Autonomous Built-in Self-Test Methods for SRAM Based FPGAs

Steven Kopman, Student
Department of Electrical and Computer Engineering
University of Central Florida
Orlando, FL 32816-2450
skopman@knights.ucf.edu

***Abstract*— Built-in Self-Test (BIST) approaches for Static Random Access Memory (SRAM) based Field Programmable Gate Arrays (FPGAs) must be capable of fully testing the resources in the device. A summary of current techniques is presented in this paper that covers the three main components of modern FPGAs: logic blocks, interconnects and embedded FPGA cores. Overhead requirements, coverage capability, transient detection and BIST evaluation times are evaluated for each approach.**

Keywords: BIST, SRAM, FPGA, self test, interconnects, logic blocks, embedded cores

I. INTRODUCTION

Field Programmable Gate Arrays (FPGAs) are widely used in commercial, government and space applications where the flexibility of a custom logic design and reconfigurability are required. The reliability and sustainability of these devices, especially in applications such as small landers or planetary rovers [1] where physical access to replacement and repair of faulty devices is impossible, is paramount to mission success.

Faults that occur in SRAM based FPGA devices can be classified into two types, transient and permanent. A transient fault is a fault that disappears after a certain time [1]. Transient faults can flip a bit in the configuration memory or a flip-flop in a configurable logic block (CLB) resulting in a temporary unexpected behavior. Permanent faults will not disappear after a certain time. These faults cause a block to output incorrect values all the time. The distinction between the two is an important aspect of fault detection and is especially important when performing a built-in self-test (BIST). Annotating a block as permanently faulty when it was actually a transient fault may preclude its use as a spare block if another fault occurs.

Transient or permanent faults can occur in any component of the FPGA. Modern FPGAs, such as the Xilinx Virtex-6, contain many internal components such as 6-input Look-up Tables (LUTs), dual-port memories, DSP48E multiply-accumulate (MAC) blocks in addition to standard flip-flops. The flip-flop/LUTs in these devices are organized as slices containing LUTs, flip-flops, multiplexors and connection logic and the slices are organized in pairs called CLBs [15]. All of these components are equally likely to exhibit a fault so the testing scheme needs to be capable of validating the correctness of each one.

This paper summarizes BIST techniques that specifically address each one of the main components of modern FPGAs. An overview of these components is presented in Section II. Each technique may address one or all of the components listed above. Section III will describe the different techniques in detail. Section IV compares the techniques based on their overhead requirements, transient detection, coverage capability and their evaluation times. Table I lists the evaluation metrics in detail. Section V summarizes and concludes the paper.

TABLE I EVALUATION METRICS

	Metric	**Description**
Fault Type	Distinguish Transients	Detection of a transient fault vs. a permanent fault
Coverage Capability	Logic Blocks	Detection of faults in a LUT or register
	Interconnects	Detection of faults in the routing resources of the FPGA
	Embedded Cores	Detection of faults in embedded cores (i.e. FIFOs, memory, DSP blocks)
Overhead Requirements	Processor, Memory	Required hardware to implement the BIST technique external to the FPGA
Evaluation Time	Latency of Detection	Amount of time required to detect a fault
	Number of Configurations	Required configurations to complete BIST

II. FPGA COMPONENT OVERVIEW

Modern FPGAs are highly advanced, digital logic devices that are designed for high-speed, logic, communications and DSP functions. Most FPGAs are designed as columns and/or rows of Input/Output Blocks (IOBs), CLBs, memories, clock modules and routing resources. The most basic element of a

reprogrammable FPGA is the CLB. The CLB generally consists of at least one set of a LUT, register/latch and multiplexor (MUX). Figure I shows a basic 4-input CLB. The LUT is a SRAM device that can be programmed to perform any Boolean function of x inputs, where x is the number of inputs to LUT. The current Xilinx Virtex-5 and Virtex-6 devices have 6-input LUTs while older FPGAs usually contain 4-input LUTs. The flip-flop can be used to store the output of the LUT and output synchronously through the output MUX. The MUX selects the CLB output either from the flip-flop output port or the LUT output. Current devices CLBs, such as the ones in the Xilinx Virtex-6, contain eight LUTs and 16 flip-flops [15].

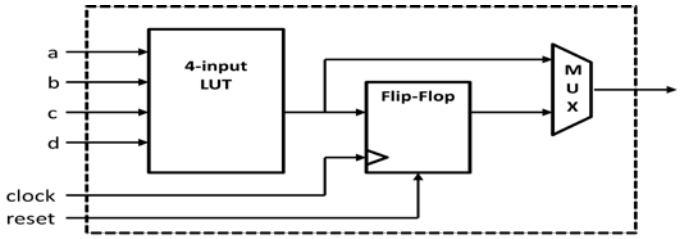

FIGURE I BASIC 4-INPUT CLB

CLBs are connected to one another via wire segments, switch boxes and connection boxes [10]. The routing resources provided by these components allow the FPGA to create a complex set of functions utilizing multiple CLBs. The wire segments also connect the CLBs to the IOBs and to embedded components within the device.

Embedded components in FPGAs have grown from simple dual-port memories to advanced Phase Locked Loops (PLLs), embedded processors and DSP emulating multiply-accumulators. Utilizing these blocks reduces the logic resources need to generate complex designs by implementing them in dedicated, timing-optimized resources. The embedded components also reduce the complexity of circuit card design by embedding processor and DSP functions inside of the FPGA.

III. BIST TECHNIQUES

Figure II shows that BIST Techniques can be broken into groups by their coverage capability. The first group is techniques that only handle logic resources. The second group is those that only can test interconnect resources. The third group is those that can only handle embedded component testing. Finally, the fourth group is those techniques that can test at least two of the three resources. A further breakdown into on-line and off-line approaches could be performed, however most BIST techniques that are "on-line" require some off-line time and therefore are loosely defined as on-line. For the purpose of this paper, all techniques will be considered off-line.

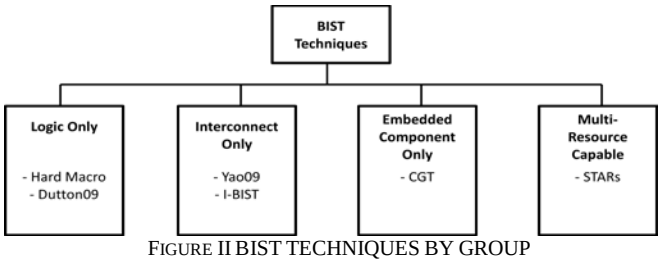

FIGURE II BIST TECHNIQUES BY GROUP

TERMINOLOGY

The terminology used among the varied BIST techniques is very consistent. The test is performed on a Block Under Test (BUT) or Wire Under Test (WUT) that is driven by the outputs of a Test Pattern Generator (TPG). The outputs of the BUT or WUT are compared in an Output Response Analyzer (ORA) to detect a fault [1][2][7][10][16][18].

In a few instances overlapping terminology is used between techniques although it is named differently. In [7], the BUT is replaced by the term Cell Under Test (CUT). The terms Self-Testing AArea (STAR) [1] and ROving TEster (ROTE) [7] are both used to describe a roaming test block that traverses a FPGA fabric. Additional varied terminology will be discussed in each techniques section.

