BUILT-IN SELF-TEST AND DIAGNOSIS OF MULTIPLE EMBEDDED CORES IN SOCS

Charles Stroud and Srinivas Garimella
Dept. of Electrical and Computer Engineering
Auburn University, Alabama, U.S.A.

Abstract - An efficient approach is presented for Built-In Self-Test (BIST) and diagnosis of embedded cores in System-on-Chip (SoC) devices using an embedded Field Programmable Gate Array (FPGA) core. The approach targets multiple regular structure cores including memories, multipliers, etc., but can be used to test any set of multiple identical cores in a SoC that also contains an embedded FPGA core with access to the cores to be tested. The approach can be used for manufacturing testing or in-system test and diagnosis for fault-tolerant applications. Experimental results are presented from the actual implementation of the BIST and diagnostic procedure in commercial configurable SoCs and FPGAs that contain multiple regular structure cores.[1]

Keywords: built-in self-test, design-for-test, test synthesis, embedded test and diagnosis, diagnostic procedure

1. Introduction

Built-In Self-Test (BIST) approaches have been developed for Field Programmable Gate Arrays (FPGAs) by programming some of the programmable logic blocks (PLBs) to function as Test Pattern Generators (TPGs) and Output Response Analyzers (ORAs) to test the remaining programmable logic and interconnect resources [1][2][3]. These techniques can be used to test embedded FPGA cores in SoC devices [4][5]. In addition, diagnostic procedures have been developed to identify the faulty PLBs in FPGAs [1][2] and can also be used for FPGA cores in SoCs. While it has been proposed that the FPGA core in SoCs can be used to test other cores in the SoC [4], this is only possible if the FPGA core has adequate interconnect access to the cores to be tested [5].

Commercial configurable SoCs typically incorporate embedded processor and FPGA cores along with multiple regular structure cores such as memories, multipliers, and digital signal processors [6][7][8]. In order to maximize the efficiency of these devices in a large variety of applications, these regular structure cores are also programmable. For example, Random Access Memory (RAM) cores can be programmed for a range of address and data bus widths, synchronous and asynchronous write and read modes, and single and multiple port operation. Furthermore, there are usually multiple instances of these programmable regular structure cores. As a result, testing these cores presents a problem in that all of these cores must be tested in their various modes of operation, requiring multiple reconfigurations of the device for complete testing. Coupled with large configuration memory sizes, testing time and cost can be considerable both in manufacturing and system-level test. The ability to test multiple cores in parallel helps to minimize the testing time and cost. In-system BIST and diagnosis is an essential requirement for fault-tolerant applications since, once identified, faulty components can be avoided by reconfiguration of the intended system function in the device.

In this paper, an efficient BIST approach and diagnostic procedure is presented for multiple embedded regular structure cores in configurable SoCs. The implementation of the approach uses the Xilinx Virtex II Pro series SoC as the target device but this approach can be applied to most configurable SoCs. A brief overview of BIST and diagnosis of FPGAs upon which this approach is based is presented in Section 2. A description of the proposed BIST architecture for SoC cores is presented in Section 3 followed by its application to the embedded RAMs and multipliers in the Virtex II Pro in Section 4. The diagnostic procedure is presented in Section 5. Experimental results from actual implementations in Xilinx Virtex II Pro, Virtex I, and Virtex 4 are presented in Section 6 and the paper concludes in Section 7.

