



System-Level BIST for Programmable I/O Cells in FPGAs and SoCs

Lee W. Lerner, Sudheer Vemula, and Charles E. Stroud

Dept. of Electrical and Computer Engineering
200 Broun Hall, Auburn University, AL 36849-5201
emails: lernelw/vemulsu/strouce@auburn.edu

ABSTRACT: A Built-In Self-Test (BIST) approach is presented for system-level testing of the programmable Input/Output (I/O) buffers in Field Programmable Gate Arrays (FPGAs) and configurable System-on-Chip (SoC). We discuss implementation methods for the BIST approach, including parameterized VHDL and FPGA-specific hardware design description languages. The fault detection capabilities and limitations of the BIST approach are presented along with the results of the application of the various implementation methods to several commercially available FPGAs and SoCs.¹

I. INTRODUCTION AND BACKGROUND

The programmable Input/Output (I/O) cells in Field Programmable Gate Arrays (FPGAs), as well as the programmable I/O cells associated with FPGA cores in configurable System-on-Chip (SoC) devices, consist of bi-directional buffers and programmable logic including flip-flops/latches and multiplexers as illustrated in Figure 1. These I/O cells have a number of programmable features which facilitate not only user control and specification of the input, output, and bi-directional ports for any given system application, but also critical system attributes such as timing, voltage standards, and drive capabilities. As new FPGA architectures are developed, the amount of programmable logic resources in I/O cells tends to be increasing to support system capabilities such as high speed data transfer. If the I/O cells are faulty, then the information exchange between other components in the system may not be possible and, as a result, the I/O cells must be included in tests for fault free operation of any device.

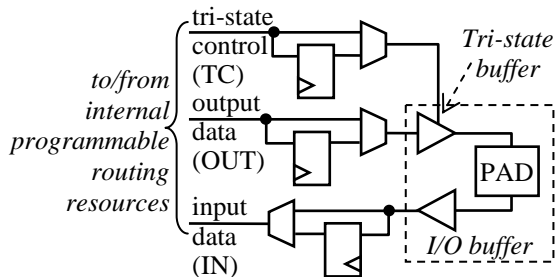


Fig. 1. Example programmable I/O cell architecture

Boundary scan, and more specifically the EXTEST feature, has traditionally been used to test input and output buffers, tri-state control, and pads along with the interconnections between devices on a printed circuit board [1]. However, the EXTEST capability can only test the I/O buffer in the programmable I/O cell of an FPGA. The EXTEST circuitry cannot test the programmable logic in the I/O cells, such as the flip-flops used for registered inputs and outputs, and cannot test the programmable routing resources connecting I/O cells to the FPGA core. While the boundary scan INTEST feature could be used to test the programmable logic and interconnect resources, very few FPGAs provide the INTEST capability since it is not a required feature of the IEEE boundary scan standard [1]. As a result, testing the programmable I/O cells in FPGAs and configurable SoCs has been overlooked for system-level testing.

There have been a number of BIST approaches developed for the programmable logic [2][3] and routing resources [4][5][6] in FPGAs and FPGA cores in SoCs [7][8]. However, none of these BIST approaches have targeted the programmable logic and routing resources associated with the I/O cells. Yet the programmable logic and routing resources in the I/O cells for unbonded pads (common in many FPGA packaging options) are frequently used by the FPGA synthesis tools to implement the system function. Therefore, the I/O cells must be tested as well.

BIST has been used to test the set-up and hold time of the registers associated with I/O buffers in Application Specific Integrated Circuits (ASICs) [9], where additional circuitry, such as delay locked loop, test registers and comparators, was required for each buffer under test. As a result, this particular BIST approach is not directly applicable to FPGAs. An I_{DDQ} testing approach for FPGA I/O buffers was proposed where the test signals were generated both internally and externally, but the results were analyzed only by externally monitoring of the signal values [10]. As a result, this technique is not a BIST approach since internal logic resources were used only for generation of test patterns to the I/O buffers to overcome limited external access, particularly in the case of unbonded pads.

A BIST approach has been developed for programmable I/O cells and their associated routing resources [11]. While applicable to any FPGA or the programmable I/O

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

cells associated with FPGA cores in a configurable SoC [12], the primary target of the BIST approach was manufacturing testing and not system-level testing. Therefore, the goal of this paper is to extend that BIST approach to system-level testing as well as to discuss a number of issues associated with system-level BIST for programmable I/O cells.

Section II describes BIST architecture and introduces issues associated with its system-level application. Section III describes approaches to implementing the BIST architecture including parameterized VHDL and FPGA-specific hardware design description languages. Section III also discusses the use of FPGA dynamic partial reconfiguration via embedded processors. Section IV presents experimental results to illustrate the effects of different loading on system-level use of the BIST approach as well as the fault detection capabilities and physical implementation in various devices. The devices primarily include the Atmel AT94K series configurable SoC [13] and the Xilinx Virtex-4 series FPGA [14] which are representative of the two ends of the I/O cell complexity spectrum. These two devices are summarized in Table I in terms of various features and attributes that will be discussed throughout the paper. The paper concludes in Section V with a summary of the capabilities and limitations of this I/O cell BIST approach.

