

# BUILT-IN SELF-TEST FOR MEMORY RESOURCES IN VIRTEX-4 FIELD PROGRAMMABLE GATE ARRAYS

Brooks R. Garrison, Daniel T. Milton, and Charles E. Stroud

Department of Electrical and Computer Engineering

Auburn University

Auburn, AL 36849-5201, USA

garribr@auburn.edu strouce@auburn.edu



**ABSTRACT:** We present a Built-In Self-Test (BIST) approach for programmable embedded memories in Xilinx Virtex-4 Field Programmable Gate Arrays (FPGAs). The target resources are the block random access memories (RAMs) in all of their modes of operation including single- and dual-port RAM, first-in first-out (FIFO), error correcting code (ECC), and cascade modes of operation. The BIST architecture and configurations needed to test these block RAMs are presented with implementation, fault detection, and timing analysis.<sup>1</sup>

## 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 over a decade [1]. The ability to reprogram an FPGA in-system to test itself for fault detection and diagnosis provides a fundamental step for fault-tolerance since FPGAs can be reconfigured to avoid faulty resources. The primary barrier to practical application has been the large number of BIST and diagnostic configurations required to achieve detection and identification of faulty resources. This 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 compounded by the increasing 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 Xilinx Virtex-4 series FPGA which exploits architectural and operational features of the FPGA for dynamic partial reconfiguration and, when needed, configuration memory readback for BIST and diagnosis of faulty resources based on the failing BIST results. We begin with a brief overview of architectural features of Virtex-4 FPGAs in Section 2. This is followed in Section 3 by an overview of our general BIST architecture and related prior work in BIST for FPGAs. An overview of Virtex-4 block RAMs is given in Section 4 along with details of the BIST configurations needed to test specific block RAM modes of operation. Experimental results including implementation and analysis of fault detection and timing capabilities are presented in Section 5. Finally, the paper is summarized and concludes in Section 6.

<sup>1</sup> This work was sponsored by the National Security Agency under contract H98230-04-C-1177 and supported in part by the National Science Foundation Grant CNS-0708962.

## 2. OVERVIEW OF VIRTEX-4 ARCHITECTURE

The general architecture of the Virtex-4 FPGA is illustrated in Figure 1 and consists of columns of configurable logic blocks (CLBs), programmable input/output blocks (IOBs), block RAMs, and DSPs with the ranges in the total number of elements included in the legend in the figure [3]. The number of rows and columns of CLBs, IOBs, block RAMs, and DSPs varies with the size and family (LX, SX, or FX) of Virtex-4 FPGAs. Unlike the example illustrated in Figure 1, Virtex-4 arrays have more rows than columns with an average ratio of about 2.3:1.

Virtex-4 supports frame addressable write and read access of the configuration memory [8]. This facilitates both dynamic partial reconfiguration of the FPGA via frame addressable write operations and partial configuration memory readback via frame addressable read operations. An important new feature in 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 CLB 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 corresponding to the equivalent of a column of 16 CLBs [8]. Furthermore, the 128 flip-flops for all 16 CLBs in the column are located in the same frame such that their contents can be observed by reading a single frame. All of these architecture and operational features of Virtex-4 also play a major role in the architecture and operation of the BIST approach as will be discussed in the subsequent sections.