---

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

2. Background in FPGA Logic BIST and Diagnosis

The architecture for BIST of PLBs in an FPGA (Figure 1) consists of a column (or row) of PLBs configured as two identical TPGs that supply pseudo-exhaustive test patterns to alternating columns (or rows) of identically configured PLBs under test (BUTs) [1]. The output responses of the BUTs are monitored by comparison-based ORAs (Figure 2) to detect and latch mismatches due to faults. The purpose of the two identical TPGs is to detect any faults in the PLBs used to construct the TPG. The BUTs are repeatedly reconfigured and tested in their various modes of operation until completely tested. The set of BIST configurations required to completely test the BUTs is referred to as a test session. Since only half of the PLBs are configured as BUTs, two test sessions are required to test all PLBs in the FPGA. With the exception of the PLBs along the edge of the array, all BUTs are monitored by two ORAs which compare the output response of the BUT with those of two different BUTs. As a result, there are very few cases where faults in the array of PLBs can go undetected. Based on this BIST architecture, a diagnostic procedure called MULTICELLO (multiple faulty cell locator) was developed to identify faulty BUTs based on the failing BIST results [1]. In most cases, MULTICELLO can achieve unique diagnosis, correctly identifying faulty and fault-free PLBs. However, along the edges of the array, where BUTs are observed by only one ORA, unique diagnosis is more difficult to achieve, requiring rotation of the BIST architecture by 90° with reapplication of the BIST configurations and the MULTICELLO algorithm.

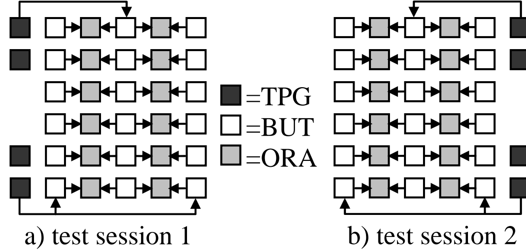

Figure 1. FPGA logic BIST architecture

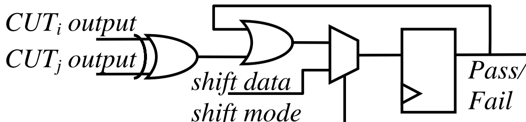

Figure 2. ORA design

An on-line BIST approach for PLBs in FPGAs was later developed for on-line fault-tolerant applications [2]. The MULTICELLO algorithm was extended for application to the combined results from eight on-line test sessions in a small 4×2 array of PLBs (Figure 3). In this arrangement, all BUTs are monitored by two ORAs and compared with the output responses of two different BUTs. However, the MULTICELLO diagnostic algorithm was never extended for use beyond its application to this 4×2 array.

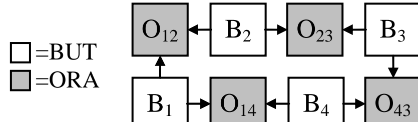

Figure 3. Combined on-line logic BIST test sessions

3. SoC Multiple Core BIST Architecture

The proposed multiple core SoC BIST architecture (Figure 4) consists of PLBs configured as a TPG which supplies algorithmic test patterns to multiple identically configured cores under test (CUTs). The outputs from pairs of CUTs are monitored by a set of ORAs with one ORA for each output of the CUT. Each ORA consists of a comparator with a flip-flop and an OR gate for latching any mismatches observed as a result of faults in the core(s) until the BIST results can be retrieved at the end of the BIST sequence via a scan chain incorporated in the ORA design (Figure 2). Alternatively, the contents of the ORA flip-flops can be retrieved via partial configuration memory read back [5]. This BIST architecture has several important differences from that of the off-line logic BIST architecture for FPGAs [1][4][5]. First, all identical cores can be test simultaneously in one test session such that test time is independent of the number of cores in the SoC. Second, only one TPG is needed in the BIST architecture assuming that the logic and routing resources in the FPGA core have been tested before being used to test the other embedded cores. This is important since the algo-

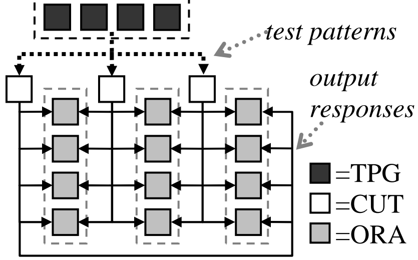

Figure 4. Multiple core BIST

rithmic TPGs needed for testing embedded cores, such as memories for example, are typically more complicated than those used to test PLBs and require a large number of PLBs for implementation. Therefore, it may be impossible to fit two TPGs in smaller FPGA cores. The final distinction is that all CUTs are monitored by two ORAs and compared with the outputs of a different CUT in each ORA in a similar manner to the on-line logic BIST architecture. As a result, there is no loss in diagnostic resolution as is the case off-line FPGA logic BIST along the edges of the array. Furthermore, there is an ORA for each output of the CUT rather than an ORA for multiple outputs of the BUT as in the case in off-line [1] and on-line [2] logic BIST; this increases diagnostic resolution for fault-tolerant applications.