Table I. Example I/O Cell Features

Feature	Atmel AT94K	Xilinx Virtex-4
# Multiplexers	4	32
# Flip-flop/latches	2	10
Flip-flop/latch	flip-flop only	both
Set/reset	asynchronous reset	programmable
Clock enable	no	programmable
Pull-up/pull-down	yes	yes plus keeper
Output drive levels	3	7
I/O standards	3	69
Delay options	4	64

II. BIST ARCHITECTURE

The basic concept of the I/O cell BIST architecture, illustrated in Figure 2, is to configure the I/O cells as bidirectional buffers to allow test patterns to be applied to output portion of the I/O cell while providing a return path through the input portion of the I/O cell. The output response of each I/O cell under test is then compared to responses of two other identically configured I/O cells by output response analyzers (ORAs), implemented in the programmable logic resources of the FPGA core. Fig. 3 illustrates the basic ORA design where any mismatches in one or more output responses that result from faults in the I/O cells are latched by the flip-flop with feedback to the OR gate. The I/O cells

under test and the ORAs are arranged to provide a circular comparison such that every cell is observed by two ORAs and compared with two different cells under test. This circular comparison provides improved diagnostic resolution when identifying faulty resources based on the failing BIST results using a relatively simple diagnostic procedure [8]. To retrieve the BIST results, a multiplexer is incorporated in the ORA design as illustrated in Figure 3 so that the ORAs can be connected together to form a shift register, or scan chain, similar to that used in boundary scan. At the end of the BIST sequence, the shift mode control signal is activated and the contents of the ORAs are shifted out in shift mode of operation [2]. Alternatively, the contents of the ORAs can be retrieved via partial configuration memory readback in FPGAs that support this readback capability.

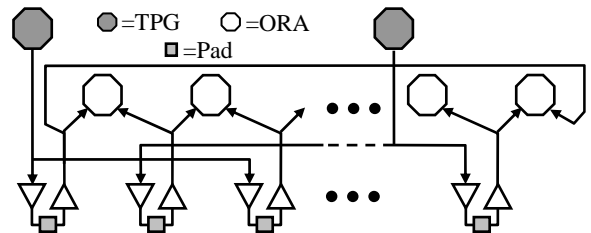


Fig 2. I/O buffer BIST architecture

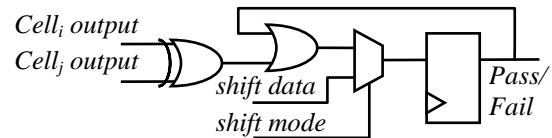


Fig. 3. ORA design

Multiple identically configured test pattern generators (TPGs) are also implemented in the programmable logic resources of the FPGA core and produce test patterns to alternating I/O cells under test. The use of multiple TPGs greatly reduces the probability that a fault in the TPG could skip test patterns that would otherwise detect faults in the resources under test [2]. With multiple TPGs driving alternate I/O cells under test, a faulty TPG would result in mismatches in all ORAs monitoring I/O cells being driven by the faulty TPG, as can be seen in Figure 2. The original BIST approach for I/O cells [11][12] implemented only one TPG under the assumption that the FPGA core had been previously tested and found to be fault-free. While this is a safe assumption for manufacturing testing, it is not a good assumption for system-level testing where the number of configurations required to completely test the logic and routing resources of the FPGA core may be prohibitively large for system-level application. Therefore, for system-level BIST we implement multiple TPGs to remove the fault-free assumption on the FPGA core.

The I/O cells are repeatedly reconfigured in all modes of operation (pull-up, pull-down, flip-flops, slew rate, etc.) that can be practically configured within the constraints of the system-level BIST approach and implementation approach as will be discussed in the next section. The number of times the I/O cells must be reconfigured is a function of the I/O cell architecture and its programmable features. For example, the Atmel AT94K40 requires a total of 303 configurations to completely test all of the I/O cells [12], yet only 64 configurations are required to completely test all of the programmable logic (8 configurations) and routing (56 configurations) resources in the FPGA core of the same device [7]. Since the download time typically dominates the total testing time for an FPGA [2], the use of dynamic partial reconfiguration is critical for effective system-level testing. The most efficient approach is to perform dynamic partial reconfiguration via an embedded processor core with access to the configuration memory [7].

During manufacturing testing, it is possible to test all I/O cells concurrently depending on the architecture. For example, in the Xilinx Virtex-4, all programmable I/O cells are identical and can be tested at the same time. However, in the Atmel AT94K, there are two types of I/O cells, primary and secondary, with different logic and routing resources. The two types of I/O cells must be tested separately with one set of 13 configurations for the primary I/O cells and another set of 10 configurations for the secondary I/O cells [12]. Furthermore, it can be safely assumed that during manufacturing testing the pins of the package are not connected to external loads or, in the case of a test machine, that the loading is identical on all pins. In system-level BIST, however, the FPGA has already been assembled on a printed circuit board with some of the I/O cells targeted to be configured as input buffers, some as output buffers, and some as bi-directional buffers. If some of the signal sources for inputs to the FPGA cannot be tri-stated, then those I/O cells cannot be tested during the system-level BIST since they cannot be configured as bi-directional buffers during BIST. As a result, only a subset of I/O cells can be tested at the same time.