A. Logic Only BIST Techniques

Logic only BIST techniques are designed to specifically test the LUT, flip-flop and/or MUX within a logic block. The tests that are performed cannot distinguish if an articulated fault occurred due to the input or output wire segments connected to the logic block or if it was the logic block itself. The entire block will be marked as faulty if a fault is detected.

1) Use of a Hard Macro

The use of a hard macro proposed in [16] requires utilizing specialized components of a specific family of FPGAs to generate parameterized, user-defined circuits to meet desired design specifications. This method takes advantage of the homogeneous architecture of a particular FPGA family. The target of this research is the Xilinx Virtex family of FPGAs.

The hard macro method uses the Xilinx FPGA Editor software and the physical-macro library to create circuits for the BIST implementation. The nmc files generated by the physical-macro library provide low level control over logic resources unlike using an abstracted language such as VHDL. This provides simpler logic for BIST configurations.

The BIST approach utilizes individual columns made up of TPGs, BUTs and ORAs. There are two identical TPGs composed of a LFSR creating a pseudoexhaustive test generator. Figure III shows the CLBs in columns in their test configurations. Two test sessions are run on each column set and the roles of the CLBs are reversed in the second test. In the first test, the TPGs use the first column of the test set as the TPGs. The next five columns are alternating sets of BUTs and ORAs, with three BUTs and two ORAs. The ORAs

compare outputs of the two neighboring BUTs and the outputs of the ORAs are passed on through the Scan Chain. The second test moves the BUTs and ORAs one column to the right and moves the TPGs to the last column of the test set.

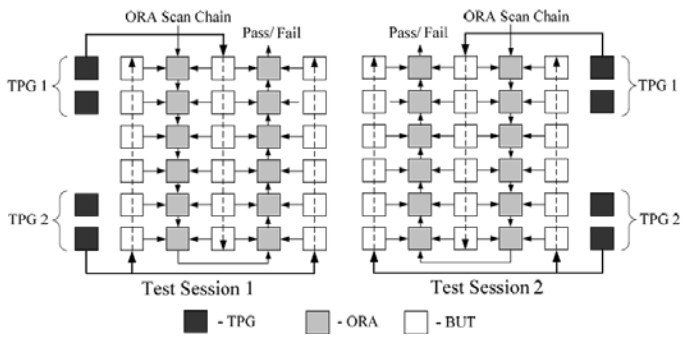

FIGURE III BIST ARCHITECTURE FOR THE HARD MACRO METHOD [16]

The Xilinx Virtex FPGA has 4-input LUTs and a CLB contains four LUTs, two in each logic cell. The configuration with a hard macro requires 12 test cases for 100% fault coverage. This provides pseudoexhaustive coverage of the 16 possible test cases. Figure IV shows the fault coverage total for the 12 configurations per session. Each of the configurations requires only four outputs to be observed by the ORA. The tests are composed of predefined macros, each covering a different component of the CLB. The configurations include a test of the Shift Register, five Arithmetic and MUX tests and six tests of FFX and FFY. The total number of tests for the CLBs is 24, with 12 tests per session.

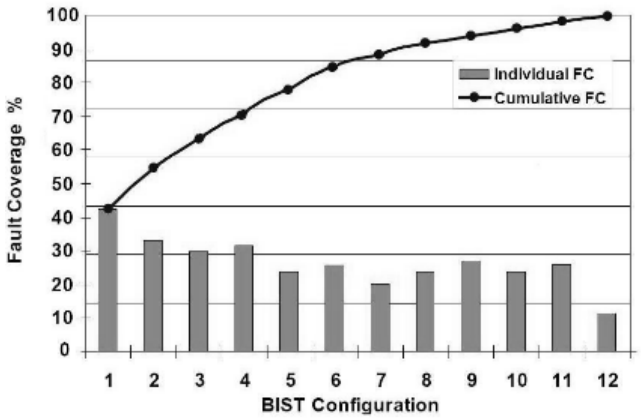

FIGURE IV BIST FAULT COVERAGE [16]

The test time required for the BIST is based on the number of clock cycles needed to generate the test patterns and implement the ORA scan chain. The sum of these is the total test time. The total scan time to read out an entire FPGA consisting of m rows and n columns is $\left(\frac{n}{2}-1\right) x\, m\left(\frac{n}{2}\text{-}1\right)$ xm clock cycles. Using the internal 50MHz clock of the Virtex XCV50 FPGA with a 24x16 CLB array [12], the verification requires 3.9 µs.

The hard macro method is extensible to other FPGAs however each FPGA's macros must be generated independently due to their varied architectures. In addition, the scan time, and thus the test time, of larger FPGAs grows linearly but significantly. For example, a Virtex XCV1000 FPGA with a CLB array of 64x96, the scan time is approximately 60 µs. A Virtex-4 XC4VLX200 with an array of 192x116 has a scan time of approximately 219 µs. While these times are not significant in themselves, if the BIST is to be useful during the operation of a mission it needs to be run a number of times. Since the FPGA must be taken off-line for each of these tests a large number of outputs cannot be realized.

2) Configurable Logic Block Testing

The application of BIST to the in-system test of a Xilinx Virtex-5 FPGA is proposed in [1]. Specifically, the method suggested in this paper applies to the test of a CLB within the device. Previous approaches have been applied to older Xilinx parts such as the Spartan and Virtex FPGAs; however this method improves the ORA to reduce to total number of required tests. The blocks used in the CLB test are the ORA, TPG, BUT and LUT.

The device used in [3] is a Virtex-5 that contains a 6-input LUT, flip-flop/latch, and multiplexor in each basic logic element. The LUT can be configured as a set of functions, a shift register or a distributed RAM. A CLB consists of two slices, each of which contains four logic elements. The LUTs can be cascaded to form a 128-bit shift register or in parallel to form a 16x8 shift register.

The BIST approach utilizes rows of ORAs and BUTs in a circular fashion to achieve the complete test of the FPGA fabric as shown in Figure V. This method works due to the homogeneous architecture of each CLB. Each ORA compares an input from two independent BUTs. The BUTs are fed inputs from identical TPGs that are connected to alternating columns. This allows for a fault in the TPG to be detected when an entire column of BUTs fails. The BIST is run twice, reversing the assignments of the ORAs and BUTs. The BUTs are tested in both logic (SliceL) and memory (SliceM) modes, covering both possible utilizations of the LUT.

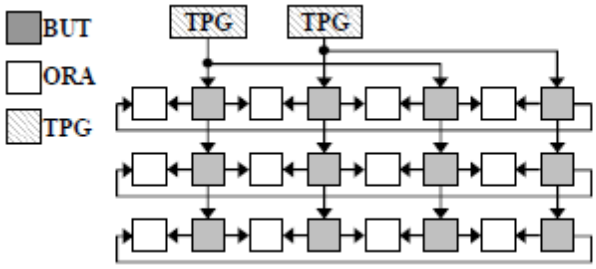

FIGURE V CIRCULAR BIST [3]

In SliceL mode the two competing BUT outputs are compared by the ORA. A mismatch is logged by the flip-flop in the ORA. Logic '1' is a passing value and logic '0' is a failure. The dedicated carry logic of the CLB is used to form an

iterative-OR of the ORA outputs. The first carry-in in the chain is connected to the to the Boundary Scan Test Data In (TDI) and the last to the Test Data Out (TDO). This eliminates the need to read the configuration memory for each ORA. Readback of the configuration memory is only needed if an error is detected on the TDO pin. In that case the ORA outputs can be read to determine where the error occurred.

