Chapter 12: Test Technology Trends in the Nanometer Age

12.5 FPGA Testing

*Field programmable gate arrays* (FPGAs) are generally composed of a two dimensional array of *programmable logic blocks* (PLBs) interconnected by a programmable routing network with programmable I/O cells at the periphery of the device as illustrated in Figure 12.5.1. Typical array sizes in terms of the number of PLBs range from 100 to over 22,000. A trend in most recent FPGAs is the addition of cores for specialized functions such as single and dual-port RAMs, FIFOs, multipliers, DSPs, and microprocessors. Memory cores vary in sizes from 128 bit to 18 Kbit depending on the series of FPGA where all memory cores are the same size in a given series. The number of these specialized cores is greater than 800 in the largest FPGAs currently available. As a result, the largest FPGAs, with over 22,000 PLBs and 800 specialized cores, easily reach the 100 million transistor mark and pose a testing challenge in terms of their size and diversity of functions.

![Figure 12.5.1: Typical FPGA Architecture](image)

The system function performed by the FPGA is controlled by an underlying configuration memory. In most current FPGAs, the configuration memory is a RAM ranging in size from 32 Kbits to 50 Mbits. The system function can be changed at anytime by simply rewriting the configuration memory with new data, referred to as reconfiguration. Another trend is FPGAs that support *dynamic partial reconfiguration* where a portion of the FPGA can be reconfigured while the remainder of the FPGA is performing normal system operation, also referred to as runtime reconfiguration. The size of the configuration memory is an important factor in testing FPGAs since the total testing time is usually dominated by downloading configuration data. However, dynamic partial reconfiguration and partial configuration memory readback capabilities can help to reduce testing time as will be discussed.

Each PLB consists of one or more *look-up tables* (LUTs) and flip-flops. The LUT typically has three or four inputs and is used to implement combinational logic. In some FPGAs, the LUT can also be programmed to function as a small RAM or shift register. The flip-flops, programmable as level sensitive latches in some FPGAs, are used to implement sequential logic. Typical PLB sizes range
from two 3-input LUTs with one flip-flop to eight 4-input LUTs with eight flip-flop/latches. A considerable amount of additional logic is incorporated in the PLB for implementing functions such as array multipliers, fast carry logic for adders, combining LUTs to construct larger combinational logic functions, etc. In addition to classical stuck-at faults in the PLB logic, the configuration memory bits that control logic function performed by the PLB must also be tested for stuck-at-0 and stuck-at-1 faults [Abramović 2001]. For complete testing, the PLBs must be tested in all of their modes of operation.

The programmable interconnect network consists of wire segments of various lengths and programmable switches that connect or disconnect the wire segments to form the signal nets required by the system function. Each programmable switch is controlled by a bit in the configuration memory. The typical number of wire segments associated with each PLB ranges from 50 to over 400 while the number of programmable switches ranges from 80 to over 1,000 per PLB. The number of configuration memory bits associated with the programmable routing resources is typically three to four times the number of configuration memory bits associated with the PLBs. As a result, the programmable interconnect network poses a bigger testing challenge than the programmable logic resources. The fault models used for testing the routing resources include shorts (bridging faults) and opens in the wire segments, wire segments stuck-at-1 and stuck-at-0, as well as programmable switches stuck-on and stuck-off, which include the controlling configuration memory bits stuck-at-1 and stuck-at-0. While the programmable switch stuck-off fault can be detected by a simple continuity test, stuck-on faults are similar to bridging faults and require opposite logic values be applied to the wire segments on both sides of the switch while monitoring both wire segments in order to detect the stuck-on fault [Stroud 2002b].

The programmability of FPGAs facilitates the implementation of a wide range of applications and, as a result, presents a number of testing solutions as well as a number of testing challenges. For example, FPGAs can be reprogrammed during system-level off-line testing to test other components and functions on a printed circuit board [Stroud 2002a]. Similarly, the PLBs and routing resources can be reprogrammed to test the other embedded cores within the FPGA such as memory and DSP cores [Stroud 2005b]. On the other hand, the programmability of the FPGA poses a number of challenges when it comes to complete and comprehensive testing of the FPGA itself. First, a large number of configurations must be downloaded into the FPGA to test the various programmable resources. Dynamic partial reconfiguration can reduce the total time associated with downloading these test configurations by writing only the portions of configuration memory that change from one test configuration to the next. The FPGA testing problem is further complicated by the growing size of FPGAs in terms of the PLB array, frequently changing architectures, as well as the introduction of specialized embedded cores such as RAMs and DSPs. If the FPGA can be completely tested and determined to be fault-free, the intended system function can be programmed onto the FPGA with a high probability of
proper operation. When faults are detected, the system function can be reconfigured to avoid the faulty resources if the faults can be diagnosed (identified and located). Therefore, diagnosis of the faulty resources is an important aspect of FPGA testing in order to take full advantage of the fault and/or defect tolerant potential of these devices.