4. BIST for RAMs and Multipliers

The Virtex II Pro series SoC architecture consists of an FPGA core, from 12 to 444 18Kbit dual-port RAM cores, from 12 to 444 18-bit multiplier cores, and 1 or 2 PowerPC processor cores [6]. The 18 K-bit RAMs are dual-port RAMs that can be programmed in six sizes ranging from 16K×1-bit to 512×36-bits [6]. A total of eight RAM BIST configurations (6 single-port and 2 dual-port) are required to completely test these RAMs. In each single-port configuration, both ports of the dual-port RAM are tested in turn using the *March LR* algorithm [9]. Background data sequences (BDS) are used to test the RAMs for coupling faults and pattern sensitive faults, but the use of BDS in all configurations is redundant. The *March LR* algorithm will detect coupling and pattern sensitivity faults in single-bit memories. However, the 16K×1-bit mode does not provide access to all 18K-bits of the RAM. Therefore, to test the complete RAM for coupling and pattern sensitivity faults, and to test for bridging faults in the write and read data busses of the RAM, *March LR* with BDS is used to test the 512×36-bit mode. Since the RAMs are dual-port RAMs, two additional march algorithms (*March* d2pf and *March* s2pf [10]) are used to test dual-port access of the RAMs. During these eight RAM BIST configurations, the other programmable features are tested including active clock edge and active level for control signals, such as the RAM write enable, clock enable, and synchronous set/reset, along with other programmable options such as write-first, read-first and no-change on the output data port during write operations.

The 18-bit multipliers in the Virtex II Pro can be programmed as combinatorial or registered multipliers. A total of three BIST configurations are required to test all modes of operation, one combinatorial and two registered to test the active clock edge and active level of the clock enable and reset control signals. The multipliers must be tested algorithmically since the 38 inputs (including two 18-bit data values, clock enable, and reset for the synchronous mode of operation) makes exhaustive test patterns impractical. The multipliers use the modified Booth algorithm [6], hence we construct a TPG which produces a test pattern sequence known to provide high fault coverage (>99%) in modified Booth algorithm multipliers [11][12]. This requires an 8-bit counter in the combinatorial mode of operation [11]. For the registered modes, the TPG algorithm is modified to include a 10-bit counter to test the clock enable and reset control inputs.

To implement these BIST approaches, parameterized VHDL models were developed for the TPGs and ORAs that can be used to test any number and size of multiple embedded RAM and multiplier cores in any FPGA or SoC. In SoC applications, the input and output ports of the embedded cores must be accessible from the FPGA core to be tested completely. Some support is needed from synthesis tools to control the physical placement of the CUTs with respect to their ORA logic for diagnosis of faults in the embedded cores. With control over the physical location of the CUTs, there is no need for control of placement and routing of the PLBs used for ORAs and the TPG such that the typical design flow of VHDL synthesis followed by automatic placement and routing in the FPGA core can be used to automate the BIST development process and minimize the development effort. Xilinx synthesis tools allow this control of the physical location of cores via a constraint file. Therefore, this parameterized VHDL approach is successful for testing RAM and multiplier cores in Xilinx devices including the Virtex II Pro SoC as well as Virtex I, Spartan II, Virtex II and Spartan III FPGAs as well as Virtex 4 FPGAs and SoCs. Furthermore, the BIST configurations can be generated in a matter of minutes compared to the months typically required for developing BIST configurations for programmable logic and routing resources in FPGAs.

