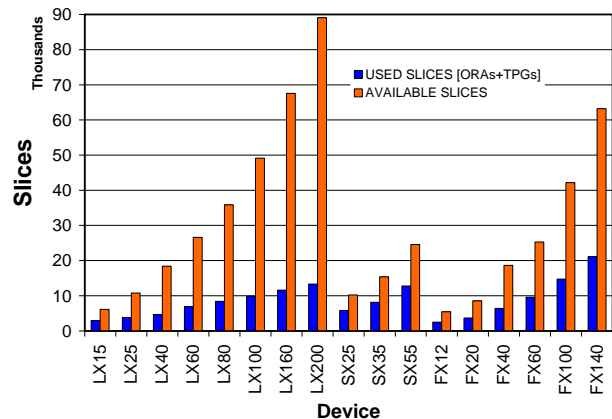


Figure 10. RAM BIST Device Utilization

5.4 BIST Implementation

The TPGs are implemented by synthesizing a VHDL model and constraining the placement and routing of the slices to the required area within the FPGA. The remaining connectivity is implemented using Xilinx Design Language (XDL), a Xilinx netlist format for describing designs at the slice-level with complete control over placement and routing. This allows us to integrate the TPGs, implemented using a regular design flow, with the ORAs and the block RAMs whose XDL has been generated by a parameterized C program. With the exception of the block RAM TPG implementation, the process of developing parameterized C programs to generate BIST configurations is also used for the PLB

and LUT RAM BIST. The parameterized C programs facilitate generation of BIST configurations for any size and family of Virtex-4 FPGAs.

6. EXPERIMENTAL RESULTS

The two primary costs associated with BIST for FPGAs include the total test time and the total amount of memory required to store the BIST configurations. The testing time is typically dominated by the time needed to download the BIST configurations into the FPGA. Therefore, use of dynamic partial reconfiguration and partial reconfiguration memory readback provide improvements to both test time and memory storage requirements. The following methods were used to determine the speed-up in testing time and the reduction in memory storage requirements when using the BIST in conjunction with partial reconfiguration and readback.

Method 1: Full configuration of FPGA followed by full configuration memory readback to retrieve BIST results after every BIST configuration. This approach serves as the baseline for the worst case test time.

Method 2: Partial reconfiguration of FPGA followed by full configuration memory readback to retrieve BIST results after every BIST configuration. This approach serves to illustrate the affect of partial reconfiguration alone.

Method 3: Partial reconfiguration of FPGA followed by partial configuration memory readback to retrieve BIST results after every BIST configuration.

Method 4: Dynamic partial reconfiguration of FPGA followed by partial configuration memory readback to retrieve BIST results at the end of the test session. The ORAs retain the BIST results from one configuration to the next and are read only at the end of the test session. This further reduces the total test time at the expense of a slight loss in diagnostic resolution.

The test time speed-up and reduction in memory storage requirements are summarized in Table 4 for the PLB BIST and in Table 5 for the LUT RAM BIST. In Table 4, the test time speed-up and memory storage reductions are compared with those obtained for Virtex and reported in [7]. As can be seen, the test time speed-up obtained for Virtex-4 is about 2.5 times better than that obtained in Virtex, with better memory reduction also.

Table 4. Test Time Speed-up and Memory Storage Reduction for Logic BIST on Virtex-4 and Virtex

Method	Test Time Speed-up		Memory Reduction	
	Virtex [7]	Virtex-4	Virtex [7]	Virtex-4
1	1	1	1	1
2	-	1.4	-	5.3
3	4.6	8.9	3.2	5.3
4	5.1	12.9	3.2	5.3

Table 5. Test Time Speed-up and Memory Storage Reduction for LUT RAM BIST

Method	Test Time Speed-up	Memory Reduction
1	1	1
2	1.3	2.8
3	4.8	2.8
4	6.2	2.8

We have downloaded and verified all of the BIST configurations we have developed on LX25, LX60, and SX35 devices. It is clear that the Virtex-4 architecture aids in the improvement of test time and reduction in memory storage requirements. Multiple frames with the same contents can be written in succession without re-loading the frame data during partial reconfiguration. This was not possible in Virtex FPGAs, where the frame data is loaded for every frame to be written. The configuration memory of Virtex-4 is better organized for partial reconfiguration as configuration bits for similar components are grouped together in frames. The fact that a single frame contains the contents of all the flip-flops in a column of 16 PLBs speeds up the BIST results retrieval when using partial configuration memory readback.