Two types of testing approaches have been developed for FPGAs: \textit{external testing} and \textit{built-in self-test} (BIST). In external testing approaches, the FPGA is programmed for a given test configuration with the application of input test stimuli and the monitoring of output responses performed by external sources such as a test machine [Huang 1998] [Renovell 1998]. As a result, external test techniques are typically only used for manufacture testing. For FPGAs with boundary scan that support INTEST capabilities, the input test stimuli can be applied and output responses can be monitored via the boundary scan interface. Otherwise, the FPGA I/O pins must be used, resulting in package dependent testing. Most external test approaches seek to test all programmable resources in the FPGA independent of the system application to be programmed onto the FPGA, referred to as \textit{application independent testing}. Application dependent test approaches, on the other hand, seek to test only those resources that will be used by the intended system function [Tahoori 2004]. This reduces the number of test configurations that must be applied as well as the total test time.

The basic idea in BIST for FPGAs is to configure some of the PLBs as test pattern generators (TPGs) and output response analyzers (ORAs). These BIST resources are then used to detect faults in PLBs [Abramovici 2001], routing resources [Harris 2002] [Stroud 2002b] [Sun 2000], and special cores such as RAMs and DSPs [Stroud 2005b]. Once the programmable resources have been tested, the FPGA is reconfigured for the intended system function without any overhead or performance penalties due to the BIST circuitry. This facilitates system-level use of the BIST configurations. Different BIST architectures are used for testing PLBs (often referred to as logic BIST), routing resources (often referred to as routing BIST), and embedded cores. It is important to note that the processes used in these BIST approaches for reconfiguring and testing the specific target resources are very similar to those used in application independent external testing of FPGAs.

The most frequently used logic BIST architecture is illustrated in Figure 12.5.2 where the programmable logic blocks under test (BUTs) and ORAs are arranged in alternating columns (or rows) and multiple identical TPGs are used to drive the alternating columns (or rows) of BUTs [Abramovici 2001]. The output responses of the identically programmed BUTs are monitored by comparison-based ORAs in neighboring columns (or rows). During a given test session, the BUTs are repeatedly reconfigured in their various modes of operation until they are completely tested. Dynamic partial reconfiguration can be used since only the BUTs must be reconfigured while the TPGs, ORAs, and interconnections remain constant for the test session. During the next test session, the logic BIST
architecture is flipped and the roles of the PLBs are reversed such that those PLBs previously configured as TPGs and ORAs become BUTs and vice versa. All PLBs can be tested in only two test sessions when at least half the PLBs are configured as BUTs during a given test session. The total number of test configurations in each test session typically ranges from five to fifteen depending on the complexity of the PLB. After the completion of each BIST sequence the Pass/Fail contents of the ORAs can be read either via partial configuration memory readback or via a scan chain constructed by incorporating a multiplexer at the input to the ORA flip-flop shown in Figure 12.5.2c [Stroud 2002a]. Alternatively, as a result of dynamic partial reconfiguration, the ORA contents can be read at the end of each test session with a slight loss of diagnostic resolution; the faulty PLB(s) can be still identified but the faulty mode(s) of operation cannot. Faulty PLBs can be identified based on the BIST results using a diagnostic procedure developed for this logic BIST architecture [Abramovici 2001]. A similar architecture can be used to test and diagnose other embedded cores in the FPGA such as memories and DSPs [Stroud 2005b].

