

Built-In Self-Test of Embedded Memory Cores in Virtex-5 Field Programmable Gate Arrays

Justin L. Dailey, Brooks R. Garrison, Mary D. Pulukuri and Charles E. Stroud

Department of Electrical and Computer Engineering

Auburn University

Auburn, AL 36849-5201, USA

dailejl@auburn.edu bgarrisn@gmail.com marypulukuri@ayhoo.com strouce@auburn.edu

Abstract—This paper introduces and details a Built-In Self-Test (BIST) approach designed for the embedded Block Random Access Memories (BRAMs) found within Xilinx Virtex-5 Field Programmable Gate Arrays (FPGAs). The BIST is designed to test the BRAMs in all configurable modes of operation including single-port, dual-port, first-in first-out (FIFO), first-in first-out with error correcting code (FIFOECC), and error correcting code (ECC) modes. The BIST architecture and implementation in actual Virtex-5 FPGAs will be discussed along with fault detection and timing analysis data taken from those implementations.

I. INTRODUCTION AND BACKGROUND

As complexity of Very Large Scale Integration (VLSI) devices including Field Programmable Gate Arrays (FPGAs) continues to increase, the need for an efficient and economical testing method for testing becomes critical for high reliability systems [1]. The ability to reprogram an FPGA in-system to test and diagnose faults becomes extremely valuable to fault tolerance since the FPGA can be configured to avoid faults. However, the FPGA must be reconfigured multiple times in order to achieve sufficient detection and identification of faulty resources. The main hindrance to FPGA Built-In Self-Test (BIST) has been the large number of configurations needed to achieve fault detection and diagnosis. Thus the download time for the testing configurations accounts for a large majority of the total test time [2]. Furthermore, the increasing size and complexity of FPGAs along with the addition of multiple specialized cores within the device such as random access memories (RAMs) and digital signal processors (DSPs) continues increase the testing challenge [3].

In this paper, we discuss the full implementation and analysis of a BIST for the block RAMs in Xilinx Virtex-5 series FPGAs. The BIST design takes advantage of the architectural and operational features available in the FPGA to exploit dynamic partial reconfiguration for improvements in download times as well as partial configuration memory read back to facilitate diagnosis of faulty resources. Section II presents a brief overview of the architecture of Virtex-5 FPGAs. This is followed in Section III by details of our general BIST architecture as well as related prior FPGA BIST work. Section IV describes the details of our BIST approach for Virtex-5 Block RAMs and the configurations needed to test the different operational modes. Experimental

results including analysis of the fault detection and timing capabilities of the BIST implementation in actual Virtex-5 FPGAs are presented in Section V before the paper concludes in Section VI

II. OVERVIEW OF VIRTEX-5 ARCHITECTURE

The general architecture of Xilinx Virtex-5 FPGAs consists of configurable logic blocks (CLBs), programmable input/output blocks (IOBs), block RAMs, and DSPs [4]. These elements are arranged in columns and rows within the FPGA as illustrated in Figure 1. In contrast with the figure, Virtex-5 devices contain more rows than columns. The number of rows and columns dedicated to each of the components varies from device to device in each family (LX, LXT, SXT, or FXT).