Even if all signal sources can be tri-stated during the system-level I/O cell BIST, it may still not be possible to test all I/O cells concurrently due to the fact that the I/O cells will have different loads present on them. This means that the charge and discharge times of I/O cells configured as bi-directional buffer are affected by these different loads. As a result, the maximum clock frequency at which the BIST circuitry will vary for different sets of I/O cells under test. Either all of the I/O cells must be tested at a single and considerably slow clock frequency, or the I/O cells must be grouped together by load characteristics to be tested at different times and at

different clock frequencies. Consequently, it is evident that BIST approaches developed for manufacturing testing, in which all I/O buffers are tested simultaneously, cannot be adequately applied I/O buffer testing at the system level. The effects of testing I/O cells with different loads will be discussed in more detail in Section IV where the sensitivity of the BIST approach will be shown for actual experimental result.

The BIST approaches developed for the programmable logic [2][3], routing [4][5][6], and embedded cores [8] in an FPGA can be used to test those FPGAs for during both manufacturing and system-level testing. Since not all I/O cells can be tested concurrently, the system-level BIST implementation must be completely parameterized so that a user can specify the location and the number of I/O cells to be tested concurrently. The system-level BIST implementation also becomes package specific when the unbonded buffers are included in the test.

III. BIST IMPLEMENTATION

We investigated two different approaches for implementing system-level BIST for programmable I/O cells. The first approach consists of BIST implementation through the use of parameterized VHDL. The second approach relies on FPGA vendor-specific hardware design description languages (HDDLs), such as Macro Generation Language (MGL) for Atmel FPGAs [15][16] and Xilinx Design Language (XDL) for Xilinx FPGAs [17]. The advantages and disadvantages of both of these approaches are discussed in the subsequent subsections.

A. VHDL BIST Generation

A parameterized VHDL model has been successfully used for the development and implementation of BIST configurations for embedded cores in FPGAs [8]. The advantage of this approach is that it can easily be applied to almost any FPGA with little modification to the VHDL. Furthermore, it can easily be applied to any subset of cores within a given FPGA as a result of the parameterization. Therefore, this appeared to be an attractive approach for system-level implementation of the I/O cell BIST approach. The parameterized model facilitates relatively easy and straightforward specification of the number of I/O cells to be configured as bi-directional buffers in a given BIST configuration for the system application. Figure 4 illustrates a VHDL instantiation of I/O buffers in an Atmel AT94K series SoC where a GENERATE statement is used to parameterize the number of I/O cells to be tested in a given BIST configuration. In this example, the GENERATE statement instantiates N bi-directional buffers by calling the BIBUF macro. The BIBUF macro used in this example is specific to the Atmel AT40K and AT94K macro library [15].

```

BUFFs: for I in 0 to N-1 generate
  B_I: bibuf port map (A => TPG(0),
    OE => TPG(1), Q => Bdata(I),
    PAD => PAD(I));
end generate BUFFs;

```

Fig. 4. Parameterized VHDL I/O buffer instantiation

The VHDL model can be applied to any FPGA which supports a similar macro to instantiate bi-directional I/O buffers. For example, this same code can be used for Xilinx FPGAs (excluding Virtex-4) by changing the BIBUF macro to the IOBUF macro supported by Xilinx synthesis tools. In the case of Virtex-4 FPGAs, a single macro to instantiate I/O buffers as bi-directional buffers is not supported since the complete I/O cell consists of three separate components including a pad (PAD), input logic (ILOGIC), and output logic (OLOGIC).

Once the parameter N has been specified, the VHDL model is synthesized in conjunction with the use of a constraints file or a physical constraints editor to specify the physical locations of specific I/O cells to be tested. Constraint files can be generated via a higher-level programming language, such as C or C++, or with a constraints file editor. Alternatively, specific VHDL synthesis attributes can be included directly in the VHDL. Graphical physical constraints editor software, such as the Figaro place and route tool in Atmel's System Designer tool suite or the PACE editor in the Xilinx ISE tool suite, can also be used to specify the physical location of the specific I/O cells to be tested. Both tools allow a user to visually locate and place modules synthesized from VHDL onto FPGA components including the I/O cells. It was at this point that we encountered a major problem with the VHDL-based approach. FPGA synthesis tools do not permit access to unbonded I/O pads and, as a result, the VHDL-based BIST approach is limited to testing only the bonded pads of an FPGA. Yet, the logic and routing resources associated with unbonded pads are frequently used by the synthesis tools for implementing the system function and, as a result, should be tested.

The I/O cells contain flip-flops that must be activated to be tested. Another shortcoming of the VHDL approach is that vendor-specific macros are required that activate the specific elements to be tested, such as flip-flops. Otherwise, the clock and TPGs signals for testing set/reset, clock enable, etc. are not connected to the I/O cell. Figure 5 provides an example of how the VHDL can be synthesized to an I/O buffer in the absence of a vendor-specific macro that includes the flip-flops in the I/O cell. Note that the BIBUF macro in Figure 4 is only a bi-directional buffer and does not have connections to the flip-flops. As a result, the flip-flop operation must be included in the behavioral description of the VHDL.