5. Diagnostic Procedure

Once the BIST configurations have been downloaded and executed, the faulty core(s) can be identified based on the BIST results retrieved from the ORAs. The diagnostic procedure is based on the BIST architecture where the outputs of two cores are compared by a given set of ORAs to detect mismatches that result from faults in one or both of the cores. It is possible that equivalent faults in two cores being compared by the same set of ORAs will go undetected by that ORA. However, all cores are being observed by two sets of ORAs and are being compared to a different core in each set of ORAs such that the few combinations of faulty cores that can go undetected by the BIST approach is highly improbable (for example, if all cores in the SoC have equivalent faults, this would go undetected). The diagnostic procedure is an extension of the original MULTICELLO algorithm developed for diagnosing faulty PLBs in FPGAs [1][2]. The extended MULTICELLO procedure assumes there are at most two consecutive CUTs in the circular interconnection of alternating CUTs and ORAs with equivalent faults; supporting theorems and lemmas for this assumption and the diagnostic procedure are the same as those of [1] and [2]. The steps of the diagnostic procedure (with an accompanying example applied to eight CUTs in Table 1) are as follows:

Step 1: Record the ORA results and set the faulty/fault-free status of all CUTs as unknown as indicated by an empty entry in the table. This is illustrated in Step 1 of the diagnostic Example A of Table 1 where eight cores have been tested. The rows are denoted as C_i for cores C_0 to C_7 and as O_{ij} for ORAs where i and j denote the cores (C_i and C_j) being compared by the ORA. Note that the ORA denoted as O_{70} has entries at the top and bottom of the table to denote the circular nature of the comparison. A '1' in an ORA row entry indicates that a failure was observed by at least one of the ORAs associated with each output of the core. A '0' indicates that no failure was observed by the ORA.

Step 2: For every set of two or more consecutive ORAs with 0s, enter a 0 for all CUTs observed by these ORAs to indicate that the CUTs are fault-free. This is illustrated in Step 2 of Example A in Table 1. At this point, we have determined that six of the eight cores (C_0-C_3 and C_6-C_7) 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 core is faulty. This is illustrated in Step 3 of Example A in Table 1. At this point we have determined that two cores are faulty (C_4 and C_5).

Step 4: Consistency check: If an ORA indicates a failure but the CUTs on both sides of the ORA are determined to be fault-free, then there is a fault in the routing resources between one of the two CUTs and the ORA. This is illustrated in Step 4 of Example A in Table 1 where there is an inconsistency in ORA O_{01} indicating a fault in the ORA or in the associated routing between the ORA and one (or both) of the cores C_0 or C_1. Note we do not assume routing associated with the cores has been tested.

Step 5: If all CUTs have been marked as faulty or fault-free then a unique diagnosis has been obtained; otherwise, any CUT that remains marked as unknown may be faulty. A unique diagnosis has been obtained in Step 5 of Example A of Table 1. Furthermore, it has been determined that cores C_4 and C_5 have equivalent faults since there were no failures in ORA O_{45}.

In Example B of Table 1, a unique diagnosis is not obtained since core C_4 could be faulty or fault-free. If C_4 is faulty, it does not have equivalent faults with either of the other two known faulty cores (C_3 and C_5). Example C of Table 1 illustrates a scenario where no diagnosis can be obtained; we cannot determine which cores are faulty and which, if any, are fault-free. For example, among other scenarios, 1) all cores could be fault-free with ORA inconsistencies in O_{01}, O_{23}, O_{45}, and O_{67}, 2) all cores could be faulty with equivalent faults in C_0 and C_7, C_1 and C_2, C_3 and C_4, and C_5 and C_6, 3) cores C_1, C_2, C_5, and C_6 could be fault-free with equivalent faults in C_0 and C_7 and in C_3 and C_4., or vice versa. Another scenario with no diagnosis would be where all ORAs indicate failures. However, these scenarios are unlikely to occur in practice and the likelihood decreases as the number of CUTs increases. When a unique diagnosis cannot be obtained, the pair-wise comparison of cores by the ORAs can be changed by changing the core location in the physical constraint file, re-synthesizing, downloading, executing the new BIST configuration, and re-applying the diagnostic procedure while taking the results of the previous diagnosis into account.