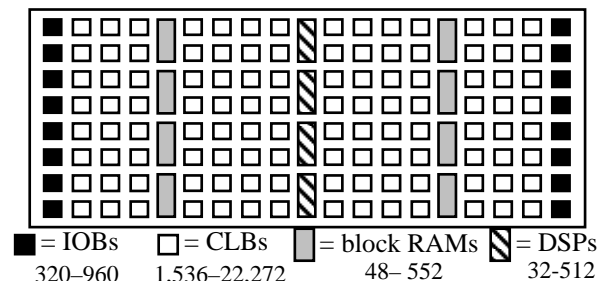


Figure 1. Basic Virtex-4 Architecture

### 3. OVERVIEW OF BIST ARCHITECTURE

Our basic BIST architecture is illustrated in Figure 2 and is used for testing all programmable logic and memory resources in Virtex-4 series FPGAs including CLBs, IOBs, block RAMs, and DSPs. Multiple test pattern generators (TPGs) supply identical test patterns to identically configured blocks under test (BUTs) which can be CLBs, IOBs, block RAMs, or DSPs. The outputs of each BUT 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. The BUTs are tested concurrently and, as a result, the length of the BIST sequence is a function of the resource under test and not a function of the size of the array or the number of resources under test. The BUTs are repeatedly reconfigured, and retested, until they have been completely tested in all of their modes of operation. Since download time is a function of the size of the array, an important goal is to reduce the number of times the BUTs must be reconfigured without sacrificing fault detection capabilities. By maintaining constant placement and routing of TPGs, ORAs, and BUTs, partial reconfiguration can be used to only write those frames of configuration memory that contain the memory bits that control the programming of the BUTs. As a result, reconfiguration time is only a function of the number of BUTs.

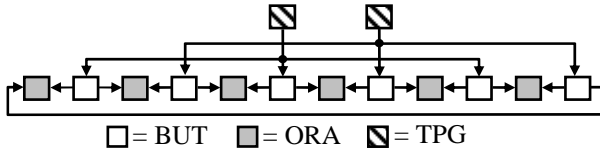


Figure 2. Basic BIST Architecture

The circular comparison based BIST architecture was originally developed for testing the 4K-bit block RAMs in Virtex-I FPGAs and the 18K-bit block RAMs in Virtex-II FPGAs [2]. When a diagnostic procedure was developed for use when testing these block RAMs, the circular comparison was found to provide significant improvements in diagnostic resolution over previous FPGA BIST architectures [2]. The diagnostic procedure is capable of not only identifying the faulty block RAM(s) but also the faulty output(s) of those RAMs as well as the faulty mode(s) of operation.

The comparison-based ORA is illustrated in Figure 3. The contents of the ORA flip-flops, which are used for the diagnostic procedure, can be obtained via partial configuration memory readback. However, we are only interested in obtaining ORA contents for diagnosis when a fault has been detected. Therefore, we exploit the dedicated carry logic to create an iterative OR chain that provides a single bit pass/fail indication at the end of the BIST sequence such that partial configuration memory readback is not required to determine pass/fail status. This helps to reduce total test time for fault-free BIST sequences. We assume that CLBs used for ORAs have been

previously tested and found to be fault-free using BIST configurations specifically for CLBs [1].

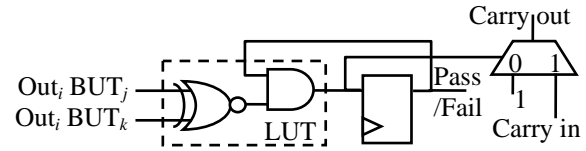


Figure 3. ORA Implementation

### 4. BIST FOR VIRTEX-4 BLOCK RAMS

Virtex-4 block RAMs are capable of storing up to 18K-bits of data. They can be configured with address and data widths given in Table 1. Only the data widths 2Kx9, 1Kx18, and 512x36 contain parity bits. The input parity bits can be used as user-supplied parity or can act as additional data, while the output parity corresponds with each byte in the output data [3].

Table 1. Block RAM Address and Data Bus Widths

Address Bus Width	Address Locations	Data Bus Width	Parity Width	Total Data Bus Width
14	16K	1	0	1
13	8K	2	0	2
12	4K	4	0	4
11	2K	8	1	9
10	1K	16	2	18
9	512	32	4	36

The block RAMs have three different write modes of operation: 1) WRITE FIRST: the input data is simultaneously written to the memory and passed to the data output, 2) READ FIRST: the data previously stored in the memory is passed to the output while the input data is written to memory, and 3) NO CHANGE: the output data is not changed during a write operation [3]. The block RAMs provide optional pipelining through the use of an output data register which can increase performance by reducing propagation delay.

The block RAMs can be configured in one of four modes of operation: 1) normal mode where the block RAM can act as a single- or dual-port RAM, 2) FIFO RAM mode, 3) ECC RAM mode, or 4) cascade RAM mode where two adjacent RAMs can be combined to provide a larger RAM. When the block RAM is configured as a dual-port RAM, each port has its own address bus, input data bus, output data bus, clock, clock enable, and write enable [3].

