Diagnosis of Logic and Memory Resources in FPGAs Using Embedded Processor Based Built-In Self- Test

C.Arunabala

(Deartment of ECE,Samskruti college of engineering and technology-Ghatakesar-Hyderabad)
hodece@samskruti.ac.in,arunabala4700@gmail.com

Abstract : *We present an embedded processor based approach for Built-In Self-Test (BIST) and diagnosis of pro-grammable logic and memory resources in Field Programmable Gate Arrays (FPGAs). The resources under test include the programmable logic blocks (PLBs), large random access memories (RAMs), and digital signal proces-sors (DSPs) in all of their modes of operation. The approach is applicable to any FPGA that supports dynamic par-tial reconfiguration via an embedded processor core. As a case study, we present the application to test and diag-nose logic and memory resources in the Xilinx Virtex-4 series FPGA. We also exploit architectural features to en-hance dynamic partial reconfiguration for BIST to reduce the amount of program memory storage for the embedded processor core as well as the total time required for BIST.*[1]

Keywords – BIST,FPGA,PLB,DSP

I.   INTRODUCTION AND BACKGROUND

Built-In Self-Test (BIST) of programmable logic and routing resources in Field Programmable Gate Arrays (FPGAs) has been a topic for research and development for the past decade [1][2]. 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 the practical application of this idea has been the large number of BIST and diagnostic configurations required to achieve detec-tion and identification of faulty resources. This barrier is compounded by the time required to download these con-figurations into the FPGA since the download time represents the major portion of the total testing time [1]. The problem is further confounded by increases in size and complexity of FPGAs with the addition of specialized cores, including large random access memories (RAMs) and digital signal processors (DSPs). However, the advent of em-bedded processor cores in FPGAs with the ability to perform on-chip dynamic partial reconfiguration of the FPGA via write/read access to the FPGA configuration memory provides a mechanism for efficient and practical imple-mentation of on-chip BIST and diagnosis of FPGA resources for fault-tolerant applications.In this paper, we present a case study of BIST and diagnosis of the Xilinx Virtex-4 series FPGA. The BIST ap-proach exploits the use of embedded processors, both hard and soft cores, for dynamic partial reconfiguration for BIST and partial configuration memory readback for BIST results retrieval and for on-chip diagnosis of faulty re-sources. We begin with an overview of the architectural and operational features of the Virtex-4 series FPGAs in Section 2. The BIST architecture, which is capable of providing high diagnostic resolution under the control of an embedded processor core, is presented in Section 3. The BIST configurations for programmable logic blocks (PLBs) as well as for RAMs and DSPs are discussed in Sections 4 and 5, respectively. The diagnostic procedure is described in Section 6. Experimental results are presented in Section 7 and the paper is summarized in Section 8.

II.   OVERVIEW OF VIRTEX-4 ARCHITECTURAL AND OPERATIONAL FEATURES

The Virtex-4 FPGA is easily the most complex FPGA yet offered by Xilinx. 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 18Kbit 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 18×18-bit signed multiplier and a 48-bit

adder/subtractor with registers for ac-cumulator operation. The numbers of columns of PLBs, block RAMs, and DSPs varies, as does the number of rows, with the size and family (LX, SX, or FX) of Virtex-4 FPGA. The ranges for the total number of PLBs, block RAMs, and DSPs is given in the legend in Figure 1
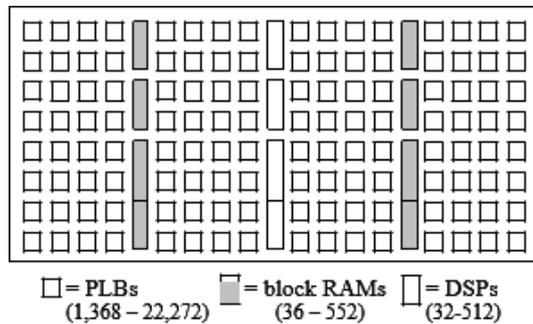

□ = PLBs   ▨ = block RAMs   ▯ = DSPs
(1,368 – 22,272)   (36 – 552)   (32-512)