Table 1. Diagnostic Procedure Example

Step	Example A					Example B					Example C				
	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
O_{70}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C_0		0	0	0	0		0	0	0	0					
O_{01}	1	1	1	1	1	0	0	0	0	0	1	1	1	1	1
C_1		0	0	0	0		0	0	0	0					
O_{12}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C_2		0	0	0	0		0	0	0	0					
O_{23}	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
C_3		0	0	0	0			1	1	1					
O_{34}	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0
C_4			1	1	1										
O_{45}	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
C_5			1	1	1			1	1	1					
O_{56}	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0
C_6		0	0	0	0		0	0	0	0					
O_{67}	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
C_7		0	0	0	0		0	0	0	0					
O_{70}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

It should also be noted that the examples shown here assume a single ORA failure indication for a complete *N*-output core. In reality, the diagnostic algorithm is applied to the *N* ORAs, each associated with one output of the core. As a result, it is more likely that unique diagnosis will be obtained without the need for re-synthesis since any known faulty output of a core would indicate that the core is faulty. Furthermore, the per-output comparison by the ORAs facilitates identification of the faulty bit(s) of a RAM core, for example, for fault-tolerant applications by reconfiguring the system function to use other bits of the RAM when the RAM is not fully utilized.

This diagnostic procedure and supporting examples show that the MULTICELLO algorithm can be extended beyond the 4×2 array used in [2]. In addition, due to the circular nature of the CUT to ORA connections, there is no loss in diagnostic resolution as was the case in the original off-line version of the algorithm for PLBs in [1]. An alternative BIST and diagnostic approach is to generate (or store) the expected output responses in the TPG for direct comparison with the output of the individual CUTs. This ensures that a unique diagnosis can always be obtained with a much less complicated diagnostic procedure since the position of the ORA indicating a failure uniquely identifies the failing core and failing output of the core. This is possible with algorithms such as *March LR*. However, for some cores and their associated test algorithms, the complexity of generating (or storing) expected results can far exceed the complexity of test pattern generation and make the TPG too large to fit in the embedded FPGA core.

6. Experimental Results

We have successfully implemented and downloaded the eight BIST configurations in a Virtex II Pro XCVP30 to test all 136 RAMs simultaneously. The 136 multipliers are then tested simultaneously by a separate set of three BIST configurations. Implementing the BIST approach with the capability to test all of the embedded cores of a given type simultaneously requires sufficient PLBs in the embedded FPGA core to implement the TPG and ORA functions. The actual number of slices required for implementing each TPG and ORA in a Virtex II Pro as well as Virtex II and Spartan III FPGAs are shown in Table 2 for each of the eight RAM and three multiplier BIST configurations. The PLB count can be obtained by dividing the slice count by 4 since there are 4 slices per PLB in these devices. It should be noted the slice count given in Table 2 represents a minimum number required for BIST since as the number of RAM cores increases, additional slices are needed to buffer the heavily loaded nets driven by the TPG. Alternatively the TPG can be replicated such that each TPG drives a subset of the RAM cores.