When the block RAM is configured as a FIFO, it may be only be configured as a 4Kx4-bit, 2Kx9-bit, 1Kx18-bit, or 512x36-bit. One port of the RAM is used to read and the other is used to write. There are status flags that indicate FULL and EMPTY conditions as well as 12-bit programmable ALMOST FULL and ALMOST EMPTY flags. It is up to the user to use these flags to stop writing or reading to avoid causing a write or read error. The FIFO can also be configured as either standard write mode or as First-Word-Fall-Through (FWFT). The latter mode allows

the first data word written to be visible immediately on the output data bus [3].

Two adjacent block RAMs can be cascaded to create a 32K×1-bit RAM (this is the only mode available for a cascaded RAM). One of the RAMs is designated as the UPPER RAM and one as the LOWER RAM. The most significant address bit is decoded to select the appropriate RAM to read or write. Two adjacent block RAMs can also be configured to function as a 512×64-bit ECC RAM [3]. Hamming code is generated for the incoming data and stored in the RAM using four parity bit locations in each RAM for a total of seven Hamming bits (for single error correction) and one overall parity bit (for double error detection). When an address location is read, the Hamming code is regenerated for the outgoing data and compared with the stored Hamming bits. Single-bit errors are corrected at the RAM output while double-bit non-correctable errors are flagged.

The BIST configurations for Virtex-4 block RAMs are summarized in Table 2 along with the test algorithms used. The March LR algorithm with background data sequences (BDS) is used to verify that the RAM core containing the memory cells is fault-free. March LR has been shown to be a superior test to March C-, while marginally increasing the test complexity [4]. A procedure is given in [4] for converting March LR to a word-oriented test by incorporating BDS. March LR without BDS is of order  $O(16N)$  where  $N$  is the number of address locations. When applying March LR to a bit-oriented memory, BDS is not needed. However, the 16K×1-bit memory configuration does not provide access to the full 18K memory cells. Therefore, the widest memory configuration (512×36-bit) provides the shortest test time when applying BDS. After March LR with BDS testing, the RAM core can be assumed to be fault-free. 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 core.

**Table 2. BIST Configurations for Block RAMs**

BIST Config	Test Algorithm	Address Locations	Data Width	Clock Cycles
1 (C)	March LR A	512	36	59,392
2 (P)	March LR B	512	36	512
3 (P)	March s2pf- & d2pf	512	36	11,776
4 (P)	MATS+	8K	2	81,920
5 (P)		16K	1	163,840
6 (P)		512	36	5,120
7 (C)	Cascade MATS+	32K	1	20
8 (P)		32K	1	20
9 (C)	ECC	512	72	5,696
10 (P)		512	72	5,696
11 (C)	FIFO	2K	9	16,834
12 (P)		512	36	4,096
13 (P)		1K	18	8,192
14 (P)		4K	4	32,768
15 (P)		4K	4	4,096
(C) Compressed (P) Partial		<b>Total BIST clock cycles = 99,532</b>		

The MATS+ test algorithm [6] is used to test the programmable address decoder. MATS+ is  $O(5N)$  and 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. Dual-port testing is achieved by using March s2pf- and March d2pf algorithms [6]. The single- and dual-port block RAM modes of operation are tested by the first six BIST configurations in Table 2.

Virtex-4 block RAMs also support cascading two adjacent block RAMs (either above or below) 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/select information as needed. This implementation does not need to be tested by a thorough March test; only a functional test is needed to completely test the additional cascading circuitry. Two BIST configurations (7 and 8 in Table 2) are needed to test each RAM in Upper and Lower modes.

Testing the Hamming code generation and bit error correction circuitry presents the interesting problem of detecting faults in a fault-tolerant circuit. However, the block RAMs can be initialized at configuration time and we can exploit this feature to initialize memory locations with test data so that the ECC circuit will correct single-bit correctable errors as well as detect and flag non-correctable double-bit errors during an initial read cycle of all memory locations in the RAM. Alternately, the ECC write feature can be disabled to allow test patterns to be written directly to the RAM core (bypassing the Hamming code generation circuitry at the RAM input) and then applied to test the Hamming code regeneration as well as the error detection and correction circuitry at the RAM output. The Hamming code generation circuitry at the input of the RAM can be tested by writing a sequence of test patterns and reading those address locations; if the Hamming code was generated correctly there will be no errors detected by the RAM output Hamming circuitry. Alternately, the ECC read feature can be disabled to allow the generated Hamming bits to be read directly from the memory locations. The test patterns needed to detect all faults in the Hamming code generation circuitry at the RAM input and Hamming code regeneration circuitry at the RAM output include all combinations of a single 1 and two 1s in a field of 0s [7]. The test patterns needed to detect all faults in the single-error correction and double-bit error detection circuits include all combinations of Hamming values with all 0s in the data field [7]. Two BIST configurations (9 and 10 in Table 2) are needed to test the ECC write and ECC read circuitry, respectively, using the alternate approaches described above.