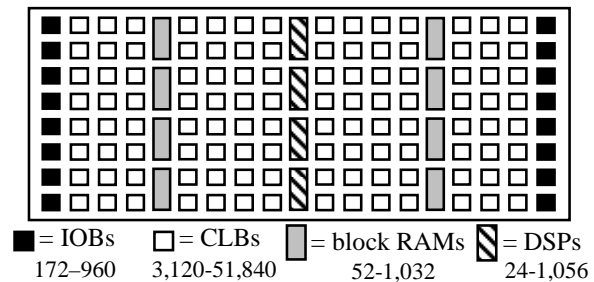


Figure 1. General Virtex-5 Architecture

Virtex-5 FPGAs support frame addressable read and write operations [5]. This feature facilitates dynamic partial reconfiguration of the FPGA with frame addressable write operations as well as partial configuration memory read back via frame addressable read operations. Another important feature in Virtex-5 devices is the ability to perform multiple frame write operation where the configuration data contained in the Frame Data Register is written to multiple frames on the FPGA. The multiple frame write can provide a significant reduction in the total reconfiguration time of the device depending on the size of the device and the regularity of the design structure. In order to fully exploit this reduction, our BIST architecture has been designed to be extremely regular across the array and aligned with the frame structure of the configuration memory. During configuration memory read back the contents of the memory elements, which include CLB flip-flops/latches and block RAMs are read along 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. The flip-flops within the CLBs of a column are located in the same frame such that their contents can be observed by a simply reading a single frame. Our BIST approach takes advantage of these Virtex-5 architectural and operational features in order to provide efficient implementation and operation.

III. OVERVIEW OF BIST ARCHITECTURE

The basic architecture of our BIST implementation is illustrated in Figure 2. This architecture has been used previously in order to test elements in Virtex-4 FPGAs including CLBs, IOBs, block RAMs, and DSPs. A pair of test pattern generators (TPGs) are used in order to supply identical test patterns to alternating identically configured blocks under test (BUTs) which may be CLBs, IOBs, DSPs, or, in our case, block RAMs. The outputs of each of the BUTs are monitored by two separate output response analyzers (ORAs). Each of these ORAs compares the outputs of two BUTs. This results in a circular comparison architecture seen in Figure 2. All BUTs are tested concurrently such that the length of the BIST sequence is a function of the resource being tested and not the number of resources being tested. BUTs are repeatedly reconfigured and retested during the test session until they have been completely tested in all modes of operation. Since download time dominates the total testing time, it is important to minimize the number of configurations needed to perform a complete test session. By maintaining consistent placement and routing of TPGs, ORAs, and BUTs for the various test configurations, we are able to take advantage of partial reconfiguration in order to change the configuration of the BUTs without affecting the other resources. Thus, reconfiguration time is only a function of the number of BUTs.

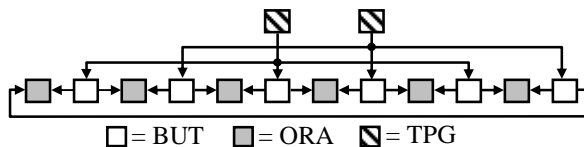


Figure 2. Basic BIST Architecture

The circular comparison based BIST architecture was originally developed for testing block RAMs in Virtex-I and Virtex-II FPGAs [3]. The circular comparison implementation provides significant diagnostic improvements in diagnostic resolution over previous FPGA BIST architectures [3]. The diagnostic procedure is capable of identifying the faulty block RAM(s), the faulty output(s) of those RAMs, and also the faulty mode(s) of operation.

The comparison-based ORA is illustrated in Figure 3. Two outputs are compared in each ORA via the two exclusive-NOR gates. This double comparison slightly reduces the diagnostic resolution but more importantly reduces the resources required for implementing ORAs by half. The contents of the ORA flip-flops can be obtained via partial configuration memory read back for use in diagnosis. However, we only become 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 providing a single bit pass/fail indication at the end of the BIST sequence resulting in significant test time reduction. Thus, partial configuration memory read back is only needed for diagnosis and not to determine pass/fail status. We assume CLBs used for ORAs have been previously verified as fault-free using BIST configurations for CLBs [2].