Testing of the CLB in a Virtex-5 FPGA requires a minimum of six configurations, one to test each of the six inputs to the multiplexor. The six configurations also test the combinatorial logic output multiplexors using the first five configurations. The LUTs are tested using alternating XOR and XNOR functions. By using this pattern, all stuck-at faults in the LUT can be detected in two configurations. The configurations for the testing the Virtex-5 are shown in Table II.

TABLE II VIRTEX-5 TEST CONFIGRATIONS [3]

Config.#	A-D LUTs	FF/Latch	CYINIT	CLKINV
#1	XOR/XNOR	FF INIT1	#OFF	CLK
#2	XNOR/XOR	FF INIT0	AX	CLK
#3	XOR/XNOR	FF INIT0	0	CLK
#4	XNOR/XOR	LAT INIT1	1	CLK
#5	XOR/XNOR	FF INIT0	0	CLK
#6	XNOR/XOR	FF INIT1	AX	CLK_B
Config.#	**A-D FFMUX**		**A-D MUX**	
#1	O6, O6, O6, O6		CY, CY, CY, CY	
#2	O5, O5, O5, O5		XOR, XOR, XOR, XOR	
#3	AX, BX, CX, DX		O5, O5, O5, O5	
#4	XOR, XOR, XOR, XOR		O6, O6, O6, O6	
#5	CY, CY, CY, CY		F7, F8, F7, CY	
#6	F7, F8, F7, DX		F7, F8, F7, CY	

The TPGs are generated by utilizing the embedded DSP48E blocks inside the FPGA. They are configured to accumulate the prime number 0xCA6691 that is placed on their inputs. The number, as shown in [4], generates an exhaustive sequence of 12-bit test patterns in 2^{12} clock cycles. This requires 4,096 clock cycles in the Virtex-5 for an exhaustive test of the CLB.

In addition to standard SliceL configurations, the Virtex-5 also has SliceMs in CLBs in alternating columns. Only the bottom slice of a CLB is a SliceM. The other slice is a SliceL that can be used as an ORA for each test. There are also SliceM columns to the left of the DSP48E column. To test the alternating columns, a BUT from one column is compared by the ORA to another BUT in the same row in an adjacent column. The test pattern inputs to the BUTs are stored in a 2048x18 block RAM and are coordinated with an address counter in a DSP48E block to create a TPG. Each TPG drives alternating rows of BUTs. The RAMs formed in SliceMs can be single-port or dual-port memory. To test a single-port memory the TPG uses a March Y test pattern that requires 8N test patterns, where N is the number of address locations. The test configurations for all of the SliceM tests are shown in Table III.

TABLE III SLICEM TEST CONFIGURATIONS [3]

Config.#	RAM mode	DI1MUX	WEMUX	FFMUX
#1	SPRAM64	DX	CE	O6
#2	SPRAM32	A-DX	CE	O6
#3	DPRAM32	DX	WE	O5
#4	SRL32	MC31	WE	MC31
#5	SRL16	A-DX	WE	O6
Config.#	**OUTMUX**	**WA8used**	**WA7used**	**BIST CCs**
#1	O6	0	0	2,048
#2	O6	#OFF	#OFF	2,048
#3	O6	#OFF	#OFF	2,048
#4	O6	#OFF	#OFF	2,048
#5	MC31	#OFF	#OFF	2,048

Testing of the SliceL and SliceM configurations was performed using the LX30T and SX35T devices. Faults were injected using configuration memory bit faults. The BIST configurations were then executed with the fault programmed onto the device. Utilizing 17 BIST configurations, all faults were detected. In addition, three of the SliceM faults were detected using the SliceL configurations. The results of each test were readback using the TDO error detection as described above as well as the partial configuration memory readback when a fault was detected.

The configuration time of the BIST tests dominates the total test and readback time. For a LX30T using the 32-bit parallel interface, the configuration time for all 17 configurations totaled approximately six milliseconds. The execution and readback added another three milliseconds. The LX30T has a configuration size of 2,630 kB and a max BIST clock speed of 74.0MHz.

This test method of CLBs provides extensive coverage of the LUTs in both logic and memory mode. It also indirectly tests the DSP48E modules by using them as TPGs; however it cannot detect if the fault is the TPG or the row or column of BUTs being tested by the TPG. The total time to run the tests however is very large and grows linearly with the size of the FPGA due the dominance of the configuration time on the total time. For example, a LX110T that has a configuration size of 8,837 kB takes approximately 29 ms to test when using the 32-bit parallel interface test time. When using the boundary scan interface, the test time increases to almost 1.8 s. Running these tests, while exhaustive, do not cover interconnect or the embedded resources. They also take the FPGA offline for a very long period of time.

B. Interconnect Only BIST Techniques

Logic Only BIST does not test the interconnect wire-segments between logic blocks under test. This can lead to a false reading of a failure of a block when in fact it was the interconnect resource in between that was at fault. Programmable Interconnect Points (PIPs) typically account for 80% of the total configuration bits in a FPGA [16], therefore testing of them is critical to system reliability.

1) Cross-Coupled Parity BIST

The cross-coupled parity BIST approach proposed in [16] addresses the interconnect testing of a Xilinx Virtex-4 FPGA. The Virtex-4 utilizes a programmable routing network to connect CLBs, input/output (I/O) cells and embedded cores throughout the device. The network consists of horizontal and vertical wire segment along with PIPs to provide global and regional routing resources. The wires are named according the number of switch boxes they span, with double lines spanning two, hex lines spanning six and long lines spanning 24 [13]. Long lines are bi-directional, allowing sourcing from both ends, whereas double and hex lines are directional. Double and hex lines are also contain wire segment terminals named BEG (begin), MID (middle) and END to show their relationship to the source or destination. The Virtex-4 limits internal switchbox connections from hex lines to LUT inputs therefore MID and END terminals must be tested separately.

The cross-coupled BIST approach utilizes the Virtex-4 FPGA's slice architecture to create TPGs and ORAs. Each slice contains two flip-flops and two LUTs. A CLB consists of four slices. The first two slices are used for TPGs, one programmed with an odd parity, one with an even parity. Even parity TPGs contain a 2-bit down-counter that is initialized to all ones. Odd parity TPGs contain a 2-bit up-counter initialized to all zeros. The MSB from each counter provides the parity for each count value. The remaining two slices are programmed to be ORAs with each containing an even and odd parity checker. The architecture of the CLB is shown in Figure VI. Each TPG drives two ORAs and the values are validated. Testing can detect stuck-at faults as well as open faults.