Two routing BIST approaches have proven to be effective in testing the programmable interconnect resources in FPGAs including the wire segments, programmable switches, and configuration memory bits that control the switches. One is a comparison-based approach, illustrated in Figure 12.5.3a, in which the TPG drives exhaustive test patterns over two sets of $N$ wires under test that are compared at the other end by comparison-based ORAs [Stroud 2002b]. The other approach is parity-based, illustrated in Figure 12.5.3b, where the TPG sources exhaustive test patterns over a set of $N$ wires under test and produces a parity bit that is sent to the ORA [Sun 2000]. The ORA generates parity over the data observed on the wires under test and compares the generated parity with the parity bit sent by the TPG. This approach was later modified to send the parity over a wire under test for a total of $N+1$ wires under test during a given BIST configuration. As in the case of logic BIST, the sets of wires under test are repeatedly reconfigured to test the various routing resources (wire segments and programmable switches) in the FPGA. The total number of test configurations required to completely test the routing resources typically ranges from 50 to 300 depending on the complexity of interconnect network and the PLB architecture used for constructing the TPGs and ORAs. Dynamic partial reconfiguration can be used to reduce the time to download test configurations. While both routing BIST approaches have been shown to be effective in detecting faults, the
A comparison-based approach has been extended to the diagnosis of faults in the programmable interconnect network for fault tolerant applications [Harris 2002]. By constructing many, small routing BIST circuits consisting of independent TPGs, ORAs, and sets of wires under test in the FPGA, diagnostic resolution is improved since an ORA fault detection indication identifies the self-test area containing the fault [Stroud 2002b].

More recent trends in FPGA testing include delay fault testing and using embedded processor cores for on-chip test configuration generation and application. Testing for delay faults in FPGAs is important since the transmission gates used to construct the programmable switches in the interconnect network are particularly susceptible to defects that affect the delay though the switches. External test techniques [Chmelar 2003] and BIST approaches [Abramovici 2003] have been developed to detect delay faults in FPGAs. The incorporation of embedded microprocessor cores that can write and read the FPGA configuration memory has facilitated the algorithmic generation of test configurations from within the FPGA instead of downloading test configurations. A relatively small program is stored in the program memory of the embedded processor core which is then used to reconfigure and test the programmable logic and routing resources as well as other embedded cores such as memories and DSPs. The embedded processor can then retrieve the test results and perform diagnosis [Stroud 2005a].

Recent complex programmable logic devices (CPLDs) are similar to FPGAs in that they contain programmable logic and routing resources as well as embedded cores such as RAMs, FIFOs, etc. The only noticeable difference is that CPLDs use programmable logic arrays (PLAs) for implementing combinational logic functions instead of the LUTs typically found in FPGAs. In addition, the PLBs in CPLDs tend to be larger in terms of the size of the PLAs and the number of flip-flops. Slightly different test techniques are used to test the reprogrammable PLAs [Stroud 2002a] with the remainder of CPLD testing being the same as that for FPGAs. Now that FPGAs are incorporating embedded cores such as memories, DSPs, and microprocessors, FPGAs are more closely resembling System-on-Chip (SOC) implementations. At the same time, SOC's are incorporating more embedded FPGA cores. As a result, FPGA testing techniques are becoming increasingly important to a broader range of system applications.
12.6 High-Speed Interface

12.7 RF Testing

12.8 Concluding Remarks

Acknowledgments

The authors wish to thank Dr. Yao-Ting Wang, Chairman and CTO of Clear Shape, for contributing the DFM section, T. M. Mak, Senior Researcher of Intel, for contributing the Soft Errors, Fault Tolerance, and Defect and Error Tolerance sections, Dr. Samy Makar of Azul Systems for contributing the RTL Testing section, Prof. Charles Stroud of Auburn University for contributing the FPGA Testing section, Prof. Wen-Ben Jone of University of Cincinnati for contributing the MEMS Testing section, Dr. Mike P. Li, CTO of Wavecrest Corp., for contributing the High-Speed Interface section, Prof. Abhijit Chatterjee of Georgia Institute of Technology for contributing the RF Testing section, as well as Teresa Chang, and her lovely daughter Alice Yu, of SynTest Technologies for typing the text and drawing the figures.
References

R12.1 Books and Test Technology Roadmap


R12.2 Delay/Performance Testing

R12.3 Coping with Physical Failures, Soft Errors, and Reliability Issues


Chapter 12: Test Technology Trends in the Nanometer Age


**R12.4 System-Level Testing**

**R12.5 FPGA Testing**


...