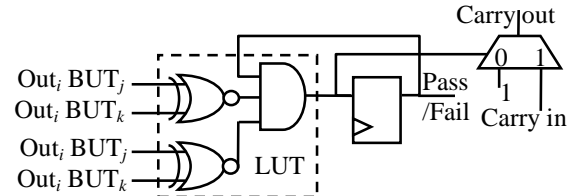


Figure 3. ORA Implementation

IV. BIST FOR VIRTEX-5 BLOCK RAMS

Virtex-5 block RAMs are capable of storing up to 36K-bits of data. They can be configured with address and data widths given in Table 1. Only the data widths 4K×8, 2K×16, 1K×32, and 512×64 contain 1, 2, 4, and 8 parity bits, respectively. 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 [4]. It should be noted that these parity bits contain Hamming code in error correction code (ECC) modes of operation.

Table 1. Block RAM Address and Data Bus Widths

Address Bus Width	Address Space	Data Bus Width	Parity Width	Total Data Bus Width
15	32K	1	0	1
14	16K	2	0	2
13	8K	4	0	4
12	4K	8	1	9
11	2K	16	2	18
10	1K	32	4	36
9	512	64	8	72

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 [4]. Block RAMs provide optional pipelining through a programmable output data register which can increase performance by reducing propagation delay. Block RAMs may be configured in one of five different modes of operation: 1) normal mode where the block RAM may act as a single- or dual-port RAM (denoted BRAM in this paper), 2) first-in first-out (FIFO) mode, 3) FIFO ECC mode, 4) ECC RAM mode, or 5) cascade mode where two adjacent RAMs can be combined to operate as a single 64K×1-bit RAM. When 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 [4].

When the block RAM is configured as a FIFO, it may only be configured as 8K×4-bit, 4K×9-bit, 2K×18-bit, or 1K×36-bit. Additionally, the FIFO may be used as in 512k×72-bit ECC mode (denoted FIFOECC in this paper). One port of the RAM is used for read operations, while the other is used for write operations. Two status flags indicate a FULL or EMPTY condition as well as 13-bit hex programmable ALMOST FULL and ALMOST EMPTY flags. It is the responsibility of the user to recognize these flags to avoid errors. Additionally, the FIFO may be configured as either standard write mode or as First-Word-Fall-Through (FWFT) which allows the first data word written into the FIFO to become immediately visible on the output data bus [4].

Two adjacent block RAMs within a column may be cascaded together to create a single 64K×1-bit RAM. In this case one RAM is configured as the UPPER RAM and the other will be configured as the LOWER RAM. The most significant address bit in the address bus will then be used to decode which RAM the data will be read from. In this mode only the UPPER RAM will display data on its output. If data is read from the LOWER RAM it will be passed through the dedicated cascade routing and displayed on the output of the UPPER RAM.

The BIST configurations we developed for Virtex-5 block RAMs are summarized in Table 2 along with the RAM 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 only marginally increasing the test complexity [6]. A procedure is given in [6] 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 32K×1-bit memory mode does not provide access to the full 36K RAM core. Therefore, the widest memory configuration (512×72-bit) provides full access to the RAM core as well as 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 BIST configuration #8 since once tested there is no need to repeat pattern sensitivity and coupling fault tests in the RAM core.

The MATS+ test algorithm [7] 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 d2pf algorithms [7]. The single- and dual-port block RAM modes of operation are tested by the first two BIST configurations in Table 2.

Virtex-5 block RAMs also support cascading any two adjacent block RAMs (either above or below) to create a 64K×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 32K×1-bit RAMs and manipulating multiplexers to direct/select infor-

mation 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 (18 and 19 in Table 2) are needed to test each RAM in UPPER and LOWER modes.