Figure 1. Basic Virtex 4 Architecture

□ = BUT   ▨ = ORA ⊞ TPG
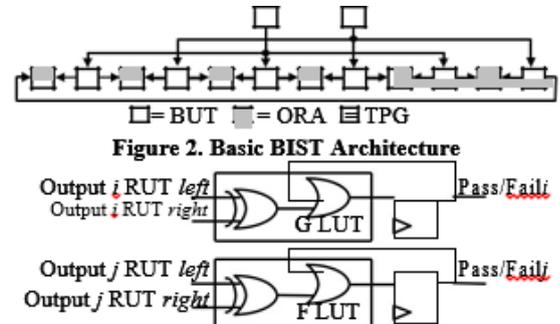
Figure 2. Basic BIST Architecture

Figure 3. Two ORAs Implemented in One Slice

The Virtex- 4 FX family includes one to two PowerPC 405 embedded RISC processor cores and all Virtex-4 FPGAs support the synthesis of soft embedded processor cores such as the 32-bit Microblaze and the 8-bit Picoblaze proces-sors [3]. Furthermore, Virtex -4 provides the ability to perform dynamic partial reconfiguration of the FPGA con-figuration memory. This can be done internally by any embedded processor core via an internal configuration ac-cess port (ICAP) in the FPGA. In most cases an embedded processor can reconfigure the FPGA much faster than external download. For example, in Virtex-4, the PowerPC processor core is advertised to operate at 450 MHz while the boundary scan interface, typically used for in-system test applications, can only operate at 50 MHz. As a result, a nine times speed-up in test time is possible by using an embedded processor core for BIST control and exe-cution, retrieval and diagnosis of BIST results, and algorithmic dynamic partial reconfiguration for subsequent BIST configurations. It should be noted that support of embedded processor cores with access to the configuration mem-ory for on-chip dynamic partial reconfiguration is also supported by other FPGAs. As a result, while the focus of this paper is the Virtex-4, the BIST and diagnostic approaches can be applied to many other FPGAs as well.

Virtex-4 supports frame addressable write and read access of the configuration memory which facilitates both dy-namic partial reconfiguration and partial configuration memory readback. An important feature in the Virtex-4 is the ability to write multiple frame addresses with the same configuration data which 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 regu-lar-structured designs (like the BIST approach) that repeat across the array. During configuration memory readback, the contents of memory elements, flip-flops and 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 each column of 16 PLBs . 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.

III. THE BEST ARCHITECTURE AND OPERATION

The basic BIST architecture is illustrated in Figure 2 and is used for testing all programmable logic and memory re-sources in the Virtex-4 FPGA including PLBs, block RAMs, and DSPs. Multiple identically configured test pattern generators (TPGs) supply test patterns to identically configured resources under test (RUTs). The outputs of the RUTs are monitored by two output response analyzers (ORAs) and compared with the outputs of two different RUTs. This results in a circular comparison that provides improved diagnostic resolution over previous methods of BIST for FPGAs [4]. 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 as illustrated in Figure 3 where each ORA monitors one output from the RUT to the left and the other to the right of the ORA. At the end of the BIST sequence, or at the end of a series of BIST sequences, the ORA contents are

retrieved via partial configuration memory readback. The TPGs can be implemented in the embedded processor core or by using a variety of programmable resources based on the resources being tested, as will be discussed in the subsequent sections. Multiple TPGs alleviates the problem of faults in a TPG and/or routing resources for TPG-to-RUT connections from escaping detection [1].

The RUTs are repeatedly reconfigured in all of their modes of operation and this set of BIST configurations is re-ferred to as a test session. Since the embedded processor core is used for dynamic partial reconfiguration for BIST and since the processor core requires programmable logic and routing resources for the program memory, only half of the FPGA can be tested at any given time. When one half of the FPGA has been tested, the positions of the proc-essor core and the resources under test are swapped, as illustrated in Figure 4. The process of performing BIST con-sists of the following steps:

Step 1. Download the processor core and interface circuitry configuration along with the initial BIST architecture for a given resource under test.

Step 2. The processor core executes the BIST sequence, retrieves BIST results via partial configuration memory readback, and performs diagnosis if faults are detected.

Step 3. The processor reconfigures the RUTs for the next BIST configuration for the given resources under test.

Step 4. Repeat Steps 2 and 3 until all BIST configurations have been applied to the RUTs.

Step 5. Repeat Steps 1 through 4 for all resources to be tested.

