Enhanced Independent Functional Testing of Xilinx FPGAs

Travis Haroldsen, Matthew French, Andrew Schmidt, and Devang Khamar
Information Sciences Institute
University of Southern California
Arlington, Virginia 22203
Email: tharoldsen/mfrench/aschmidt/dkhamar@isi.edu

***Abstract*—Independent functional testing of commercial FPGAs is an important first step in establishing a trusted supply-chain, determining the usability of devices stored in inventory, and monitoring the health status of fielded systems. In this work, we present the Independent FPGA Functional Testing tools for third-party testing of Xilinx FPGAs. These tools exhaustively exercise the millions of logic and interconnect settings and resources on the FPGA fabric to detect stuck-at-faults in the fabric and can be used to detect errors due to wear-out or to identify counterfeit parts.**

I. Introduction

Functional testing of commercial Field Programmable Gate Arrays (FPGAs), independent of in-house FPGA vendor production testing, is an important first step in establishing a trusted supply-chain, determining the usability of devices stored in inventory for long periods of time, and for determining the health status of fielded systems. While current and next-generation FPGAs are increasingly using emerging technology to thwart counterfeiting attempts, older FPGA generations are easily recycled and sold as new. Devices in deep storage may not have been stored properly, and devices under heavy use or in strenuous operating environments may experience wear out effects. Independent functional testing of the FPGA VLSI provides a sanity check that the device is in fact the device it claims to be and is in good working order. This is no trivial feat as modern FPGA devices contain over 1B transistors, a dozen types of Hard IP, 35M user wires, and 380M user routing switches.

The Independent FPGA Functional Testing (IFT) tools enable testing Xilinx FPGAs for stuck-at-faults in the fabric. The tools generate and use a suite of test bitstreams specifically developed to perform targeted testing of the underlying FPGA VLSI circuitry in a highly parallel manner. The controller and checker logic in IFT are fully implemented in the bitstream such that the only requirement for running IFT tests is a means of programming the FPGA and of reporting a result using standard programming interfaces. As such, IFT can be used at multiple stages in the supply chain, including in deployed systems. This makes IFT suitable for detecting errors in devices due to wear-out and in identifying counterfeit parts and can be used both to validate acquired parts and to monitor the state of fielded devices. While previous work has explored various aspects of FPGA self-test, we believe this is the most complete, scalable set of tools for independently testing Xilinx FPGAs in a timely and thorough manner.

II. Independent Functional Testing Tools

The IFT tools generate and perform independent functional tests that can be used to cross-check the FPGA manufacturer's testing and can also be used for field testing of potentially counterfeit, damaged, or aging parts. The ability to develop and employ such tests relies upon exhaustive knowledge of the internal FPGA architecture and the ability to perform low-level, precise manipulations to a circuit. By leveraging the regular structures found in FPGAs, the IFT tools replicate tests for one resource across the other similar components on the chip. This enables testing significant portions of an FPGA in parallel and provides a tractable and scalable testing solution.

IFT utilizes stuck-at-fault modeling and generates tests as customized bitstreams which are programmed onto an FPGA, exercise the circuitry, detect any stuck-at-faults in the exercised portions of the underlying VLSI of the FPGA resources of interest and alert the presence of one or more stuck-at-faults via a single bit output pin. To perform tests and report results, IFT requires access to a clock and an output pin; however, IFT can make use of the configuration clock and DONE pin on the STARTUP primitive on Xilinx FPGAs thereby removing any requirements for access to external IO. With this approach, results can be obtained by the host via either the JTAG or SelectMap interfaces making IFT suitable for use on previously deployed systems.

IFT currently covers the configuration memory, the user logic (configurable logic and block RAM), and much of the user routing interconnect of the Xilinx 7 Series devices. This approach is scalable to include other hard-IP resources, can be extended to include other device failure tests, and can be ported to other device generations or FPGA vendors.

A. Configuration Memory Testing

Configuration memory is tested by programming a bitstream to the device, performing readback of the bitstream and comparing the read back bitstream with the original.

Vendor provided masks are used to identify untestable bits in the bitstream. The configuration memory tests provide a verification of device power up and basic device capabilities such as configuration controller operation and possibly provide early detection of defective parts before more extensive testing is performed.

B. Configurable Logic Block Testing

Extensive testing is performed on the configurable logic blocks (CLBs) and includes searching for stuck-at faults:

- through all paths through the CLBs
- in configurations of and paths through the look-up tables (LUTs)
- through the dedicated vertical carry chains paths
- in the functionality of the flip-flops, including reset and clock enable
- in the "distributed RAM" [1]
- in the Shift Register LUTs (SRL)

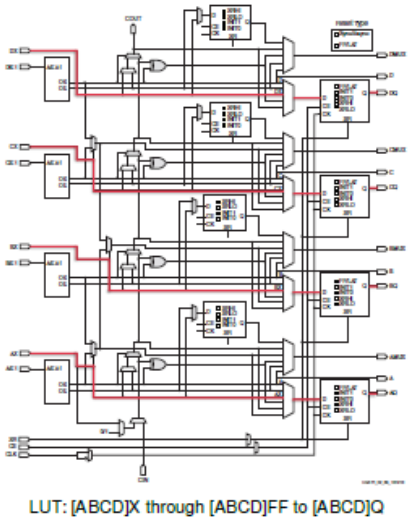

Fig. 1. One path testing the configurable logic.