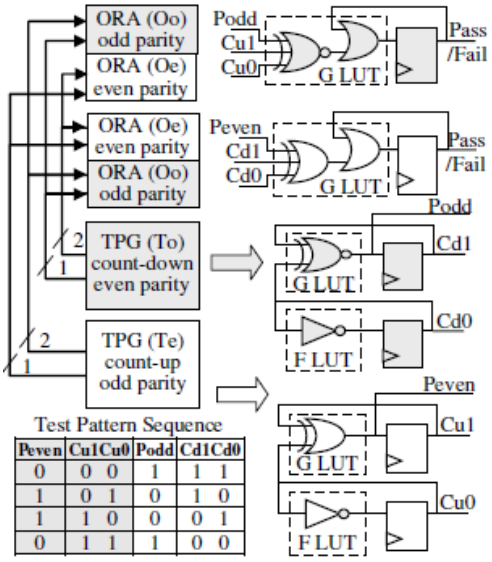

FIGURE VI NORTH HEX LINE BIST CONFIGURATION [16]

In the implementation presented, north hex lines are tested by configuring each of the column CLBs with Slice 2 and 3 as TPGs and Slice 0 and 1 as ORAs. MID terminals are tested by using Slice 1 ORAs and END terminals are tested using Slice 0 ORAs. A test pattern enters each switchbox at the BEG terminal and is propagated up to the next switchbox via the MID or END terminals. The parity is cross-coupled by using the switchboxes that drive each ORA. Each switchbox has ten hex line wire segments; however each BIST test can only test six. This requires four tests in each direction to cover each of the wire segments, with some tests overlapping lines. The overlap validates that all lines are tested with each adjacent line. South hex lines reverse the direction flow but are identical in implementation.

All columns within the Virtex-4 do not contain CLBs. Some contain embedded RAM, DSP slices, PLLs, etc. and therefore cannot be used as TPGs or ORAs. This structure creates a connection problem for testing east and west hex lines. To avoid this issue, the switchboxes in the non-CLB columns are used to pass test patterns between CLB columns. A similar problem occurs with north and south adjacent CLB columns are used to generate the TPGs and ORAs with the non-CLB switchboxes passing the test patterns through. The connections between the adjacent CLBs and non-CLB switchboxes are double lines. The testing of non-CLB north/south lines and CLB north/south lines must be done in different configurations due to contention for CLB resources.

Double lines are tested in the same way except that they require a test in each of the four directions. The increase in the directionality of tests is mitigated by the increase in the number of double line connections within a switchbox. This allows for the MID and END terminals to be tested simultaneously. The total number of test configurations needed is summarized in Table IV.

TABLE IV GLOBAL ROUTING BIST CONFIGURATIONS [16]

Routing Resource	Direction				Total Configs
	N	*S*	*E*	*W*	
CLB column double lines	2	2	2	2	8
Non-CLB column double lines	4	4			9 (1 extra)
CLB column hex lines	4	4	4	4	16
LX Non-CLB column hex lines	4	4			8
SX25/35 Non-CLB column hex lines	8	8			16
SX55 Non-CLB column hex lines	20	20			40
CLB column long lines	1	1	3	3	8
Non-CLB column long lines	1	1			2
Total BIST Configs	LX=51, SX25/35=59, SX55=83				

Long line tests differ from double and hex line tests because there are only five long line connections in to each switchbox. The implementation uses one of the long lines to source the test pattern and the remaining for to terminate the WUTs at ORAs. Instead of a 2-bit counter as used above, a 3-bit up and down counter is used. The additional bits require the two slices for each counter and one for each ORA.

North direction long lines use six adjacent CLBSs to send out test patterns. The CLBs are configured to alternate up and down counters to produce opposite parity on adjacent lines. This is used to detect bridging faults. East/West and non-CLB

column north/south lines are tested in a manner similar to the hex lines described above.

The results the BIST tests are stored in ORAs and are read back using partial reconfiguration memory. This allows not only the detection of faults but assists in the diagnosis of faulty wire segments. The ORAs can be divided into two groups covering the MID and END terminals. The TPG is covered by the BEG terminal. A failure indicated by both ORAs indicates a failure in the feedback routing of the TPG. A single failure indicates that the fault is in the WUT.

The experiments were generated by developing configurations in Xilinx Design Language (XDL) and converted to NCDs. Bitgen.exe is then used to convert the NCDs into downloadable configurations. A total of 51 configurations are needed to test a LX device (logic heavy), 59 tests for smaller SX25 and SX35 devices (DSP heavy) and 81 for SX55 devices. These tests indirectly test the CLBs connected to the logic by using them as TPGs and ORAs. It is assumed that they are golden elements in this off-line test and an error in a CLB cannot be distinguished from a wire segment error. No route-around solution for fault wire segments is presented.

2) Interconnect BIST (I-BIST)

A novel approach to interconnect testing is presented in [9] that utilize a ROving TEster (ROVE) to evaluate a portion of the FPGA while the rest of the device stays online. This method can provide up to 100% fault coverage of wire stuck-at, wire open, bridge and switch stuck-open/closed faults. Bridge faults are defined as wired-OR and wired-AND faults. I-BIST achieves high fault detection rate while minimizing the number configurations needed. In a nxn FPGA, only $2n$ configurations are required.

As described above, FPGAs contain wire-segments that are short and long. They can be classified as a length of one (single), two (double) or n (long) in a typical nxn FPGA. The wire segments run in both a horizontal and vertical orientation and therefore require two testers. This can be achieved by using a Horizontal-ROTE (H-ROTE) and a Vertical-ROTE (V-ROTE) [2]. Each V-ROTE area consists of three channels, or a set of tracks, each formed of multiple wire-segments and switches, that span either vertically or horizontally across the FPGA.

The I-BIST techniques main goals are to maximize diagnosability in the presence of multiple faults and to reduce test and configuration time. The goals are achieved by dividing testing into two phases, global and detailed. The global test phase is used to isolate faults to a small set of interconnects. That small set, known as suspects, is then tested independently to isolate the exact interconnect fault. Global testing utilizes V-ROTEs and H-ROTEs that span entire length and width of the FPGA. The nets that are formed within the ROTEs belong to the same track set and two adjacent nets act as a WUT pair. The pair is tested by the TPG using a 2-bit test vector and the outputs are compared in an ORA using a 2-bit XOR. Detailed testing uses a similar approach however the nets span a smaller section of the FPGA.

Bridge faults can be tested by applying a complimentary bit-vector to the two nets. Stuck-at faults require three test vectors, two complimentary and one identical, to detect en error on the nets. This is because a stuck-at fault on one wire will result in one of the two complimentary bit-vectors to match the other wire. Thus an identical bit-vector is used to detect this type of error. Stuck-open faults only affect the portion of the WUT that is between the TPG and the ORA. To handle this fault, two tests, one with the TPG at each end of the WUT are performed. A switch stuck-closed (SSC) fault cannot be detected with a switch on a net. To detect them a mirror-image net on different channels sharing a switchbox are used. The pair of WUTs in the mirror nets is tested using complimentary bit-vectors and a failure results in a bridge fault. The faults and tests are shown in Figure VII.