The synthesis tools must then recognize the flip-flops, include them in the synthesized I/O cell, and make the necessary connections (i.e., clock, clock enable, set/reset, etc.), a process we refer to as *activation*. In this example, the flip-flop in the output portion of the I/O cell was not activated while the flip-flop in the input portion of the I/O cell was activated. The flip-flop for the output portion of the I/O cell was instead placed in the FPGA core. Note that only one input from the TPG is connected to the multiplexer. As a result, this particular I/O cell will not be completely tested. This example has been observed frequently in practice by synthesizing designs on different vendor's FPGAs.

In summary, there are two major problems with the VHDL-based approach. First, we cannot test I/O cells with unbonded I/O pads even though their resources may be used to implement the system function. Second, and perhaps more important, we are limited to guaranteeing the testing only of those resources activated by the vendor-specific VHDL macros. The problem is that majority of the vendor-specific macros only activate the bi-directional I/O buffer which limits the resources that are tested to the same resources that can be tested by boundary scan EXTEST. Therefore, VHDL is not a viable approach for implementing system-level BIST to completely test I/O cells.

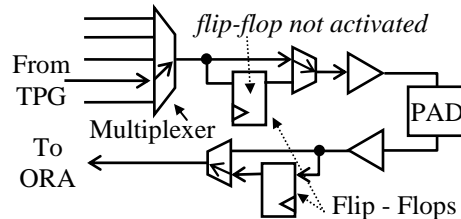


Fig. 5. Bi-directional I/O cell from VHDL

B. FPGA Vendor-Specific HDDL BIST Generation

FPGA manufacturers typically provide some type of HDDL, such as MGL or XDL, to implement high performance custom designs on their product family of FPGA. Because MGL and XDL are architecture specific languages, they provide more control over the placement and routing of the design than VHDL. Furthermore, the enhanced control of the FPGA resources helps to improve the fault coverage of the BIST approach. We used MGL and XDL to implement the BIST approach for Atmel AT94K and Xilinx Virtex-4 devices, respectively.

MGL provides users of the AT40K FPGAs and AT94K FPGA cores with an integrated method for creating parameterized, user-defined circuits to meet desired design specifications [16]. MGL is similar to a programming language and supports basic constructs similar to most programming or hardware description languages, such as VHDL. In order to instantiate a design using MGL in

a particular FPGA array, three components must exist in the MGL code: user-defined functions (i.e., ORAs, TPGs), the target FPGA device (i.e., AT94Kxx), and the inputs and outputs to the circuit (i.e., clock input, reset input, pass/fail output). Pre-processor directives, global variables and constants may also be defined.

XDL has a structural netlist format where components (i.e., programmable logic blocks, I/O cells) are instantiated and interconnected [17]. In addition to defining the configuration of each component in terms of its programmable resources, the location of the component in the array can also be specified. Routing resources used to interconnect components can be specified in terms of the actual wire segments and programmable switches used to route a signal or the source and destination(s) for a given signal can be specified with routing tools used to perform the final routing. XDL is good for very regular designs and has been successfully used for many years to implement BIST configurations for logic and routing resources in the core of Xilinx FPGAs [3][17]. In these cases, a program written in C or C++ is used to algorithmically generate an XDL file for the BIST implementation. This XDL generation program is typically parameterized to support any size FPGA in a given family [17].

HDDL implementations for the system-level I/O cell BIST approach overcome the two problems encountered with VHDL. The HDDLs typically have much more control over the placement and utilization of the resources of an FPGA allowing unbonded I/O cells to be tested and allowing resources such as flip-flops in I/O cells to be easily activated as the BIST requires. This can be seen in Figure 6 where both flip-flops are activated and, as a result, their associated clocks, set/resets, and clock enables will be connected and routed. Also note that TPG signals are routed to all inputs of the multiplexer at the input to the output portion of the I/O cell. This is because most vendor-specific HDDLs provide the capability to route partially connected nets, such as routing from the TPG outputs to the unselected inputs of the multiplexer shown in Figure 6. As a result, the I/O cell can be completely tested (or almost completely tested) through the use of dynamic partial reconfiguration as will be discussed in the next subsection.

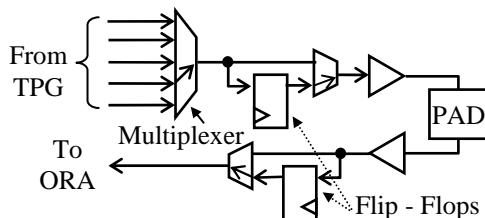


Fig. 6. Bi-directional I/O cell from HDDL

The primary disadvantage of using vendor-specific HDDLs is the considerable time required both in terms

of learning the language and in developing the BIST configurations for each type of resource under test and for each family of FPGAs. The amount of time required to develop parameterized BIST using vendor-specific HDDL is considerably high compared to the amount of time required to develop parameterized BIST in VHDL. This difference in development time is due primarily to the fact that the parameterized MGL program is much larger than the parameterized VHDL model. Similarly, the XDL has to be generated using a high level programming language, like C or C++, in order to maintain its flexibility in specifying the number and location of I/O cells to be tested. To give an idea of the difference, the number of non-commented lines of source code for each approach is summarized in Table II, where the entry for XDL is the number of lines of code from the C program that generates the XDL description.