Table 2. BIST Configurations for Block RAMs

BIST Config	RAM Mode	Test Algorithm	Address Space	Data Width	Clock Cycles
1 (C)	BRAM	March s2pf-	1K	36	20,000
2 (P)		March d2pf	1K	36	15,000
3 (P)		MATS+	2K	18	25,000
4 (P)			4K	9	45,000
5 (P)			8K	4	85,000
6 (P)			16K	2	165,000
7 (P)			32K	1	330,000
8 (C)	ECC	March LR w/BDS	512	72	23,000
9 (P)		ECC Read	512	72	7,000
10 (P)		ECC Write	512	72	7,000
11 (C)	FIFO	FIFO March X	1K	36	8,500
12 (P)			2K	18	34,000
13 (P)			4K	9	66,000
14 (P)			8K	4	131,500
15 (P)			8K	4	131,500
16 (C)	FIFOECC	ECC Read	512	72	10,000
17 (P)		ECC Write	512	72	10,000
18 (C)	CASC	March Y	1K	64	36
19 (P)			1K	64	36
Total BIST clock cycles = 1,113,572					
(C) = Compressed Configuration, (P) = Partial Reconfiguration					

Testing the Hamming code generation and bit error detection and correction circuitry poses an 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. However, 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 [8]. 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 [8]. Two BIST configurations (#9 and #10 in Table 2) are needed to test error detection and correction circuitry and the Hamming code generation circuitry, respectively, using the approaches described above.

The FIFO March X test algorithm described in [8] is $O(8N)$. It begins with an empty FIFO which is written until full and then read until empty to test the FULL and EMPTY flags for the different FIFO address/data width modes (configurations 11-15 in Table 2). Configurations 14-15 test the ALMOST FULL and ALMOST EMPTY flags in the $8K \times 4$ -bit mode of operation, where the ALMOST flags use the full 13 bits of address space. Between these two BIST configurations we alternate values 0xAAAA and 0x5555. An alternate approach is to test the two ALMOST flags with a different BIST configuration for each of the 12-bits in the ALMOST FULL and ALMOST EMPTY comparators. This will result in slightly high fault coverage at the expense of longer test time due to the 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. A modified version of FIFO March X algorithm was developed to include the patterns described previously to test the Hamming ECC circuitry and is used to test the FIFOECC mode in configurations 16 and 17.

The TPG design for the BRAM 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 were designed to generate the test patterns for cascade (CASC), ECC, FIFO, and FIFOECC modes. The TPGs are implemented by synthesizing VHDL models and constraining the placement of the CLBs to the desired area within the FPGA. The five TPG implementations are summarized in Table 3 in terms of the number of CLBs required.

Table 3. TPG Implementations

TPG	# Slices	TPG	# Slices	TPG	# Slices
BRAM	148	FIFO	34	CASC	4
ECC	205	FIFOECC	43		

The ORA routing is the same for all BIST configurations except CASC. In the other configurations the ORAs compare adjacent RAMs in the column since all RAMs operate identically. In the CASC configurations every other RAM is instead routed to the ORA for comparison as illustrated in Figure 4. This is done because the RAMs are configured in alternating order as either a LOWER or UPPER RAM. Therefore, each UPPER RAM will be compared to another UPPER RAM and each LOWER RAM will be compared to another LOWER RAM. However, the RAMs which do not have a RAM located directly below them (located on the bottom of a column or above a PowerPC module, for example) present a special case. Since data being read will only be seen on the output ports of CASC RAMs configured as UPPER

RAMs, we are able to leave the outputs of the LOWER RAMs in this situation un-routed [4]. This allows us to greatly simplify our CASC TPG from the ones described in [9] and [12] since we do not have to account for any expected mismatches. Using this method we maintain our circular comparison and overcome the problem of testing cascade modes of operation with a circular comparison-based BIST approach as described in [10].