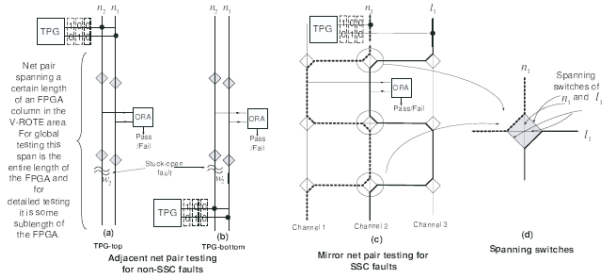

FIGURE VII FAULT CONFIGURATIONS FOR I-BIST [7]

Global testing utilizes simultaneous adjacent-&-mirror (SAMP) pair comparisons within a ROTE area. A single TPG is used to drive all adjacent and mirror net pairs and a separate ORA is used for each pair. A similarly configured set of nets in a channel is called a net-set. Fault detection is achieved by performing intra-net-set and inter-net-set comparisons. In intra-net-set comparisons, each pair of adjacent nets in a net-set is compared using a 2-bit XOR in the ORA. This configuration is used to detect non-SSC faults in the ROTE area. Inter-net-set comparisons use mirror net of two net sets to detect SSC faults. All of these tests are completed in five configurations.

Configuration 1 is used to detect stuck-closed faults. This is achieved by configuring the wire-segments in both the north/south and east/west sides of the switch to be contained on different nets. Configuration 2 is a mirror image of configuration 1. It is used to detect stuck-at faults in the south-west and north-east sides of a channel. Configurations 3-5 are used to detect stuck-closed faults in the adjacent channel switches. The nets in each of these configurations are extended so that they pass through each interconnect in at least one configuration. This allows sufficient coverage to detect all non-SSC faults.

Detailed testing is used to diagnose faults in the nets that global testing detects as faulty. To minimize test time, a divide-and-conquer (D&C) method is used. This is accomplished by dividing the ROTE area into two halves, applying the failed global testing configurations to the suspect track set in each half ROTE and repeating until the ROTE sub-area spans two rows. This is the minimum area required to apply the test configurations. Both intra- and inter-net-set testing is performed during this phase as long as a failure in the global testing configuration for adjacent track-sets resulted in a failure of the intra-net testing of the nets. A pass output would indicate that no bridge-fault or interconnect fault exists between these two nets, so only inter-net-set ORAs are used in detailed testing.

The I-BIST approach was simulated on a 32x32 FPGA with 16 tracks/channels. Each track was formed of single-length wire-segments. The entire design was simulated using C++. The results show that with a fault density (% of interconnect blocks that are faulty) of one to four percent, 100% of faults are detected. Fault densities of 5-10% resulted in fault coverage of 99.8% down to 99.3%.

C. Embedded Component Only BIST Techniques

Modern FPGAs consist of significant routing resources and logic blocks. They also may provide the designer with built-in, hard cores such as RAMs, Multiply-Accumulators, FIFOs and embedded processors. Embedded cores reduce the logic-resource requirements for complex designs and can decrease the power draw of the device. Testing of these blocks is needed to ensure a fully functioning device and to provide a reliable platform for long duration or critical missions.

Reference [7] introduces a group-testing based technique to perform BIST on these blocks with minimal overhead. The target devices for the technique are the Xilinx Virtex-5 FPGAs. These FPGAs have up to 1,056 embedded DSP48E cores and up to 18Mbits of embedded Block RAM [14]. The next generation Virtex-6 FPGAs have up to 2,016 DSP48E1 cores and up to 38Mbits of RAM. The embedded cores are organized in the device using a column-based, Application Specific Modular Block (ASMBL) architecture.

Testing of these blocks can be achieved by utilizing the adjacent CLB columns to implement TPGs and ORAs. Blocks such as 36-Kbit Block RAMs and DSP48E Multiply-Accumulators (MACs) can be targeted as BUTs; however the experiments demonstrated focus solely on the DSP48E MACs. It is assumed that the CLBs in the device have been tested and are functioning correctly.

The Combinatorial Group Testing (CGT) method begins with dividing the $\boldsymbol{m}$ embedded DSP cores into four groups of BUTs. Each group is then tested using non-adaptive, single-stage group testing. Each pair of BUTs is compared and a PASS/FAIL result is generated. A group of four BUTs, as shown in Figure VIII, requires six comparators to compare all BUT combinations.

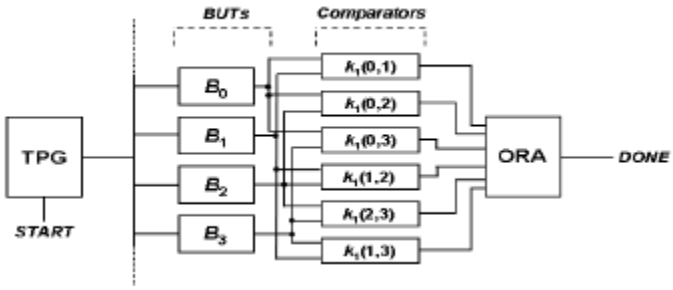

FIGURE VIII CGT BIST STRUCTURE FOR FOUR BUTS [7]

The control inputs to the DSP48E are 14-bits wide and include a 7-bit *opmode*, 3-bit *carryin-sel* and 4-bit *alumode*. A finite state machine (FSM) that generates the 400 test states and outputs the 14-bit control values is implemented using Block RAM configured as a 512x14 ROM. The addresses to the ROM are driven by a 9-bit adder. Data inputs to the DSP48E are generated using a 48-bit Linear Feedback Shift Register (LFSR), a 48-bit LFSR and a 30-bit LFSR. A total of six comparator blocks, one for each of the pairs of BUTs in a group, feed into each ORA. The comparators are made up of a 48-bit comparator, four 1-bit comparators both set to values of 0xFF to detect a mismatch and a 2x1 multiplexor to serialize the results. The test process is kicked off by a START signal to the TPG. The test is complete when the DONE signal is asserted and the results are retrieved.

A corner case of testing four BUTs in a group is that two compared blocks will have an identical failure. In these cases it is necessary to isolate the faults in each block so a well known signature can be used. The signature can be generated off-line via simulation. The value can be verified against the DSP output at the very last clock cycle of each test. Enhanced signature response analyzer methodologies can also add the ability to detect more than two defective blocks in a group.

The CGT method was tested in a Virtex-5 XC5VLX30 device that contains 32 embedded DSP cores. Since each BUT group contains four BUTs, there are a total of eight groups. The six multiplexers for each group, 48 overall, were optimized into six, 8-to-1 multiplexers by the synthesizer. Figure IX shows the structure for the XC5VLX30. Each multiplexor has one input from each of the BUT groups, 0-7, from one of the pairs of BUTs. The pairs are labeled $k_n(i,j)$ where n is the group number and $0 \leq i,j \leq 3, \forall i \neq j$. The six optimized multiplexers each feed a results flip-flop. A controller drives the 3-bit counter and the outputs of one group are placed into the flip-flops. The counter is then incremented until all eight groups have been tested. The fault diagnosis scripts isolate the faults to two of the four defective BUTs within a group.

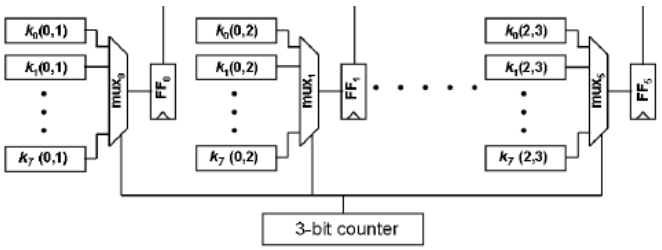

FIGURE IX BLOCK DIAGRAM OF SCHEME FOR TESTING THE XC5VLX30 [4]

The utilization in this test was 7% of LUTs and 2% of registers. Scaling of the BIST to a larger XC5VSX95T with 640 DSP48Es results in a utilization of 30% of LUTs and 2% of registers. The CGT BIST technique is an off-line method so the utilization of the FPGA is temporary and does not affect the size of the application configuration. The only additional overhead requirements are the memory to store the BIST configuration(s) and a controller to drive the test and evaluate the results. In some cases, the controller can be an embedded core within the FPGA; however it is subject to faults as well and must be assumed golden for the purposes of these tests.