Table II. Non-Commented Lines of Source Code

VHDL	MGL	XDL
74 lines	1,495 lines	1,519 lines

C. FPGA Partial Reconfiguration

Many recent FPGAs either have embedded processors or support synthesis of soft processor cores which can be used to write the FPGA configuration memory and, in some cases, read the configuration memory. The use of embedded processors in FPGAs to perform partial reconfiguration significantly reduces total test time [7]. During partial reconfiguration in this BIST approach, the embedded processor only has to modify configuration bits associated with the I/O cells under test while leaving existing connections from the TPGs and to the ORAs unchanged along with the actual configuration of the TPGs and ORAs themselves. Therefore, once the initial BIST configuration is downloaded, the embedded processor 1) executes the BIST sequence, 2) reads the BIST results, and 3) reconfigures the I/O cells under test before repeating this 3-step process. In the case of dynamic partial reconfiguration, the test time can be further reduced since the BIST results can be read at the end of the complete set of BIST configurations with only minor loss in diagnostic resolution; specifically, the faulty I/O cell can still be identified but the faulty mode of operation cannot.

FPGAs that lack the ability for configuration memory readback via the processor core further increase the difficulties associated with testing. This is the case with the Atmel AT94K series SoC. The configuration memory in AT94K SoC is byte addressable and typical byte associated with an I/O cell contains both logic bits and routing bits. This is illustrated by the example in Figure 7 where five bits control the multiplexer selection while the other three bits control the output buffer drive capability. Without the ability to read the configuration

memory, it is difficult, if not impossible, to analyze the configuration memory bits to determine which input of the multiplexer is selected. Thus, testing of the drive capabilities D2 and D3 through reconfiguration from the embedded processor cannot be achieved without possibly disconnecting the TPG. Consequently, amount of programmable resources that can be tested in the I/O cell is further reduced. When the configuration memory can be read by the embedded processor core, on the other hand, we are able to test more of the logic and routing resources associated with the I/O cells by performing read-modify-write operations on the configuration memory from the embedded processor core as we reconfigure the I/O cells under test for the next BIST configuration.

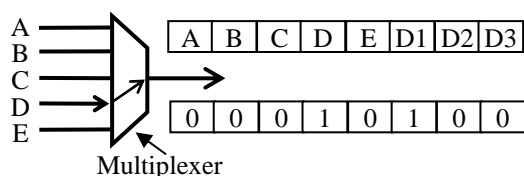


Fig. 7. Problem with no configuration memory read

IV. EXPERIMENTAL RESULTS

We implemented and verified the system-level I/O cell BIST approach on a number of FPGAs to observe the behavior and problems that can be encountered on real printed circuit boards. We discuss these observations in the following subsections. Of most interest, were the effects of system loading on the performance of the I/O cells. Other experimental results include the fault detection capabilities of the BIST approach for the Xilinx Virtex-4 and Atmel AT94K series SoC.

A. Effects of System Loading

The bi-directional buffers configured to test the I/O cells during BIST will have different load characteristics depending on the way they are terminated and whether they are normally an input, output, or bi-directional ports during system operation. Therefore, we would expect the I/O cells that are connected to heavy external loads to fail if they are tested at a high frequency. All of the I/O buffers can be tested at a single considerably slow frequency that is guaranteed to be sufficiently slow to allow fault-free I/O cells to pass. However, this may result in faulty I/O cells escaping detection in the case of delay faults. On the other hand, the I/O buffers can be grouped together by loading characteristics to be tested at different frequencies. The following example illustrates the sensitivity of the BIST approach to external loading factors.

A printed circuit board with an Atmel AT94K40 SoC has adjacent output buffers connected to light emitting diodes (LEDs) that are then connected to ground via

adjacent resistors in a common resistor network, as illustrated in Figure 8. The routing on the printed circuit board is identical for all of these nets in terms of length and width of the routes. As a result, the loading characteristics are similar for all output buffers. The only difference is the colors of the LEDs and their associated voltage and current characteristics as noted in Figure 8. The BIST was performed on the I/O cells and the I/O buffers indicated failures as noted in Figure 8. When the test clock frequency is reduced gradually, the I/O cells connected to the red and yellow LEDs began passing while the other I/O cell connected to the green LED continued to fail. As the BIST clock frequency was reduced further, all I/O cells began to pass the BIST. The BIST clock period and frequency is indicated in Figure 8 at the point where the I/O cells fail the BIST and at the point where the I/O cells pass the BIST. It should be noted that the failing test configuration in this example used the pull-down resistor on the pad. A similar situation occurred when configuring the Schmitt-trigger option on the input buffers but, in this case, the I/O cells connected to red LEDs failed while the while the I/O cells connected to the yellow and green LEDs passed. When the BIST clock frequency was reduce, these I/O cells passed and the BIST clock frequency was the same as that which the red LEDs passed in the pull-down example above. All other programmable options including output drive capabilities, TTL/CMOS input options, and pull-up configurations showed no difference; all pass the BIST at the maximum BIST clock frequency. This example gives a good indication of the sensitivity of the BIST approach to external system loading effects as well as the need to group I/O cells for testing at different BIST clock frequencies.