The  $O(8N)$  March X FIFO test algorithm described in [7] begins with an initially empty FIFO which is written until full and then read until empty to test the FULL

and EMPTY flags, respectively, for the four different FIFO address/data width modes (configurations 11-14 in Table 2). The last two BIST configurations (14 and 15) in Table 2 test the ALMOST FULL and ALMOST EMPTY flags in the 4K×4-bit mode of operation, where the ALMOST flags use the full 12 bits. Between these two BIST configurations we alternate values 0xAAA and 0x555. An alternate approach is to test the ALMOST flags with a different BIST configuration for each of the 12-bits in the ALMOST FULL and ALMOST EMPTY comparators. This results in slightly high fault coverage at the expense of longer test time due to the ten additional downloads. In this case, the individual bits of the programmable ALMOST FULL flag are tested by repeatedly reconfiguring the ALMOST FULL flag value to higher values and then continuing to write more data. Similarly, the ALMOST EMPTY flag values are reconfigured while emptying of the FIFO.

The TPG design for the first six BIST configurations incorporates a finite state machine that is able to generate several RAM test algorithms including March LR with BDS, MATS+, March s2pf-, and March d2pf. Similar TPGs are designed to generate tests for cascade, ECC, and FIFO mode testing. The TPGs are implemented by synthesizing VHDL models and constraining the placement of the CLBs to the required area within the FPGA. The four TPG implementations are summarized in Table 3 in terms of the number of CLBs required.

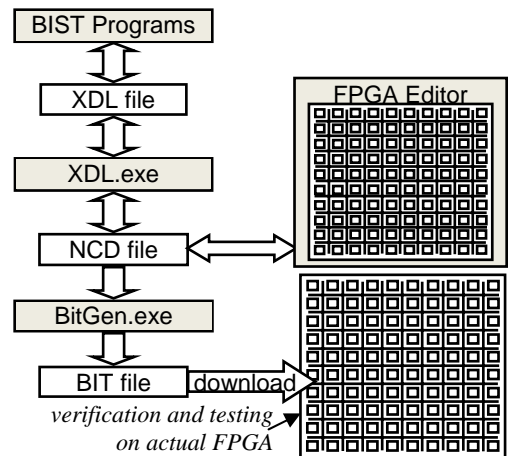
**Table 3. TPG Implementations**

TPG	# CLBs	TPG	# CLBs
BRAM	118	FIFO	23
ECC	48	CASC	3

The arrangement of ORAs in the circular comparison scheme are different for the ECC and Cascade BIST configurations when compared to the first six block RAM BIST configurations and FIFO BIST configurations. In the block RAM and FIFO BIST configurations, the ORAs compare adjacent RAMs in a column since all RAMs operate identically. In the ECC and Cascade BIST configurations, on the other hand, the ORAs compare alternating RAMs in a column since two adjacent RAMs must be combined to create the particular mode of operation. In all cases, a given ORA compares only one output, and the same output, from both RAMs being monitored, regardless of whether the RAMs are adjacent in the column (block RAM and FIFO) or separated by a row of RAMs (ECC and Cascade). During Cascade mode BIST, the ORAs which observe RAMs located at the bottom of the column and above PowerPC module include a clock enable controlled by the TPGs to disable the ORAs for those test vectors where the unconnected cascade inputs to those RAMs will cause a mismatch. This also overcomes the problem of testing cascade modes of operation with a circular comparison-based BIST approach [11].