7. SUMMARY

The Virtex-4 FPGA poses a number of new testing challenges due to its large size and new specialized cores. As a result of its large array, previous work in BIST for FPGAs that used only two TPGs would incur excessive loading on the TPG outputs which, in turn, would result in a slow operating clock speed for BIST. In Virtex-4, we have allocated a pair of TPGs for every four rows of PLBs when testing the logic and memory resources in the PLBs. This greatly reduces the loading on the TPG outputs as only the BUTs in four rows of the FPGA are driven by a pair of TPGs. The TPG loading for the block RAM BIST is much lower than that of the PLB BIST, even when only two TPGs are used. The incorporation of circular comparison in all BIST, including that for PLBs, provides improved diagnostic resolution over most previous BIST approaches.

BIST was developed and implemented to test the programmable logic and memory resources of all families of Virtex-4 devices. A much more pronounced speed-up in test time and reduction in memory storage requirements was observed for Virtex-4 over previous Virtex FPGAs. This is due to the architectural and operational features provided by Virtex-4 in the way configuration memory is organized and the way reconfiguration can be performed. As a result, the regularity of the BIST architecture can be designed to exploit these features in order to obtain maximum gains from dynamic partial reconfiguration and partial configuration memory readback.

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 and S. Garimella, "BIST and Diagnosis of Multiple Embedded Cores in SoCs," *Proc. International Conf. on Embedded Systems & Applications*, pp. 130-136, 2005
- [3] ___, "Virtex-4 User Guide," UG070 (v1.4), Xilinx, Inc., 2005, (available at www.xilinx.com)
- [4] ___, "Virtex-4 Configuration Guide," UG071 (v1.4), Xilinx, Inc., 2005. available at www.xilinx.com
- [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. Stroud, K. Leach, and T. Slaughter, "BIST for Xilinx 4000 and Spartan Series FPGAs: A Case Study," in *Proc. IEEE International Test Conf.*, pp. 1258-1268, 2003
- [7] S. Dhingra, S. Garimella, A. Newalkar, and C. Stroud, "Built-In Self-Test of Virtex and Spartan II Using Partial Reconfiguration," *Proc. IEEE North Atlantic Test Workshop*, pp. 7-14, 2005
- [8] C. Stroud, S. Garimella and J. Sunwoo, "On-Chip BIST-Based Diagnosis of Embedded Programmable Logic Cores in System-on-Chip Devices," *Proc. ISCA International Conf. on Computers and Their Applications*, pp. 308-313, 2005
- [9] S. Gupta, J. Rajski, and J. Tyszer, "Test Pattern Generation Based on Arithmetic Operations," *Proc. IEEE International Conf. on Computer-Aided Design*, pp. 117-124, 1994
- [10] A. van de Goor, I. Tlili, and S. Hamdioui, "Converting March Tests for Bit-Oriented Memories into Tests for Word-Oriented Memories," *Proc. IEEE International Workshop on Memory Technology Design and Testing*, pp. 46-52, 1998
- [11] ___, "Compiled Megacell Testing," Application Note 0696C, Atmel Corp., 1999, available at www.atmel.com
- [12] 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
- [13] V. Agrawal and M. Bushnell, *Essentials of Electronic Testing: for Digital Memory & Mixed-Signal VLSI Circuits*, Kluwer, 2000.
- [14] S. Hamdioui, *Testing Static Random Access Memories*, Springer, 2004