Step 6. Swap positions of processor core and BIST architecture and repeat Steps 1 through 5.

It should be noted that the BIST results from previous execu-tions of the BIST sequence will not be lost or corrupted dur-ing partial dynamic reconfiguration. Therefore, the retrieval of BIST results in Step 2 can be deferred until the completion of Step 4 to reduce overall test time. In this case, there is some loss of diagnostic resolution in that, while the faulty RUT can be still be identified, the faulty mode of operation cannot. The diagnostic procedure associated with the circular comparison based BIST architecture is described in more de-tail in Section 6.
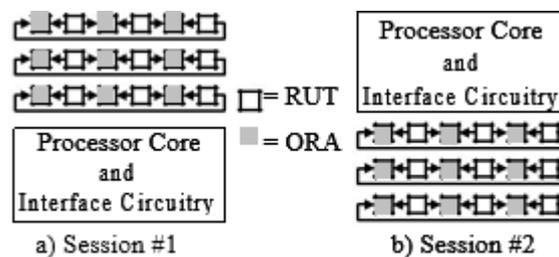

Figure 4. Processor Core Based BIST Sessions

In order to algorithmically and efficiently reconfigure the RUTs from the embedded processor core, the BIST archi-tecture, including TPG-to-RUT and RUT-to-ORA routing, for a given resource under test must be constant for all BIST configurations associated with that particular resource being tested. This not only helps to reduce the time re-quired to reconfigure the RUTs, but more importantly reduces the size of program memory needed to store the pro-gram for BIST reconfiguration. This is critical since the program memory is limited to half of the block RAMs in the FPGA. If the program memory size requires more than half of the block RAMs or the embedded processor core requires more than half of the PLBs, then the BIST architecture must be confined to a smaller portion of the FPGA (one-fourth of the array, for example); this increases the total number of BIST sessions and the resultant test time. Table 1 compares the additional resources needed for Power PC, Microblaze, and Picoblaze processors in terms of the number of slices and 18K-bit block RAMs required to implement our BIST approach including program mem-ory and processor access to the FPGA configuration memory via the Internal Configuration Access Port (ICAP). The ICAP interface requires approximately 170 slices and one block RAM. As can be seen from Table 1, the proc-essor core and associated circuitry can fit into half of the smallest Virtex-4 device such that the BIST approach can be applied to the programmable resources in the other half of the device. Note that the PowerPC requires more logic resources than the Microblaze, but

can operate at a higher clock frequency, although not at the advertised 450 MHz. While this helps to reduce the overall test time, the PowerPC is only incorporated in the FX family of devices such that the BIST approach would be limited to the devices in the FX family that contain two PowerPC cores. There-fore, we use the Picoblaze soft processor core and can apply the BIST approach to all Virtex-4 and Virtex II FPGAs.

Table 1. Resources Required for Embedded Processor Cores

Embedded Processor	Core Type	Maximum ClockFrequency	Slices	PLBs	Block RAMs
PowerPC	Hard	222 MHz	1000	250	9
Microblaze	Soft	180 MHz	940	235	9
Picoblaze	Soft	221 MHz	333	84	3
Picoblaze (optimized)	Soft	233 MHz	274	69	3

IV. BIST FOR PROGRAMMABLE LOGIC BLOCKS

Only half of the PLBs can be tested at a given time due to the use of other PLBs as ORAs.