The complete BIST architecture instantiation and connectivity is created using Xilinx Design Language

(XDL), a netlist format for describing designs at the CLB/RAM-level with complete control over placement and routing. This allows us to integrate the synthesized TPGs with the ORAs and the block RAMs whose XDL has been generated by a parameterized C program, using the design flow illustrated in Figure 4. The process consists of generating a BIST configuration template in XDL format by our C program, *V4ramBIST.exe*, converting the XDL to NCD format for routing with Xilinx place and route (PAR) software. The routed NCD file is then converted to XDL for modification by our C program, *V4ramMOD.exe*, to generate the various BIST configurations. These two C programs facilitate generation of BIST configurations for any family and size Virtex-4 FPGA. The final XDL files are converted to NCD format from which the actual download configuration bit files are generated via Xilinx *BitGen.exe*. These bit files can be generated in one of three ways: full, compressed, and partial. Full bit files contain values for every configuration bit within the device. Compressed bit files utilize the multiple frame write feature in the bitstream to reduce the size of the bitstream and in turn reduce the size of the bit file [8]. Partial bit files are created by comparing a bit file (full) to the NCD file of the subsequent BIST configuration such that only the differences between the two designs are downloaded during partial reconfiguration [9]. The difference in file sizes for each of the three types of bit files is device and design dependent. An example of this is shown in Table 4 for an LX60 FPGA where the total set of BIST configurations can be compared to a single full configuration download file of 2,163 K-bytes.



**Figure 4. BIST Configuration Generation Process**

**Table 4. LX60 Bit File Size Comparison**

BIST Config	File Size K-bytes	BIST Config	File Size K-bytes	BIST Config	File Size K-bytes
1 (C)	1,154	6 (P)	67	11 (C)	1,061
2 (P)	55	7 (C)	951	12 (P)	16
3 (P)	55	8 (P)	55	13 (P)	4
4 (P)	55	9 (C)	1,146	14 (P)	4
5 (P)	67	10 (P)	67	15 (P)	4
<b>Total File Size = 3,615 K-bytes</b>					

Figure 5 shows that actual implementation of BIST configurations 1-6 (Table 2) in a Virtex 4 LX15. The two TPGs are located in the center CLBs. One TPG and its routing is highlighted in red and the other in green. Each TPG drives alternating rows of block RAMs as can be seen in the figure. The RAM to ORA connections are highlighted in blue where the ORAs are located in the CLB columns adjacent to the block RAM columns.

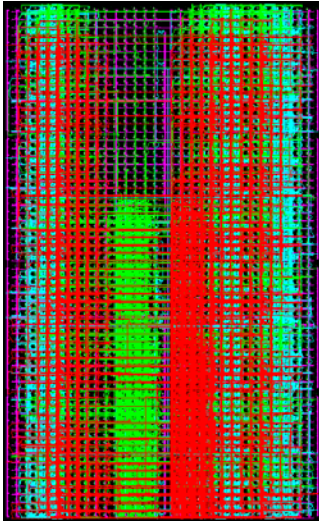


Figure 5. Block BIST Implementation in LX15

## 5. EXPERIMENTAL RESULTS

We performed fault injection on an FX12 device, which involved 456 configuration memory bit faults. For each BIST type, a single fault in the configuration memory was emulated by overwriting the desired configuration memory bit to the desired stuck-at value after each BIST configuration was downloaded but before the BIST sequence was executed. This process was repeated for each of the 15 BIST configurations.

Figure 6 shows an example of the individual and cumulative fault detection for the first six BIST configurations in Table 2 (note that March s2pf- and d2pf algorithms are shown separately in the figure). Note that the number of faults detected by each configuration is relatively similar. However, cumulative fault coverage for the block RAM was approximately 84% using only the first six block RAM BIST configurations. Figure 7 illustrates the overall block RAM fault detection achieved as a function of each set of BIST configurations where the ECC and cascade BIST configurations have been combined. Of the 456 fault emulated, only six faults were not detected (for total fault coverage of 98.7%) and we are in the process of investigating these six faults to determine the conditions necessary for their detection if, in fact, they are indeed detectable.