D. Multi-Resource Capable BIST Techniques

The previous methods described in section III are only capable of directly testing one of three main components of modern FPGAs. In some cases, the other components of the FPGA are assumed to be golden and are used in testing of the interconnect logic or embedded blocks. This assumption is may not always be correct. All testing methods must assume some portion of the system is golden to perform accurate testing; however any component within the FPGA that is programmed using configuration memory is subject to a fault with equal probability. Therefore testing of more than one component type is needed to validate the FPGA's reliability. One method for achieving multi-component coverage is to combine two or more of the methods described above. Another method is to use Self Testing AReas (STARs) as proposed in [1].

The STARs method shown in Figure X is similar to [9] as it utilizes a Horizontal STAR (H-STAR) and Vertical STAR (V-STAR) to test the FPGA while moving application logic to another row/column during the test. This divides the FPGA into two types of regions, an operational area and a STAR. This method is pseudo-online in that it keeps application logic functioning during a BIST; however the region under test remains inaccessible to the application. The inclusion of two roving STARs reduces the test time of the FPGA and tests the north/south and east/west global long lines. Local interconnect is also tested in each STAR. A full sweep of the FPGA occurs when the V-STAR roves from left to right and the H-STAR roves from top to bottom. The process is then repeated in reverse.

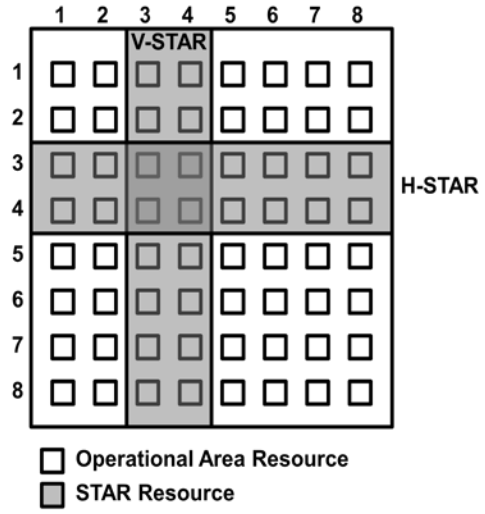

FIGURE X STAR AND OPERATIONAL AREA RESOURCES

The roving of the STARs is controlled by a Test and Reconfiguration Controller (TREC) located in a microprocessor. The TREC accesses the FPGA via boundary scan and partially reconfigures each column or row to move the STARs. The process for moving the STAR from its current position to the next area for test starts by the TREC copying the functions from next set of PLBs to the current STAR. In Figure X, the PLBs in columns 5 and 6 would be copied to columns 3 and 4. The system clock would then be stopped to allow the copying of any state information or RAM values to columns 3 and 4. Next the TREC reconfigures columns 3, 4, 5, 6 and to move the interconnect logic from 5 and 6 to 3 and 4. The system clock is then restarted and the tiles within the new STAR are configured for testing.

When the STAR divides the operational area into multiple regions, a performance penalty occurs for any interconnects that must traverse the STAR. The clock frequency of the initial system must contain enough slack to compensate for the maximum introduced delay. This allows the clock to be slowed as the delay increases.

The BIST structure, called a BISTER, consists of a TPG, an ORA with a flip-flop and two BUTs or WUTs. The TPG feeds either BUTs or WUTs the same test pattern so the ORA can use concurrent error detection (CED) to identify a fault. The pass/fail output of the ORA is then read out via boundary scan. The PLB functions are then rotated so that all logic and interconnects are tested as BUTs or WUTs. To reduce fault latency, fault detection and diagnosis are broken up into two distinct processes. The diagnosis process is not invoked until a fault has been detected in one of the STARs.

Logic fault detection is achieved by configuring the LUTs in each PLB for two different modes, combinatorial logic and RAM. The combinatorial logic testing requires the TPG, via a counter, to generate all 2^n configurations where n is the number of LUT inputs. The RAM configuration is tested by

using a state machine using standard exhaustive, RAM test sequences. In most cases the number of inputs of a PLB exceeds the number of outputs so more than one PLB must be used to generate a TPG. This is achieved by creating BISTER areas with enough PLBs to have two BUTs, one ORA and enough TPGs to generate exhaustive testing. This strategy allows each PLB to be a BUT twice resulting in guaranteed detection of single PLB permanent faults in at least two BISTER configurations. Any pair of faulty PLBs will also be detected in at least one configuration. All PLBs in a device will be tested after a single pass of the STAR.

Interconnect fault detection tests configuration interconnect points (CIPs) that consist of a transmission gate controlled by a memory bit. There are three types of CIPs that connect different configurations of wire segments. The first type is a crosspoint CIP that connects intersecting vertical and horizontal wire segments. The second type, a breakpoint CIP, connects two wire segments that are on the same plane. The last type is a multiplexer CIP that connects multiple crosspoint CIPs with a single output. The wire segments are also divided into two types, inter-PLB (global) and intra-PLB (local). These lines are tested for stuck-closed, stuck-open, stuck-at, open and shorted wire faults. A bridge fault (short) introduces a wired-AND or wired-OR function.

A typical WUT configuration is shown in Figure XI. The TPG is implemented as a counter and is connected to two chains of WUTs. PLBs can be used as identity functions (input directly connected to output) to validate intra-PLB wire segments. An exhaustive pattern of test vectors is sent through the WUTs and the ORA evaluates the correctness. A pass/fail result is stored in the ORA flip-flop for retrieval via boundary scan.

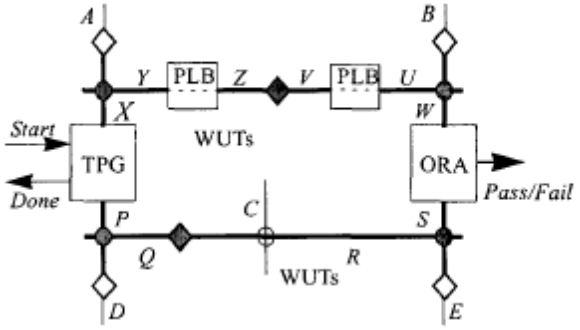

FIGURE XI TYPICAL WUT CONFIGURATION [1]

Crosspoint CIPs that connect global long lines may be used to traverse a STAR when the operational areas are disjoint. Complete testing of these lines in each round would require $O(N^2)$ configurations, so only a subset only requiring $O(N)$ configurations is used. The entire set of crosspoint CIPs is tested every N rounds.