Therefore, a total of two test sessions are required to completely test the logic in all of the PLBs in half of the array (a total of four test sessions to test all PLBs in the entire array). During the second test session the positions of the PLBs under test and the ORAs are swapped as illustrated in Figure 5. To test the PLBs, we use 12-bit TPGs constructed from DSPs programmed to continuously accumulate a constant value, 0x691, which produces test patterns with pseudo-exhaustive state coverage [5]. To limit loading on each TPG (and maximized the BIST clock frequency) in the large array sizes of Virtex-4, we construct a BIST circuit, including two DSP-based TPGs, in every four rows of PLBs driving alternate columns of PLBs under test. A total of 12 BIST configurations are needed to completely test all of the logic resources in the PLBs (excluding the LUT RAM modes of operation) with all four slices in a given PLB tested concurrently. Only 10 BIST configurations are needed to completely test the two slices that do not feature LUT RAM and shift register modes. Two additional BIST configurations are needed to test the shift register modes in the other two slices. Only the modes of operation of the PLBs under test change from one BIST configuration to the next.The LUT RAMs in half of the FPGA can be tested in one test session since only half of the slices of the PLBs con-tain LUT RAMs such that the other half can be used for ORAs. These RAMs can operate in four modes but only three BIST configurations are needed including 64×1 single-port, 32×1 single-port, and 16×2 dual-port RAM modes of operation. Each TPG is implemented by using a DSP as a counter which addresses an associate block RAM that is used as a ROM to contain the test patterns to be applied to the LUT RAMs. This BIST structure is repeated every four rows of PLBs in the FPGA with the two TPGs in a set of four rows of PLBs driving alternating columns of LUT RAMs under test. Only the LUT RAM mode of operation and test patterns stored in the block RAM change between BIST configurations. The RAM test algorithms stored in the block RAMs include March Y [8] to test the single-port RAM modes and March DPR algorithm [2] developed specifically for dual-port LUT RAM operation.

V. BIST FOR BLOCK RAMS AND DSPS

The block RAMs in half of the FPGA can be tested in one test session. The processor core or PLBs can be used as the TPG to implement RAM test algorithms. The ORA functions, on the other hand, are more efficiently imple-mented in PLBs due to the large number of RAM outputs to be monitored. There are sufficient PLBs in half of the array in all sizes of Virtex-4 FPGAs to facilitate testing the block RAMs concurrently. A total of ten BIST configu-rations, summarized in Table 2, are required to test the 18K-bit block RAMs in Virtex-4. Note that BDS indicates the application of background data sequences for detecting neighborhood pattern sensitivity and coupling faults in word oriented memories [7]. BDS is applied only during the first BIST configuration since once tested there is no need to repeat pattern sensitivity and coupling fault tests during the

subsequent BIST configurations. Therefore, the less time intensive RAM test algorithms can be used to simplify the TPG and to minimize the overall testing time. Programmable modes of operation are also tested including active edges and levels of clocks, clock enables, resets, registered inputs and outputs, various write/read options, and initialization of block RAM contents and registers.

Table 2. BIST Configurations for Block RAMs

BIST Configuration	Test Algorithm	Address Locations (*A*)	Data Width (*D*)	Clock Cycles
1	March LR [6] w/ BDS [7]	512	36	58×*A*
2	MATS+ [8]	8K	2	2×5×*A*
3		16K	1	2×5×*A*
4	March s2pf- [9]	512	36	14×*A*
5	March d2pf [9]	512	36	9×*A*
6	March Y FIFO [9]	4K	4	6×*A*
7		2K	9	6×*A*
8		1K	18	6×*A*
9		512	36	6×*A*
10	MATS FIFO ECC	512	64	3×*A*

TPG slice count = 608 ORA slice count = 72×*N* where *N* = number of block RAMs under test

While only eight BIST configurations are required to test the block RAMs in Virtex II FPGAs [4], Virtex-4 block RAMs provide additional modes of operation including FIFO and ECC modes [3]. For Virtex -4, the BIST configu-rations are partitioned into two sets. The first set tests the block RAMs in single and dual port modes while the sec-ond set focuses on testing the FIFO modes. The FIFO modes of operation are tested using a March Y based algo-rithm which includes testing status circuitry such as Full and Empty flags [8]. Dynamic partial reconfiguration via the embedded processor core facilitates efficient testing of the programmable Almost Full and Almost Empty flags in the FIFO mode of operation by reconfiguring the "Almost" values during the test sequence without requiring nu-merous downloads to the FPGA for the sole purpose of reprogramming threshold values for these flags. The ECC mode of operation, on the other hand, presents the interesting problem of detecting faults in a fault-tolerant circuit. Fortunately, bit errors can be introduced for the purpose of testing the bit error detection/correction circuitry by ini-tializing the RAM with errors in both data and Hamming code bits.