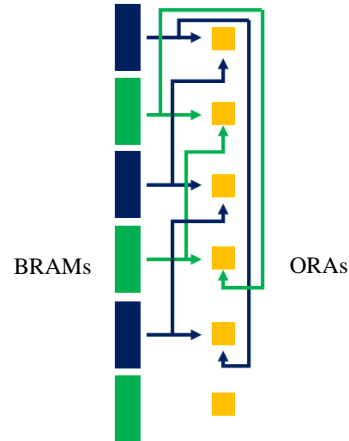


Figure 4. Cascade ORA Routing

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 5. The process consists of generating five BIST configuration templates (one for each RAM mode in Table 2) in XDL format by our C program, *V5RAMBIST.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, *V5RAMMOD.exe*, to generate the 19 different BIST configurations. These two C programs facilitate generation of BIST configurations for any family and size Virtex-5 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 configuration bit files contain values for every configuration bit within the device. Compressed bit files utilize the multiple frame write feature in the FPGA to reduce the size of the configuration file [5]. Partial reconfiguration files are created by comparing the NCD files of the reference and subsequent BIST configurations such that only the differences between the two designs are downloaded during partial reconfiguration [11]. The difference in file sizes for each of the three types of configuration is device and design dependent. An example of this is shown in Table 4 for an LX30 FPGA where the total set of BIST configurations can be compared to a single full configuration download file of 2,163 K-bytes.

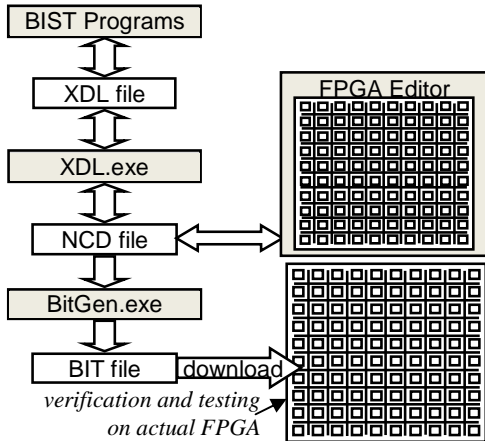


Figure 5. BIST Configuration Generation Process

Table 4. LX30 BIST Configuration File Sizes

BIST Config	File Size K-bytes	BIST Config	File Size K-bytes	BIST Config	File Size K-bytes
1 (C)	583	7 (P)	49	13 (P)	4
2 (P)	55	8 (C)	592	14 (P)	4
3 (P)	49	9 (P)	3	15 (P)	4
4 (P)	49	10 (P)	50	16 (C)	564
5 (P)	49	11 (C)	532	17 (P)	4
6 (P)	49	12 (P)	4	18 (C)	387
Total File Size = 3,034 K-bytes				19 (P)	3

Figure 6 shows that actual implementation of the BIST BRAM configurations in a Virtex-5 LX30. The two TPGs are located in the CLBs indicated in the figure. One TPG and its associated routing is highlighted in red and the other TPG 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.

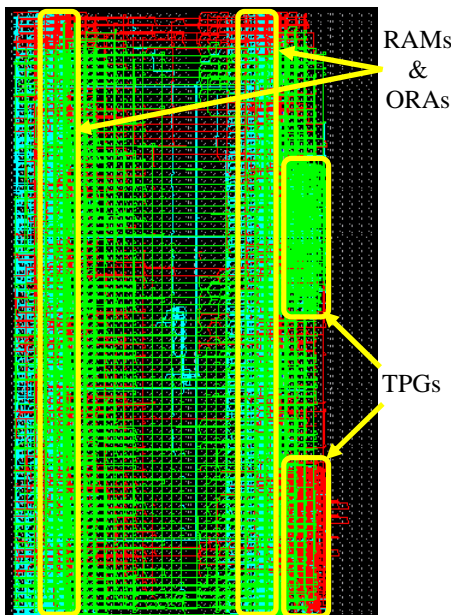


Figure 6. BRAM BIST Implementation in an LX30

V. EXPERIMENTAL RESULTS

All 19 BIST configurations were generated for every Virtex-5 FPGA. The BIST configurations for LX30T, LX50T, SX35T, SX50T, FX30T, and FX70T FPGAs were downloaded and verified on the actual devices. Fault detection capabilities were verified with physical fault injection using the bits in the configuration memory. The physical fault injection was applied to determine the fault coverage for the configuration memory bit faults that control the block RAMs. There are a total of 488 possible configuration memory bit faults associated with each block RAM. Each fault was emulated by overwriting the desired configuration bit with the stuck-at value of the desired fault before performing the BIST sequence. This process is repeated for each of the 19 BIST configurations and each of the 488 configuration memory bit faults.