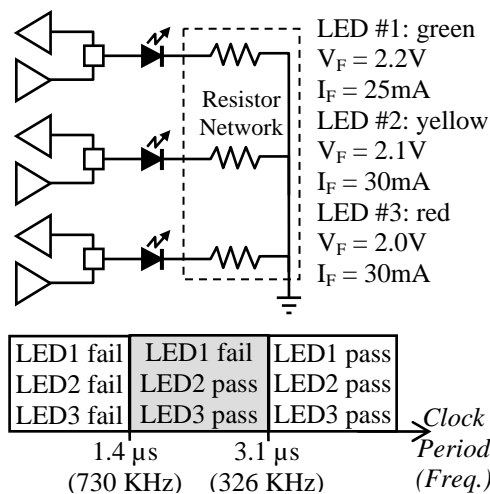


Fig. 8. LED failures vs. period of BIST clock

It should also be noted that boundary scan could not be used to test these connections without the assistance of a

mechanism for monitoring the LEDs to see if they are lit or not. Based, on the results from this example, on the other hand, it seems feasible that the BIST approach could be used to determine if the LEDs are lit or not based on the pass/fail characteristics of the test without visual monitoring.

B. Implementation with VHDL

A parameterized VHDL model of the BIST was developed and synthesized in different Atmel SoCs (AT94K10 and AT94K40) and Xilinx FPGAs (Virtex I, Spartan II, Virtex II Pro, Spartan III, and Virtex-4) using the Leonardo synthesis tool as well as the Xilinx ISE synthesis tool. The physical implementation and layout of the synthesized designs on the FPGAs were then investigated using the vendor-specific physical design tools like Figaro for Atmel and FPGA Editor for Xilinx. It was observed, as noted in the previous section, that the user has little or no control over the activation of the flip-flops in the I/O cells and that it appears that heuristics in the synthesis algorithms make the decision of whether a given flip-flop will be incorporated in the I/O cell or in the core of the FPGA.

When flip-flops are not activated, fault coverage for Atmel AT94K SoCs of only about 25% can be achieved by testing the I/O cells in two BIST configurations since only two test configurations can be applied when the flip-flops are not activated. Whereas, fault coverage for the Xilinx Virtex-4 I/O cells is much less since there are more flip-flops but none of them can be tested using the VHDL approach when the flip-flops are not activated. Even without the flip-flops, as many as seven BIST configurations can be applied to the Virtex-4 to test delay options, drive capabilities, slew rate, pull-up/down, and I/O voltage standards. But these tests must be performed on different groups of I/O cells at various frequencies once the system loading on each pin has been characterized.

C. Implementation with Vendor-Specific HDDL

A parameterized MGL program was developed to generate the BIST configurations for user-defined sets of I/O cells. The MGL program is executed in Figaro, Atmel's physical design graphical editor for place and route of the design to be implemented in an AT40K series FPGA or the FPGA core of an AT94K series SoC. In Atmel FPGAs, there are two types of I/O cells, referred to as primary and secondary, which are categorized according to their location and access to the programmable logic blocks along the perimeter of the FPGA core. The primary and secondary I/O buffers cannot be tested at the same time due to routing contentions when connected the TPGs to the I/O cells and, as a result, two different set of BIST configurations are needed. The primary I/O cells can be tested in 13 con-

figurations whereas the secondary I/O buffer can be tested in 10 configurations. Figure 9 gives the individual and cumulative fault coverage for the primary and secondary BIST configurations.

As can be seen in Figure 9, slightly less than 100% fault coverage is obtained with these two sets of BIST configurations with the remaining undetected fault in each buffer detected by a third set of BIST configurations. This fault is a programmable connection from each I/O cell to the global reset in the device and, as a result, only one I/O cell can be tested at a time such that the number of configurations in this third set is equal to the number of I/O cells in the device. Therefore, dynamic partial reconfiguration via the embedded processor is critical in obtaining efficient testing time. Since the TPG is connected to all inputs of the multiplexer, as illustrated in Figure 6, all routing resources associated with the I/O cell can also be tested via partial reconfiguration from the embedded processor. A more detailed description of these three sets of BIST configurations is given in [12].

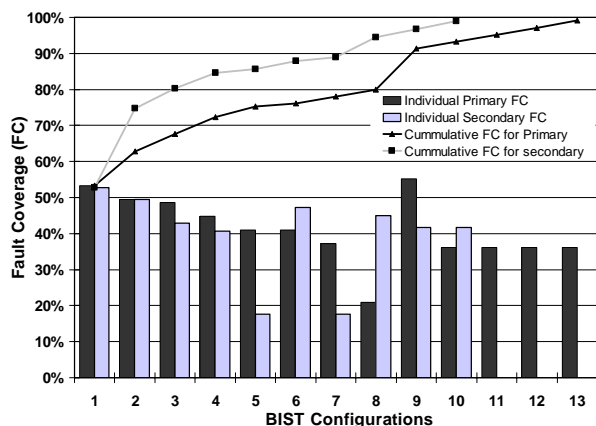


Fig. 9. Fault coverage for Atmel I/O buffers

All programmable I/O cells in the Virtex FPGAs (including Virtex-4) are identical and, as a result, can be tested at the same time. A parameterized C program was developed to generate the XDL BIST design for Virtex-4 based on the user specification of the number and the location of the I/O cells to be tested. The XDL file includes the design information (both configuration and placement) for the TPGs, ORAs and I/O cells configured for a particular mode of operation in which all flip-flops are activated. The XDL file is then converted to an NCD file and routed using Xilinx's place and route tool PAR or with FPGA Editor [18]. The routed NCD file is then converted back to XDL format where the I/O cells are reconfigured in different possible modes of operation for each BIST configuration to be executed. The reconfigured XDL files are converted back to NCD files from which the configuration bit streams can be generated using the Xilinx BitGen tool [18]. The bit streams are then downloaded into the

FPGA to execute and verify the BIST configuration. This process is summarized in Figure 10 and is similar to that described in [17]. Similar C programs would need to be developed for other the Virtex and Spartan series FPGAs.