Once a fault is found in a STAR, one of three methods of fault handling is utilized. The first method is to leave one of the STARs (vertical or horizontal) "parked" over the faulty area. This allows the TREC to diagnose the fault without impacting the rest of the online system. The TREC determines after additional testing whether or not the diagnosed fault will be articulated when the application logic in the next test area is moved to the current STAR. If it will not be articulated, the fault is logged and the roving method continues. If it will be articulated the TREC will generate a Fault-Bypassing Roving Configuration (FABRIC) to incrementally reroute around the fault. Some FABRICs can be generated at design-time; however the TREC must be capable of generating new configurations. The other STAR continues to rove the FPGA during this time. The next fault handling method is applying the generated FABRIC to release the parked STAR. If a FABRIC cannot be generated using spare resources, a third method called *STAR Stealing* is invoked. This process allows the operational area to occupy resources in the STAR to bypass the fault. This parks the STAR and disables it from further testing. It also eliminates the ability to test the routing resources in that STARs direction (vertical or horizontal). In some cases the other roving STAR may intersect the parked STAR, giving up its resources, and parking itself. The previously parked STAR can then begin roving the FPGA.

The STARs method requires a golden microprocessor to control and generate configurations. It also assumes the boundary scan interface and configuration memory used for feedback are error free. The authors tested STARs using an ORCA 2C series FPGA with a maximum configuration clock frequency of 10MHz. The device consists of a 20x20 logic array [6] that allows for 10 V-STAR and 10 H-STAR positions. A total of 15 configurations are required to test each PLB set and 17 configurations to test all of the CIPs. The total test time is 850 ms for the entire FPGA. A roving STAR requires approximately 5 ms to move between columns resulting in a worst case clock stoppage time of 6.25%. This time does not include diagnostic time. In [7] an equation for detection latency is presented for a $n\ x\ m$ logic array. C_{CLK} is the maximum configuration clock time provided by the FPGA. It is estimated that there are 33,500 effective cycles per PLB.

$$t_{latency} = \left(\frac{n}{2}\right)\left(n * \frac{33{,}500}{C_{CLK}}\right) + \left(\frac{m}{2}\right)\left(m * \frac{33{,}500}{C_{CLK}}\right) \quad (1)$$

The equation for a Virtex-4 XC4VLX200 consisting of a 116x192 PLB array is also presented. The maximum boundary scan frequency for this device is 50MHz. The resulting detection latency for this device is 17s.

IV. EVALUATION OF BIST TECHNIQUES

This section compares the various presented techniques against the metrics listed in Table I. All of the methods utilized different FPGA target devices. To compare them effectively, estimates will be normalized to a Virtex-4 FPGA if possible. In all other cases, the presented target device will be used.

TABLE V METRIC RESULTS

Metric	Sub-Metric	Hard Macro	CLB	Cross-Coupled Parity	I-BIST	CGT	STARS
Fault Type	**Distinguish Transients?**	Not Addressed	Not Addressed	Not Addressed	Yes, two phase testing	Not Addressed	Yes, multi-pass
Coverage Capability	**Logic Blocks**	Yes, 100%	Yes, 100%	No	No	No	Yes
	Interconnects	No	No	Yes	Yes, 100% to 99.3% for fault densities of 1-10%	No	Yes
	Embedded Cores	No	No	No	No	Yes	No
Overhead Requirements		Processor, Configuration storage memory, time	Processor, Configuration storage memory, time	Processor, Configuration storage memory, time	Spare column(s)/row(s), processor, configuration storage memory, time	Up to 30% LUTs, 2% FFs; processor, configuration storage memory, time	Spare columns/rows, processor, configuration storage memory, time
Evaluation Time	**Latency of Detection**	219 µs + Download Time (Virtex-4 XC4VLX200)	Boundary Scan – ~1.8s; Parallel Interface - ~29ms (Virtex-5 LX110T)	Time not addressed; Target device Virtex-4 LX	Simulated 32x32, no intrinsic implementation	Not specifically addressed; one-shot, concurrent test dependent on clock speed	17s (Virtex-4 XC4LX200)
	Number of Configurations	42	17 for Virtex-5; 25 for Virtex-4	51	5 per ROTE area	1	Dependent on Array Size
Golden Element(s)		Processor, Configuration Memory, Interconnect, Boundary Scan Chain	Processor, Configuration Memory, Interconnect, Readback Interface	CLBs, Processor, Configuration Memory, Readback Interface	PLBs, Processor, Configuration Memory, Readback Interface	CLBs, Processor, Configuration Memory, Readback Interface	Processor, Configuration Memory, Readback Interface

Four metrics are used to evaluate the BIST techniques. The first metric is fault type. Techniques should be capable of distinguishing between a transient and a permanent fault. The second metric is coverage capability. This metric is broken into three component types: logic blocks, interconnects and embedded cores. Techniques are evaluated against their ability to detect a fault in each one of the three component types. The third metric is overhead requirements. All BIST techniques require a controller to start, evaluate and end each test. Additional requirements, such as storage memory, used logic, and required spares are evaluated. The final metric is evaluation time. This is defined as the latency of detection of each technique and the number required configurations. A fifth row in the table is added to list the golden elements required for each technique. The results are displayed in Table V.

A. Fault Type Summary

Two of the six evaluated papers addressed, directly or indirectly, the ability to distinguish transient faults from permanent faults. I-BIST and STARs methods are very similar in architecture and utilize roving test blocks. The I-BIST method cannot explicitly detect a transient fault however it utilizes two phase testing. If the transient fault occurs during the global testing phase and no longer articulates during the detailed test phase the block will be declared functional. If the transient fault occurs during the detailed testing phase within the ROVE it will also be detected. STARs operates in a similar manner.

B. Coverage Capability

The presented papers each had a single coverage capability except for STARs. Limiting of the test space to a single component type allows for a simpler testing methodology and less reconfiguration. It also limits the granularity of which a fault can be detected. For example, the CLB testing approaches presented in the Hard Macro and CLB papers cannot distinguish between a LUT fault and a fault in the internal routing logic of the CLB. A detected fault is always attributed to the LUT. A similar example can be shown for both the interconnect only and embedded core components. The STARs method tests both interconnect resources as well as logic resources. It is capable of narrowing the fault to either

of the components, enhancing the ability to reuse partially faulty components.

C. Overhead Requirements

All of the methods require an external processor, memory for storing configurations and time. These resources, along with the readback interface, are always assumed to be golden elements. Most of these methods are offline so their main resource requirement is configuration memory space.

The I-BIST and STARs approaches also require spare row(s) and column(s) for moving logic and interconnects during testing. The CGT method for embedded core testing requires up to 30% of LUTs and 2% of flip-flops depending on the size of the FPGA.

D. Evaluation Time

Evaluation time can be broken into fault detection latency and the number of configurations required. A Virtex-4 XC4VLX200 was used in two of the six papers and is used here for comparisons.

1) Fault Detection Latency

Latency was addressed by three of the six papers presented. The CGT approach runs concurrently and therefore is dependent on the clock frequency.

The Hard Macro approach requires 219 µs plus the total download time. The download time is approximately 32 ms at 50MHz for an entire configuration over JTAG. There are 42 configurations required for the Hard Macro approach so the total time is $(32\,ms + 219\,\mu s) * 42 = 1.35s$. Partial reconfiguration is most likely used in this method so this time is a worst case. The other logic testing method requires approximately 1.8 s over the same interface; however this method targets a Virtex-5 FPGA containing significantly more configuration bits. Both methods are offline and their detection latencies do not negatively affect the application outputs. If the BIST is run only at startup or infrequently during operation, the actual detection latency approaches infinity. To create a truly reliable system, the device must be taken offline frequently enough to account for the probability of fault occurrence.