Table 2. BIST Slice Count for Virtex II Pro

RAM BIST					
BIST config	Test Algorithm	BDS	Data Width	TPG Slices	ORA Slices
1	March LR	No	1	62	N×2D
2			2	62	
3			4	64	
4			9	64	
5			18	68	
6	March LR	Yes	36	174	
7	March s2pf	No	36	64	
8	March d2pf	No	36	113	

Multiplier BIST				
BIST config	Test Algorithm	Mode	TPG Slices	ORA Slices
9	Count [10]	combinational	8	N×36
10	Modified count	registered	10	
11	Modified count	registered	10	

N = Number of Multiplier Cores
D = Data Width

The size of the TPG is a function of the algorithm being implemented. The largest TPG size (174 slices) is used for testing the RAM and multiplier cores in the Virtex II Pro is the *March LR* with BDS for the 512×36-bit RAM mode of operation. While the size of each ORA is small (only one slice per ORA), the number of ORAs, N_{ORA}, is given by $N_{ORA}=N_{CUT}\times O_{core}$, where N_{CUT} is the number of cores under test and O_{core} is the number of outputs of each core. The largest number of ORAs also occurs in BIST of the 512×36-bit RAM mode of operation and, since the RAMs are dual-port, there are a total of 72 outputs per RAM (hence the N×2D in Table 2). The number of slices available in the various Virtex II Pro devices is illustrated in Figure 5 along with the total number of slices needed to implement BIST in those devices. The number of RAMs and multipliers range from 12 in the XC2VP2 to 444 in the XC2VP100 but, as can be seen from Figure 5, there are sufficient logic resources in all devices to accommodate simultaneous BIST of all RAMs in the device. Due to power dissipation considerations, however, it may be important to partition the RAMs into subsets and test each subset in separate test sessions.

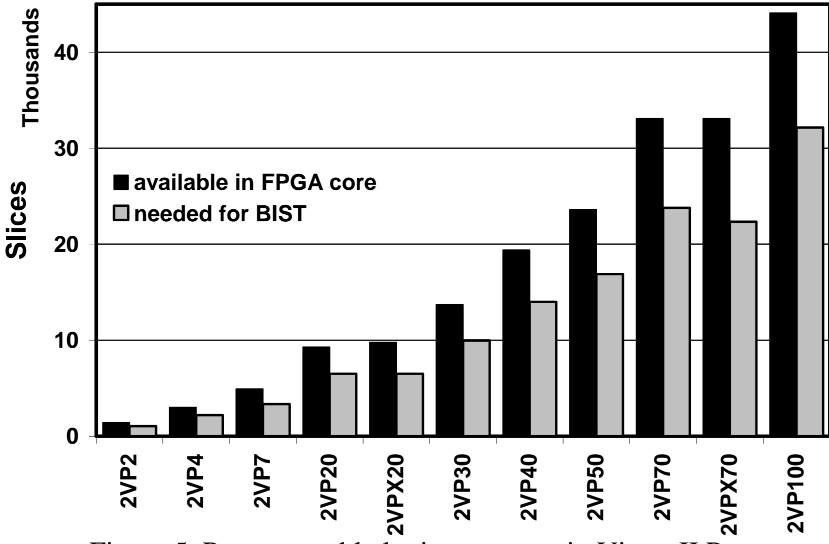

Figure 5. Programmable logic resources in Virtex II Pro

The BIST implementation yields slightly higher slice counts for the embedded RAM cores in Virtex I and Spartan II FPGAs due to less functionality per PLB for those FPGAs. In addition, the PLB count in Virtex I and Spartan II FPGAs is doubled since there are two slices per PLB in these devices. The RAMs cores in Virtex I and Spartan II are 4Kbit in size with a maximum data width of 16. As a result, *March LR* with BDS, *March s2pf* and *March d2pf* algorithms are implemented on a 16-bit wide RAMs and the slice counts obtained are 110, 49, and 76 respectively. We have successfully implemented and downloaded the seven RAM BIST configurations needed to simultaneously test the 8 and 14 RAM cores in a Spartan II XC2S50 and Virtex I XCV200, respectively. There are no multipliers in Virtex I and Spartan II FPGAs.