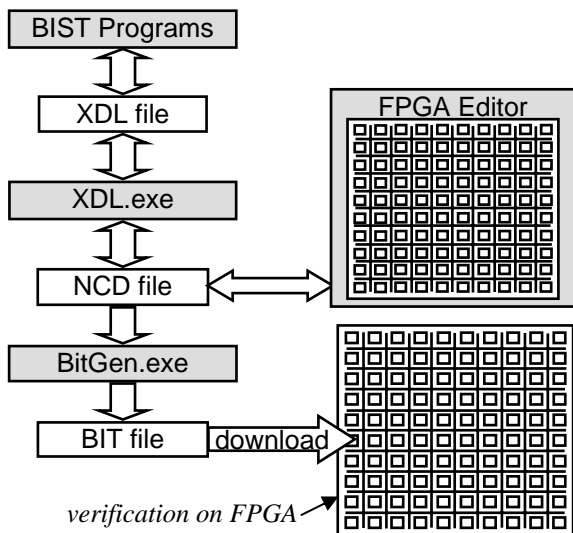


Fig. 10. BIST configuration generation process

The minimum number of possible configurations required to test the I/O cell in the bi-directional mode of operation was determined by performing fault simulation on a gate level model of the logic associated with the I/O cell. It was found that the I/O cells can be tested in 7 configurations and that there are a couple of undetectable faults when configured in the bi-directional mode of operation. While these faults are detectable when configured as an input buffer, they cannot be detected by the BIST approach. The resultant fault coverage is illustrated in the Figure 11.

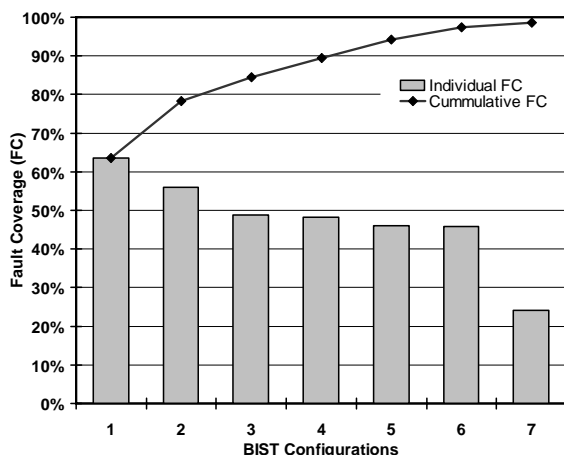


Fig. 11. Fault coverage for Virtex-4 I/O buffers

All of the features summarized in Table I for Virtex-4 can be tested with the exception of the 69 I/O voltage standards and the 64 delay options. However, there are only 7 I/O voltage standards associated with bi-

directional mode of operation and all of these can be tested. For the delay options, we can test all “power-of-2” values (i.e., 1, 2, 4, 8, 16, 32, and 64) during the seven BIST configurations which should give a reasonably good indication of whether the programmable delay options are working.

V. SUMMARY AND CONCLUSIONS

We have presented a BIST to test the programmable I/O cells in FPGAs and configurable SoCs. System-level BIST for the I/O cells is important since boundary scan is only capable of testing the I/O buffers and generally not capable of testing the programmable logic and routing resources associated with the I/O cell. We have observed that these programmable resources in unbonded pads are frequently used by synthesis tools to implement the system function. Furthermore, the amount of logic and routing resources associated with the I/O cell tends to be increasing with newer FPGA architectures. Therefore, it is critical that these resources be tested along with the programmable logic, routing, and specialized functions (like RAMs, DSPs, etc) in the core of the FPGA.

The resources in the core of the FPGA can be tested using the same BIST configurations at both manufacturing and system-level testing since all loading is internal and, as a result, remains constant. The I/O cells, however, cannot be tested using the exact same BIST configurations for both manufacturing and system-level testing due to the effects of external system loading. This was observed in practice and illustrated by in the example of the LEDs with identical nets in the previous section. Either the BIST clock frequency must be reduced to allow the BIST to pass for the slowest, most heavily loaded, I/O cell or the I/O cells under test must be grouped according to their loading characteristics. We found that the latter approach provides the better chance for fault detection in the system, particularly, given the sensitivity of the BIST to external loading.

While there has been a great deal of emphasis on grouping I/O cells and testing them separately, it should be noted that the separate testing only implies the BIST clock frequency associated with a given set of I/O cells. In practice, multiple I/O cell BIST circuits, each running at a different clock frequency, can be configured in the FPGA to run simultaneously. This helps to minimize the total number of downloads to the FPGA in the system and, hence, the overall system-level testing time. Of course, there may be architectural issues, such as the primary and secondary I/O cells in the Atmel AT94K series SoC, that prevent all I/O cells from being tested in a single download.