The individual and cumulative detection and fault coverage for the seven BRAM BIST configurations from Table 2 is shown in Figure 7. Each BIST configuration detects between 100 and 200 faults resulting in cumulative fault coverage of 84% from this set of seven BIST configurations alone. Figure 8 shows the overall fault detection and fault coverage for the entire set of 19 BIST configurations in Table 2. Where it can be seen that we achieve 100% fault coverage of detectable faults.

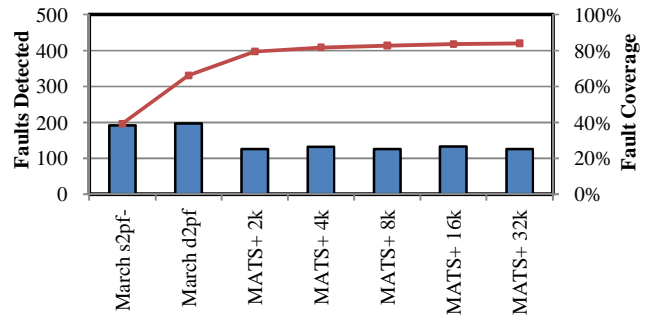


Figure 7. BRAM BIST Fault Detection

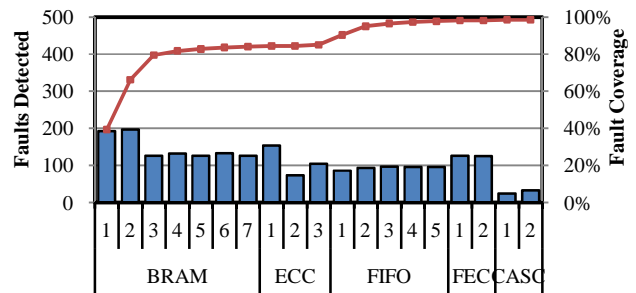


Figure 8. Overall BIST Fault Detection

Using the Xilinx timing analysis tool, *trce.exe*, we determined the maximum BIST clock frequency for each of the BIST configurations for all Virtex-5 devices. Figure 9 shows the results from this analysis obtained for the LX30T device. For this device the FIFOECC and the CASC test configurations are able to run at the fastest speed. It can also be seen that for the five different configurations the clocking

frequency for the phases remains nearly the same except in the cases of the ECC BIST configuration #3 and FIFO BIST configuration #1. This is due to these two phases testing the clock inversion of the RAMs. When the RAM clock is inverted it results in opposite edge clocking effectively halving the maximum BIST clock frequency. This problem is overcome by inverting the TPG and ORA clocks in their CLBs during these two BIST configurations. These BIST configurations with inverted TPG and ORA clocks are positioned at either the beginning or end of a set of BIST configurations so that the inversion of the TPG and ORA clocks is performed only once in the set of BIST configurations to minimize download time and test time.

Figure 10 displays the results from the maximum BIST clock frequency timing analysis on various Virtex-5 devices. For each of the five sets of BIST configurations the configuration with the lowest maximum BIST clock frequency is shown in the figure. As can be seen by comparing the LX30T clock frequency data in Figure 9 and Figure 10, the maximum BIST clock frequency for the ECC and FIFO BIST configurations increases from just over 40 MHz in Figure 9 to well over 80 MHz in Figure 10 by inverting the TPG and ORA clocks in the ECC and FIFO BIST configurations. As a result, the maximum BIST clock frequency for ECC and FIFO BIST configurations are similar to that of the BRAM BIST configurations.