We executed the block BIST configuration on a number of devices including two FX12s, ten SX35s, and eighteen LX60s, where half of the SX35s and LX60s

were Xilinx EasyPath parts with manufacturing defects [10]. When testing one LX60 EasyPath part, the first BIST configuration (see Table 2) detected faults. Using the diagnostic algorithm described in [2], we were able to determine that a fault was present in one block RAM (located at array coordinates  $X=10$   $Y=36$ ) on bit 23 and was observed on both A and B output ports. The Port A fault detected in every time the BIST configuration was executed while the Port B fault was detected intermittently, usually when the device was initially being tested (in other words, when the device is powered on).

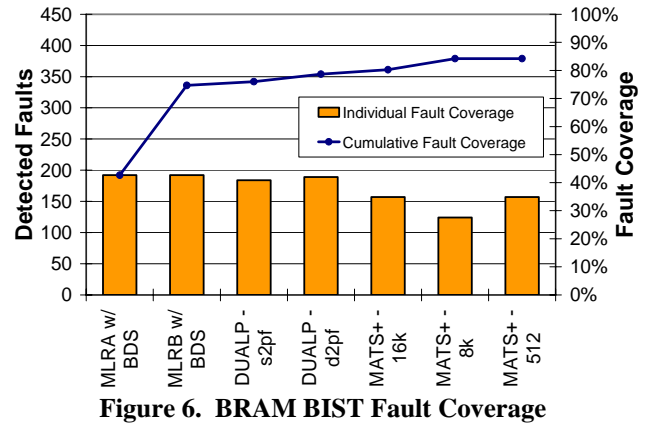


Figure 6. BRAM BIST Fault Coverage

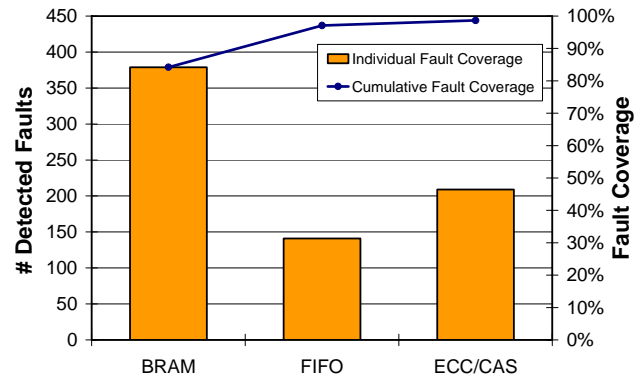


Figure 7. Overall BIST Fault Coverage

Using Xilinx timing analysis tool, *trce.exe*, we determined the maximum BIST clock frequency for every BIST configuration for every Virtex-4 device. The results of this timing analysis are summarized in Figure 8 for an LX60. It can clearly be seen that the MATS+ 512 BIST configuration and the 2K $\times$ 9-bit FIFO BIST configuration, result in the lowest BIST clock frequency for their corresponding set of BIST configurations. This is because these particular BIST configurations test the block RAM clock inversion which results in opposite edge clocking between the TPG, block RAM, and ORAs. Next, we generated those BIST configurations for every Virtex-4 device. Figure 9 shows this worst-case timing analysis for each of the four BIST sets of BIST configurations for each device along with the number of block RAMs located in the device (where the number is located above the columns).