The BIST approach can detect most, if no all, stuck-at faults present in programmable logic and routing re-

sources associated with an I/O cell. However, we expect that there may be limitations in detecting parametric faults which do not cause catastrophic changes in V_{OL} , V_{OH} , V_{IL} , V_{IH} , current sinking and sourcing, fine resolution delay, etc.. On the other hand, this approach can detect major defects present in the analog programmable features such as pull-up, pull-down, tri-state, etc. Overall, this BIST approach reliably detects faults beyond the capabilities of boundary scan. Furthermore, the ability of the BIST approach to perform the tests at different frequencies provides another facet of testing not available with boundary scan.

Another limitation of this BIST approach is the development effort required for implementation. Parameterized VHDL appeared to be the most attractive method for implementation in any given system application since it has been proven to be effective to previous FPGA BIST development [8]. However, we found that the vendor-specific VHDL macros are generally limited to the I/O buffer and do not provide a mechanism for the necessary control of the complete I/O cell, of which the I/O buffer is only a small part. As a result, the VHDL approach does not provide the desired fault detection capabilities and what it does provide is, for the most part, the same as that obtained by testing via boundary scan.

On the other hand, vendor-specific hardware design description languages, such as MGL and XDL, provide the necessary control for good system-level testing. But, these HDDLs require significant development time and effort for a test developer to learn and work with them. This limitation is magnified when one considers the lack portability between different families of FPGAs, even those from the same vendor. On the positive side, the vendor-specific HDDL approach achieves higher fault coverage of the bonded I/O cells under test and also has the ability to test unbonded I/O cells that are also used to implement the system function. In addition, parameterization provides the ability to define groups of I/O cells to be tested together in any given system function. Hence, the system-level programmable I/O cell BIST approach is applicable to any commercially available FPGA or configurable SoC. Furthermore, the approach is compatible with dynamic partial reconfiguration via an embedded processor core to minimize the overall system-level testing time.

REFERENCES

- [1] C. Maunder and R. Tulloss, *Test Access Port and Boundary Scan Architecture*, IEEE Computer Society Press, 1990
- [2] 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
- [3] C. Stroud, K. Leach and T. Slaughter, "BIST for Xilinx 4000 and Spartan Series FPGAs: A Case Study," *Proc. IEEE International Test Conf.*, pp. 1258-1267, 2003
- [4] D. Fernandes and I. Harris, "Application of Built-In Self-Test for Interconnect Testing of FPGAs," *Proc. IEEE International Test Conf.*, pp. 1248-1257, 2003
- [5] C. Stroud, J. Nall, M. Lashinsky and M. Abramovici, "BIST-Based Diagnosis of FPGA Interconnect," *Proc. IEEE International Test Conf.*, pp. 618-627, 2002
- [6] X. Sun, J. Xu, B. Chan and P. Trouborst, "Novel Technique for BIST of FPGA Interconnects," *Proc. IEEE International Test Conf.*, pp. 795-803, 2000
- [7] J. Sunwoo and C. Stroud, "Built-In Self-Test of Configurable Cores in SoCs Using Embedded Processor Dynamic Reconfiguration," *Proc. International SoC Design Conf.*, pp. 174-177, 2005
- [8] C. Stroud and S. Garimella, "BIST and Diagnosis of Multiple Embedded Cores in SoCs," *Proc. International Conf. on Embedded Systems & Applications*, pp. 130-136, 2005
- [9] C. Jia and L. Milor, "A BIST Solution for the Test of I/O Speed," *Proc. IEEE International Test Conf.*, pp. 1023-1030, 2003
- [10] L. Zhao, D. Walker and F. Lombardi, "IDDQ Testing of Input/Output Resources of SRAM-Based FPGAs," *Proc. Asian Test Symp.*, pp. 375-380, 1999
- [11] S. Vemula and C. Stroud, "BIST for I/O Buffers in FPGAs," *Proc. IEEE North Atlantic Test Workshop*, pp. 31-36, 2005
- [12] S. Vemula and C. Stroud, "Built-In Self-Test for Programmable I/O Buffers in FPGAs and SoCs," *Proc. IEEE Southeastern Symp. on System Theory*, pp. 534-538, 2006
- [13] __, "AT94K Series Field Programmable System Level Integrated Circuit," Data Sheet, Atmel Corp., 2001 (available at www.atmel.com)
- [14] __, "Virtex-4 User Guide," User Guide UG070 (v 1.4), Xilinx, Inc., 2005 (available at www.xilinx.com)
- [15] __, "Integrated Development System AT40K Macro Library Version 6.0", Atmel Corp., 1998 (available at www.atmel.com)
- [16] __, "Integrated Development System Technical Reference and Release Notes Version 6.0", Atmel Corp., 1998 (available at www.atmel.com)
- [17] C. Stroud, J. Nall, A. Taylor and L. Charnley, "A System for Automated Generation of Built-In Self-Test Configurations for Field Programmable Gate Arrays," *Proc. International Conf. on Systems Engineering*, pp. 437-443, 2002
- [18] __, "Xilinx ISE 7 Software Manuals", Xilinx, Inc., 2005 (available at www.xilinx.com)