Testing the block RAMs presents another interesting issue since they are used to construct the program memory for the embedded processor core which is, in turn, used to configure and execute BIST. One approach is to test all block RAMs in the FPGA concurrently by individual downloads for each BIST configuration prior to performing the embedded processor based BIST approach. However, due to power dissipation considerations, partitioning the RAMs into subsets and testing each subset in a separate test session is preferred. We include a number of sanity checks in the processor based BIST approach, such as emulating faults in the resources under test by manipulating configuration memory bits and verifying that these injected faults are detected during a BIST configuration. This fault injection emulation capability also provides a mechanism for verifying and evaluating the fault detection capa-bilities of the BIST configurations as well as fault isolation capabilities of the diagnostic procedure.

The DSPs in half of the FPGA can also be tested in one test session with PLBs used to implement the TPG and ORA functions. Four BIST configurations are needed to test the DSPs and their associated programmable modes for active edges clocks, active levels of clock enables and resets, registered inputs and outputs, number of pipelining stages, and cascaded modes of interconnection. The other modes of operation are dynamically controllable by the TPG via the seven operational mode (OPMODE) inputs to the DSPs. Our approach to testing the DSP is to initially focus on testing the multiplier-accumulator (referred to as a MAC, or MACC), followed by configurations to test the other programmable modes. Testing the MAC can be accomplished by taking advantage of the fact that it can be used for both test pattern generation (as in the BIST for PLBs) and output response compaction [5]. Hence, the MAC can be tested by simultaneously performing test pattern generation and output response compaction such that the mode of operation itself is self-testing [5]. However, the continuous monitoring of the DSP outputs by the circu-lar comparison-based ORA structure detects and latches errors as they appear at the DSP outputs which eliminates the possibility of signature aliasing that can occur in arithmetic response compaction circuits.

VI.   RESULTS RETRIEVAL AND DIAGNOSIS

The BIST results in terms of the ORA contents can be retrieved at the end of each BIST sequence or, as a result of dynamic partial reconfiguration, at the end of a set of BIST sequences. While the latter approach helps to reduce overall test time, the trade-off is some minor loss in diagnostic resolution; the faulty resource(s) can still be identi-fied but the failing mode of operation the faulty resource cannot. The BIST results are retrieved by reading the frames that contain the contents of the ORA flip-flips. The number of frames that must be read is minimized in the Virtex-4 architecture since all flip-flops associated with a column of sixteen PLBs are located in a single frame. By orienting the BIST architecture such that ORAs reside in the same columns, the number of frames to be read is minimized. When the processor core reads the frames containing the ORA results and encounters failure indica-tions, the results can be stored in a data RAM and on-chip diagnosis can be performed by the processor core.

The diagnostic procedure is an extension and enhancement of the diagnostic procedure described in [4]. A basic as-sumption of that procedure was that there are no more than two consecutive RUTs with equivalent faults; consecu-tive RUTs in this context means that they are monitored by a common set of ORAs. However, we extend the proce-dure to indicate when this situation may have occurred and to reconfigure the circular comparison structure such that unique diagnosis can be obtained. The enhanced diagnostic procedure is as follows:

Step 1: Record the ORA results and set the faulty/fault-free status of all RUTs to unknown as indicated by an empty entry in the table.

Step 2: For every set of two or more consecutive ORAs with 0s, enter a 0 for all RUTs observed by these ORAs to indicate that those RUTs are fault-free.

Step 3: For every adjacent 0 and 1 followed by an empty cell, enter a 1 in the empty cell to indicate that the RUT is faulty. This step is repeated recursively while such entries exist.

Step 4: Consistency check: If an ORA indicates a failure but the RUTs on both sides of the ORA are determined to be fault-free, then one of the following conditions exist: *a*) there is a fault in the routing resources between one of the two RUTs and the ORA, *b*) the ORA is faulty, or *c*) there are more than two consecutive RUTs with equivalent faults. Condition *a* or condition *b* exits if there is only one ORA inconsistency in the circular comparison. However, if there are multiple ORA inconsistencies, then condition *c* may exist. In the latter case, reconfigure the circular comparison order and repeat the test and diagnostic procedure.

Step 5: If all RUTs have been marked as faulty or fault-free then a unique diagnosis has been obtained; otherwise, any RUT that remains marked as unknown may be faulty. In the latter case, we reconfigure the circular comparison order to compare different RUTs then repeat the test and diagnostic procedure Steps 1-5.