STARs require 17s to scan the same size Virtex-4 FPGA. This latency, although significant, occurs while the rest of the system is online. A fault occurring in a component that is currently not under test will not be detected until it is part of the STAR. If the fault is a transient it may not be detected at all due to the long latency. This results the possibility of a significant number of erroneous outputs while the roving tester scans the entire FPGA. Compared to a one-time BIST approach, the overall latency is manageable.

2) Number of Required Configurations

In addition to fault detection latency, it is important to note the number of required configurations required to complete a test. The total number of required configurations affects the configuration storage memory requirement as well as the scalability of the BIST approach. The best approach for scalability is the CGT method. It requires only a single configuration to complete its test. The next best method is the interconnect only test method proposed in the I-BIST paper. Five configurations per ROTE are required for a total of 10 configurations, one for the vertical and one for the horizontal, assuming homogeneous array architecture. The CLB test approach for logic requires 25 configurations for the Virtex-4 and 17 for the Virtex-5, showing newer, larger and more modern FPGAs do not necessarily increase the required number of configurations. The last two approaches, Hard Macro and Cross-Coupled Parity require 42 and 51 configurations respectively. This is a significant number assuming, in the worst case, a total reconfiguration is required.

For total fault coverage, combining of methods may be required. STARs are the only multiple-component fault detection method presented in this paper however it does not include an estimate on the number of configurations for a Virtex-4. The next best combination would be I-BIST for interconnect and CLB for logic. Additionally, FPGAs with embedded resources would require the CGT approach. This results in a total number of reconfigurations, assuming they are executed sequentially, of 36.

E. Golden Elements

As discussed in Section IV.C, all of the BIST methods require a processor, configuration memory and a readback (parallel or boundary scan) interface to function. These elements are assumed to be golden, or fault free, for all tests. In certain tests, additional sub-components are assumed to be fault free.

The Hard Macro and CLB tests assume all interconnect logic is fault free and that any fault detected is associated with the logic in the BUT. The interconnect and CGT approaches utilize CLBs to create TPGs and ORAs and are therefore assumed to be golden. STARs tests both interconnect and logic so no additional golden elements are required.

V. CONCLUSION

This paper provides an overview and comparison of various FPGA BIST techniques. All of the methods provide successful fault detection for permanent faults in their target components. Transient fault detection is either not addressed or only detected through multiple passes of the BIST. No one method can cover all three component types so a combination of the best techniques must be used to provide complete fault coverage of an entire FPGA.

Selecting a BIST method requires balancing system offline time, overhead requirements such as processor speed and configuration memory, and fault detection latency. All of the methods except STARs and I-BIST utilize the FPGA in a completely offline state with test times ranging from 1.35s to 1.8s for a Virtex-4 XC4VLX200 FPGA. The other two methods use the FPGA in a partial online state where the current rows/columns under test are effectively offline and their previous logic or wire-segments are moved to adjacent components. STARs requires a 17s latency time to scan the same size FPGA however the test can be run continuously. These values must be weighed against the probability of fault occurrence to evaluate their effectiveness.

Further research potential exists in the testing of embedded cores in modern FPGAs alongside the logic blocks and interconnects that are used to test them. A well rounded, full FPGA test suite that can be run in a reasonable amount of system time may be difficult to achieve, but a set of test suites capable of executing multiple simultaneous component type tests may be realized.

REFERENCES

[1] M. Abramovici, J. M. Emmert, and C. E. Stroud, "Roving STARs: an integrated approach to on-line testing, diagnosis, and fault tolerance for FPGAs in adaptive computing systems," in *Evolvable Hardware, 2001. Proceedings. The Third NASA/DoD Workshop on*, 2001, pp. 73-92.

[2] M. Abramovici, C. Strond, C. Hamilton, S. Wijesuriya, and V. Verma, "Using roving STARs for on-line testing and diagnosis of FPGAs in fault-tolerant applications," in *Test Conference, 1999. Proceedings. International*, 1999, pp. 973-982.

[3] B. F. Dutton and C. E. Stroud, "Built-In Self-Test of configurable logic blocks in Virtex-5 FPGAs," in *System Theory, 2009. SSST 2009. 41st Southeastern Symposium on*, 2009, pp. 230-234.

[4] S. Gupta, J. Rajski, and J. Tyszer, "Test Pattern Generation Based On Arithmetic Operations," in *Computer-Aided Design, 1994., IEEE/ACM International Conference on*, 1994, pp. 117-124.

[5] D. S. Katz and R. R. Some, "NASA advances robotic space exploration," *Computer*, vol. 36, pp. 52-61, 2003.

[6] Lattice Semiconductor, "ORCA Series 2 Field-Programmable Gate Arrays," November 2006.

[7] M.G. Parris, C.A. Sha rma, and R. F. DeMara, "Progress in Autonomous Fault Recovery of Field Programmable Gate Arrays," accepted to ACM Computing Surveys, December 27, 2009

[8] A. Sarvi, C. A. Sharma, R. F. DeMara, "BIST-Based Group Testing For Diagnosis of Embedded FPGA Cores," in Proceedings of the *International Conference on Embedded Systems and Applications (ESA'08)*, Las Vegas, Nevada, U.S.A., July 14-17, 2008.

[9] V. Suthar and S. Dutt, "Efficient On-line Interconnect Testing in FPGAs with Provable Detectability for Multiple Faults," in *Design, Automation and Test in Europe, 2006. DATE '06. Proceedings*, 2006, pp. 1-6.

[10] S. Trimberger, "A reprogrammable gate array and applications," *Proceedings of the IEEE*, vol. 81, pp. 1030-1041, 1993.

[11] S. Vigander, "Evolutionary Fault Repair of Electronics in Space Applications," in *Department of Computer and Information Science* Trondheim, Norway: Norwegian University of Science and Technology (NTNU), February 28, 2001, p. 50.

[12] Xilinx, "Virtex 2.5V Field Programmable Gate Arrays DS003-1, v2.5," April 2, 2001.

[13] Xilinx, "Virtex-4 FPGA Users Guide v2.6," December 1, 2008.

[14] Xilinx, "Virtex-5 Family Overview - LX, LXT, and SXT Platforms," December 2007.

[15] Xilinx, "Virtex-6 Family Overview DS150," January 28, 2010.

[16] L-T Wang, C. Stroud, and N. Touba, System-on-Chip Test Architectures, Morgan Kaufmann, 2007.

[17] Y. Jia, B. Dixon, C. Stroud, and V. Nelson, "System-level Built-In Self-Test of global routing resources in Virtex-4 FPGAs," in *System Theory, 2009. SSST 2009. 41st Southeastern Symposium on*, 2009, pp. 29-32.

[18] Z. Zhiquan, W. Zhiping, C. Lei, Z. Fan, and Z. Tao, "A novel BIST approach for testing logic resources using Hard Macro," in *Neural Networks and Signal Processing, 2008 International Conference on*, 2008, pp. 379-381.