The Xilinx Virtex 4 series of FPGAs and SoCs contain from 36 to 552 RAM cores of the same size as those in Virtex II but with additional modes of operation such as a First-In First-Out (FIFO) memory [8]. There are sufficient logic and routing resources in all Virtex 4 devices to facilitate implementation and application of this BIST and diagnostic approach to all RAM cores simultaneously, assuming that power limitations can be satisfied. In addition to the eight BIST configurations used for the RAMs in Virtex II Pro, additional BIST configurations will be required to test the remaining modes of operation. Instead of multipliers, Virtex 4 contains digital signal processor (DSP) cores (referred to as Xtreme DSPs in Xilinx terminology) than number from 32 to 512 depending on the device. We are currently investigating BIST for these DSP cores and estimate a slice count of approximately 175 for the TPG. When combined with the slice count for ORAs, it appears that there are sufficient resources to test all DSP cores simultaneously in all Virtex 4 devices except the three 4VSX series devices which require two test sessions.

7. Summary and Conclusions

An efficient approach to BIST and BIST-based diagnosis of multiple identical embedded cores in configurable SoCs that contain an embedded FPGA core has been presented. The BIST approach and the diagnostic procedure are based upon the prior work done in BIST for FPGAs. However, the lower utilization of logic and routing resources needed for testing embedded cores facilitates a number of improvements in the BIST and diagnosis previously used for FPGAs. By including an extra set of ORAs, the outputs of all cores under test can be monitored by two ORAs with each ORA comparing the outputs to a different core under test. This not only improves the fault detection capabilities of the BIST approach, but also improves the ability to obtain unique diagnosis. The diagnostic procedure presented is an extension of the MULTICELLO algorithm originally developed for off-line BIST and diagnosis of PLBs in FPGAs. The algorithm was later extended to a small 4×2 array of PLBs during on-line BIST and diagnosis for fault tolerant applications. In this paper, we have further extended the diagnostic algorithm to apply to multiple identical embedded cores in SoCs. We have successfully implemented this approach for Xilinx Virtex I, Spartan II, Virtex II, Spartan III, Virtex II Pro, and Virtex 4 series FPGAs and SoCs.

8. 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] M. Abramovici, C. Stroud, and J. Emmert, “On-Line BIST and BIST-Based Diagnosis of FPGA Logic Blocks,” *IEEE Trans. on VLSI Systems*, Vol. 12, No. 12, pp. 1284-1294, 2004.

[3] C. Stroud, *A Designer’s Guide to Built-In Self-Test*, Kluwer Academic Publishers, 2002.

[4] M. Abramovici, C. Stroud, and J. Emmert, “Using Embedded FPGAs for SoC Yield Improvement,” *Proc. ACM/IEEE Design Automation Conf.*, pp. 713-724, 2002.

[5] C. Stroud, J. Sunwoo, S. Garimella and J. Harris, “Built-In Self-Test for System-on-Chip: A Case Study”, *Proc. IEEE International Test Conf.*, pp. 837-846, 2004.

[6] __, “Virtex II Pro and Virtex II Pro X Platform FPGAs,” Datasheet DS083, Xilinx Inc., 2004.

[7] __, “AT94K Series Field Programmable System Level Integrated Circuit,” Datasheet, Atmel Corp., 2001.

[8] __, “Virtex 4 Family Overview,” Datasheet DS112, Xilinx Inc., 2004.

[9] 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.

[10] S. Hamdioui and A. Van de Goor, “Efficient Tests for Realistic Faults in Dual-Port SRAMs”, *IEEE Trans. on Computers*, vol. 51, no. 5, pp. 460-473, 2002.

[11] D. Gizopoulos, A. Paschalis and Y. Zorian, “Effective Built-In Self-Test for Booth Multipliers,” *IEEE Design & Test of Computers*, vol. 15, no. 3, pp. 105-111, 1998.

[12] M. Psarakis, D. Gizopoulos, A. Paschalis, N. Kranitis and Y. Zorian, “Robust and Low-Cost BIST Architectures fro Sequential Fault Testing in Datapath Multipliers,” *Proc. IEEE VLSI Test Symp.*, pp. 15-20, 2001.