An example of the application of diagnostic procedure is illustrated in Table 3 where ORA_{ij} denotes an ORA com-paring the outputs of RUT_i and RUT_j. In this example, a unique diagnosis is obtained and, furthermore, RUT_3 and RUT_4 are found to be faulty with equivalent faults. The simplicity of this tabular based diagnostic procedure facili- tates straight forward implementation in and execution by the embedded processor core. While the new enhance-ments to Steps 5 and 6 of the diagnostic procedure remove the previous assumption of no more than two consecutive RUTs with equivalent faults, it should be noted that finding a minimum set of circular comparison configurations that guarantee unique diagnosis is still an open problem.

Table 3. Circular Diagnostic Procedure Example

Step	ORA_{61}	RUT_1	ORA_{12}	RUT_2	ORA_{23}	RUT_3	ORA_{34}	RUT_4	ORA_{45}	RUT_5	ORA_{56}	RUT_6
1	0		0		1		0		1		0	
2	0	0	0	0	1		0		1	0	0	0
3	0	0	0	0	1	**1**	0	**1**	1	0	0	0

7. EXPERIMENTAL RESULTS

We have implemented, downloaded, and verified the BIST configurations for PLBs and LUT RAMs on a Virtex-4 LX25 device to obtain our preliminary experimental results presented in this section. Total test time during BIST of FPGAs is dominated by two factors: downloading the BIST configurations and retrieving the BIST results at the end of the BIST sequence. We use the time required for full configuration of the FPGA and full configuration memory readback as a baseline for determining the speed-up in test time summarized in Table

4. The download time can be significantly reduced by partial reconfiguration where only those frames that have configuration data changes from one BIST configuration to the next need to be written. Similarly, the BIST results retrieval time can be reduced by partial configuration memory readback where only those frames that contain ORA flip-flops need to be read. These techniques can be applied externally when testing the full array. With embedded processor based BIST, only half of the array can be tested at a time, requiring full downloads of the initial configuration but only the frames in half of the array are written for BIST reconfiguration. Similarly only half of the frames are read for ORA results retrieval, but the complete process must be applied twice in order to test both halves of the array. Using the multiple frame write capability in Virtex-4, there is a further reduction in the number of frames that must be written. It should be noted that the number of frames for multi-frame write in Table 4 is given in terms of equivalent frames to account for the additional configuration register commands required to perform the multi-frame write operation. Using dy-namic partial reconfiguration, the contents of the ORA remain unchanged from one BIST configuration to the next such that failure indications can be accumulated in the ORA flip-flops over a sequence of BIST configurations with the results read at the end of the BIST session. This reduces the total number of frames that must be read with some minor loss in diagnostic resolution. When the embedded processor speed is taken in to consideration along with that fact that only one download to the FPGA is required for each BIST session with partial reconfiguration for subse-quent BIST configurations performed by the embedded processor, we obtain a final test time speed-up by a factor of

20. This speed-up is despite the fact that we are doubling the total number of BIST sessions since only half of the array can be tested during a given session. This speed-up in test time is comparable to the factor of 43 speed-up previously reported for the Atmel AT94K series configurable System-on-Chip which has a dedicated embedded processor such that the full array can be tested during a single BIST session [11].

Table 4. Embedded Processor Test Time Comparison

Method for FPGA Configuration and Readback		# Write Frames	# Read Frames	Speed-up		
				BIST Configu-ration Time	Results Retrieval	Total Test Time
Full Array External Test	Full Configuration/Readback	93,840	93,840	1	1	1
	Partial Configuration/Readback	22,656	1,512	4.1	62.1	7.8
Half Array Embedded Processor	Partial Configuration/Readback	35,168	1,512	2.7	62.1	5.1
	Multiple Frame Write	21,211	-	4.4	-	8.7
	Results at End of BIST Session	-	252	-	372.4	
	Processor Speed-up	9,104	108	10.3	867.6	20.4

VIII. SUMMARY AND CONCLUSIONS

Due to the complexity of most current FPGA architectures, algorithmic generation of a complete BIST configuration by the embedded processor core is not practical due to the program memory storage required. Therefore, a more ef-ficient approach is to download the initial BIST configuration for a given set of resources under test (such as PLBs, LUT RAMs, block RAMs, DSPs) and then use the embedded processor core for execution of the BIST sequence, retrieval and diagnosis of BIST results, and dynamic partial reconfiguration for subsequent BIST configurations for the target resource under test. This helps to minimize the number of downloads to the FPGA which is the dominant factor in testing time associated with FPGAs. However, in order to efficiently reconfigure the resources under test from the embedded processor for subsequent BIST configurations, it is important to make the BIST structure regular and reconfigurable without having to modify routing resources including the TPG-to-RUT and RUT-to-ORA con-nections. This allows the embedded processor to reconfigure only the RUTs, and in some cases the TPGs, which can be done algorithmically with minimal program memory and processor resource requirements.