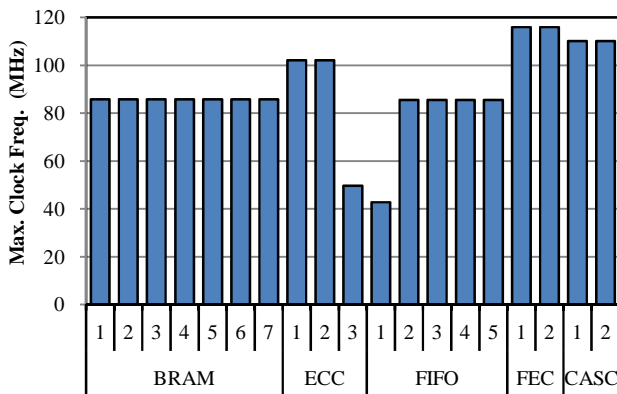


Figure 9. Maximum BIST Clock Frequency for LX30T

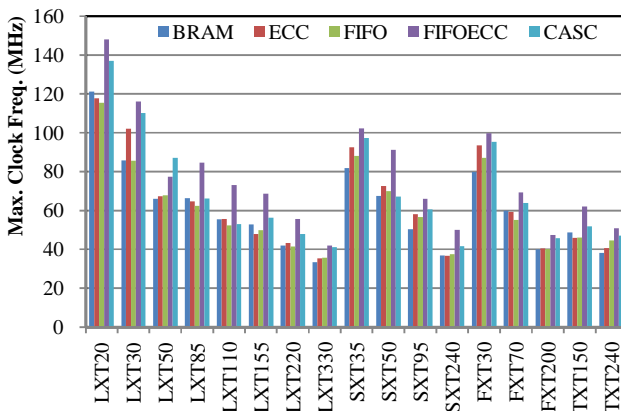


Figure 10. Maximum BIST Clock Frequency

VI. SUMMARY

In this paper we have presented a BIST approach for the embedded block RAMs in Xilinx Virtex-5 FPGAs. The BIST configurations test the RAMs in all of their modes of operation in order to achieve high fault coverage. By taking advantage of the diagnostic algorithm presented in [3] we are able to identify any faulty RAM(s), faulty mode(s) of operation, and even specific faulty output(s). These BIST configurations can be downloaded and executed in-system during off-line operation and are applicable for high reliability/availability systems as well as fault-tolerant applications.

REFERENCES

- [1] C. Stroud, *A Designer's Guide to Built-In Self-Test.*: Springer, 2002.
- [2] M. Abromovici 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.
- [3] C. Stroud and S. Garimella, "Built-In Self-Test and Diagnosis of Multiple Embedded Cores in SoCs," *Proc. International Conference on Embedded Systems and Applications*, pp. 130-136, 2005.
- [4] "Virtex-5 FPGA User Guide," Xilinx Inc., 2010.¹
- [5] "Xilinx Development System Reference Guide," Xilinx Inc., 2008.¹
- [6] A. van de Goor, I. Tlili and S. Hamdioui, "March LR: A Test for Realistic Linked Faults," *Proc. IEEE VLSI Test Symp.*, pp. 272-280, 1996.
- [7] S. Hamdioui, *Testing Static Random Access Memories.*: Kluwer Academic Publishers, 2004.
- [8] L-T. Wang, C. Stroud and N. Touba, *System-On-Chip Test Architectures.*: Morgan Kaufmann, 2008.
- [9] B. Garrison, "Analysis and Improvement of Virtex-4 Block RAM Built-In Self-Test and Introduction to Virtex-5 Block RAM Built-In Self-Test," Auburn University MS Thesis, 2009.
- [10] 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.
- [11] "Virtex-5 FPGA Configuration User Guide," Xilinx Inc., 2010.¹
- [12] B. Garrison, D. Milton and C. Stroud, "Built-In Self-Test for Memory Resources in Virtex-4 Field Programmable Gate Arrays," *Proc. International Conf. on Computers and Their Applications*, pp. 63-68, 2009.

¹ Available at www.xilinx.com