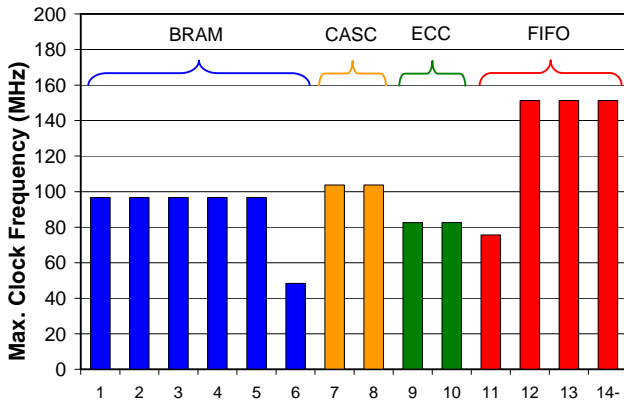


Figure 8. Max. BIST Clock Frequency for LX60

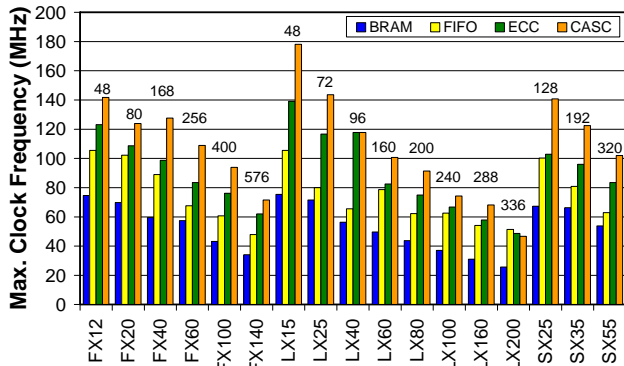


Figure 9. Max. BIST Clock Frequency for all Devices

Our goal was to achieve a maximum BIST clock frequency of 50 MHz or greater to correspond with the maximum Test Clock (TCK) frequency for the boundary scan interface [3]. However, as can be seen from the data in Figure 9, the goal was not achieved for BIST configuration number 6 (the MATs+ 512 BIST configuration) for LX80, LX100, LX160, LX200, FX100, and FX140 devices. To increase the maximum BIST clock frequency, the BIST configurations are generated for only half of the device at a time, either the top half or the bottom half of the array. This reduces not only the number of RAMs under test in a given BIST configuration, but also reduces the routing since the RAMs under test are localized to half of the array. This results in an improvement in the maximum BIST clock frequency, as shown in Figure 10, at the expense of doubling the number of BIST configurations.

Alternately, we could increase the number of TPGs to reduce the loading. However, we were advised by Xilinx engineers to avoid testing all RAMs simultaneously in the larger devices. Therefore, doubling the number of block RAM BIST configurations in the larger devices is our current methodology. In addition, to testing only half of the array at a time, we also relocate the TPG from the middle of the array to the bottom of the array when testing block RAMs in the lower half of the array. The timing analysis results in Figure 10 reflect this relocation of the TPG where it can be seen that the 50 MHz goal is met with the exception of the two largest LX devices.

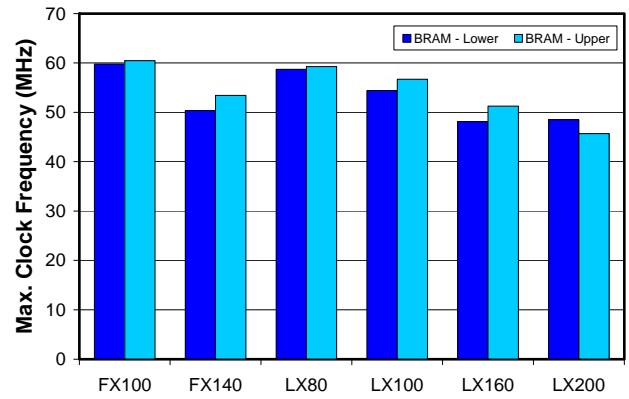


Figure 10. Max. BIST Clock Freq. for Sub-Arrays

## 6. SUMMARY

We have presented a BIST approach for the programmable block RAMs in Virtex-4 FPGAs. The BIST approach is capable of detecting faults in all modes of operation and the supporting diagnostic algorithm presented in [2] is capable of identifying the faulty RAM(s), the faulty mode(s) of operation, and the faulty output(s) based solely on the failing BIST configuration results. The BIST configurations can be applied in-system during off-line operation for high-reliability/availability systems and/or fault-tolerant applications.

## 7. 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 (v2.1), Xilinx, Inc., 2007 (available at www.xilinx.com)
- [4] 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
- [5] 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
- [6] S. Hamdioui, *Testing Static Random Access Memories*, Springer, 2004
- [7] L-T Wang, C. Stroud and N. Touba, *System-on-Chip Test Architectures*, Elsevier, 2007
- [8] \_\_, "Development System Reference Guide", Xilinx, Inc., 2005 (available at www.xilinx.com)
- [9] \_\_, "Virtex-4 Configuration Guide," UG071 (v1.5), Xilinx, Inc., 2007 (available at www.xilinx.com)
- [10] F. Toth, "The Easy Path to Cost Reduction", *Xcell J.* vol. 48, pp. 96-98, 2004
- [11] A. Sarvi and J. Fan, "Automated BIST-based diagnostic solution for SOPC," *Proc. International Conf. on Design and Test of Integrated Systems in Nanoscale Technology*, pp. 263-267, 2006