The embedded processor core can also serve as the TPG during BIST configurations. However, there are factors that discourage this approach including TPG loading in large arrays, test pattern application speed, and fault detec-tion capabilities. The number of resources under test can be large, particularly in the case of PLBs and LUT RAMs. Increasing the loading on the TPG will reduce the maximum clock frequency for test pattern application. As a re-sult, using a larger number of TPGs with less loading helps to maximize the clock frequency which in turn helps to detect delay faults. When using the embedded processor core to generate algorithmic test patterns, as in the case of RAM test algorithms, it will take multiple processor clock cycles to produce each test pattern, particularly when the number of bits in the test pattern is greater than the bit size of the processor. This also reduces the effective clock rate at which the test patterns can be applied. Finally, a single TPG, as is the case when the embedded processor core is used, produces the possibility of a fault in the TPG or TPG-to-RUT routing preventing application of the ap-propriate test patterns that will detect faults in the resources under test. This problem is eliminated with multiple TPGs driving RUTs that are observed by the same set of ORAs since a fault in one of the TPGs or TPG signal paths will cause incorrect test patterns to be applied to one set of RUTs which, in turn, will produce mismatching output responses such that is it almost impossible for a set of faults to go undetected. As a result, the optimal approach is to limit the embedded processor core to execution and control of the BIST sequence, retrieval and diagnosis of BIST results, and dynamic partial reconfiguration for subsequent BIST configurations.

The use of an embedded processor core for BIST of FPGAs comes at the expense of doubling the number of BIST sessions since the positions of the embedded processor core and the BIST structure must be swapped in order to fa-cilitate complete testing of all programmable logic and memory resources in the FPGA. However, the time savings obtained (a factor of 20) through use of the embedded processor core for BIST execution, results retrieval, and re-configuration more than make up for the cost of doubling the number of BIST sessions. This is primarily due to the fact that externally downloading test configurations and retrieving the BIST results dominate the total testing time. In addition, external control and storage requirements required to download individual BIST configurations is sig-nificantly reduced by the use of the embedded processor core since only the initial BIST configurations must be downloaded with all subsequent BIST application and reconfiguration performed by the embedded processor. As a result, with the exception of BIST for PLBs since they require four test sessions, only two BIST configurations must be stored for each type of resource under test, one for each position of the embedded processor core in the array.

REFERENCES

[1] M. Abramovici and C. Stroud. "BIST-Based Test and Diagnosis of FPGA Logic Blocks." IEEE Trans. on VLSI Systems, 9(1): 159-172, 2001.

[2] C. Stroud, K. Leach, and T. Slaughter. "BIST for Xilinx 4000 and Spartan Series FPGAs: A Case Study." Proc. IEEE International Test Conf., 1258-1268, 2003.

[3] __, "Virtex-4 User Guide UG070v1.4." Xilinx, Inc., 2005.

[4] C. Stroud and S. Garimella. "BIST and Diagnosis of Multiple Embedded Cores in SoCs." Proc. International Conf. on Embedded Systems and Applications, 130-136, 2005.

[5] J. Rajski and J. Tyszer. "Arithmetic Built-In Self-Test for Embedded Systems." Prentice Hall PTR, 1998.

[6] 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

[7] 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

[8] M. Bushnell and V. Agrawal. "Essentials of Electronic Testing: for Digital Memory and Mixed-Signal VLSI Circuits." Springer, 2000.

[9] S. Hamdioui, Testing Static Random Access Memories, Springer, 2004

[10] __, "Compiled Megacell Testing Application Note 0696C." Atmel Corp., 1999.

[11] J. Sunwoo and C. Stroud. "Built-In Self-Test of Configurable Cores in SoCs Using Embedded Processor Dy-namic Reconfiguration." Proc. International SoC Design Conf., 174-177, 2005.