IFT tests of the routing paths and LUTs using a launch and capture approach. In most cases, this tests the targeted paths for both stuck-at-0 and stuck-at-1 faults. A number of test patterns, such as the one shown in Figure 1, have been manually created to exercise the various paths through the CLBs. Paths that are not covered in these test patterns are covered in the flip-flop, RAM and SRL tests. The patterns are designed such that the output of one CLB is connected to the input of its neighbor and the results of each CLB are forward through the created a chain. This allows a significant number of resources in each device to be tested in a single pass and results in a single pass-fail output.

In IFT, the dynamic RAM in the CLBs are verified using a MATS [2] memory test patterns. This pattern detects stuck-at faults in the reading, writing and addressing functionality of the RAMs. The SRLs are verified by forward alternating 0/1s through the chain. All CLB registers are tested for writing 0s and 1s as well as validating proper functioning of the reset and clock enable signals. All together, testing of the CLBs is accomplished with 31 test patterns, though the additional controller logic means many of the patterns require multiple bitstreams to test all CLBs in a device.

C. Block RAM Testing

Block RAM (BRAM) are tested in through the execution of various test pattern generators to expose stuck-at, transition and coupling faults in the SRAM cells and faults in the address encoder. All BRAMs in a device are tested simultaneously by connecting a test pattern generator to the inputs of each BRAM and and the outputs of each BRAM to two output response analyzers (ORAs). The ORAs compare the outputs of the BRAMs against one another and forward any errors through a connected chain of ORAs. The BRAMs and ORAs are connected in such a way that any differences in outputs between BRAMs will be detected. This architecture is similar to the one described in a [3].

Test patterns used in testing include MATS+ and March LR [4], [5] test patterns, and the MarchS2PF [6] test pattern for testing the BRAM's dual port mode. Additional testing is performed to validate the functionality of the cascading and FIFO modes of the BRAMs. These tests are configured in a manner such that all BRAMs on a device are tested simultaneously.

D. Interconnect Testing

The Xilinx Kintex 7 FPGAs contain more than 10 million programmable interconnect points (PIPs). A viable hardware validation tool needs to test as many of these PIPs as possible and do so in a timely manner. This requires producing test design that can exercise a large number of these interconnect points simultaneously. To accomplish this, we create targeted, massively replicable paths to exercise user wires and PIPs in parallel. Our early efforts attempted to the Xilinx ISE router to route constrained paths containing individual tool-selected PIPs. However, analysis showed that the ISE router did not properly handle the provided constraints leading to large gaps in coverage.

In place of the Xilinx ISE router approach, we have developed our own custom "Waypoint" router within our Tools for Open Reconfigurable Computing (TORC) toolset [7] to create routing paths which correctly exercise the PIPs we are testing. With the "Waypoint" router, we can specify a particular PIP and find a path from a launch CLB, through the PIP, and to a capture CLB. Using this router, we can specify a set of PIPs, identify launch and capture CLBs for each PIP, and identify valid routes to test each of the PIPs. Results from each capture CLBs can then be forwarded along the vertical carry chain to the next capture CLB and to the final output.

In Xilinx architectures, all switch blocks contain an identical set of 3,744 PIPs each (though some PIPs on the edges of the device may not be reachable). When selecting a set of PIPs to target, our tools select one of the 3,744 classes of PIPs and attempt to test that class of PIP in each switch block in the

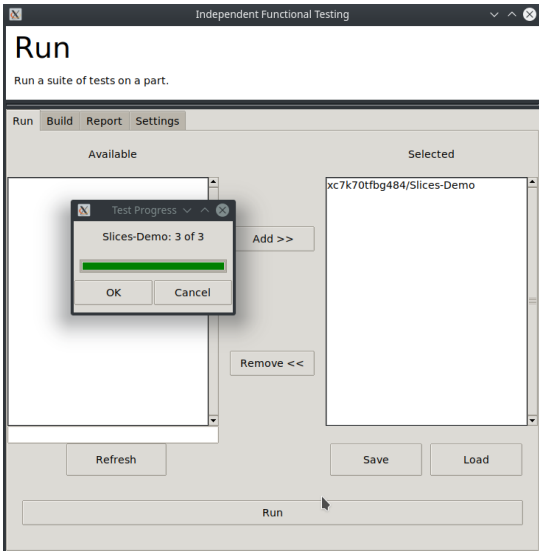

Fig. 2. Graphical User Interface for IFT application.

device. We currently limit ourselves to primarily targeting the switch blocks that are adjacent to a CLB. With 3,744 classes of PIPs, the approach yields the equivalent number of bitstreams to test the interconnect. With this approach, we are currently able to target 68% of PIPs in the switch blocks. It should be noted that this number includes PIPs that are not reachable in the fabric and other tests will cover additional PIPs leading to higher coverage.

III. Use and Verification

To aid in building and running the tests, IFT is packaged with a front end graphical user interface (shown in Figure 2). This allows a user to quickly specify a suite of tests to be performed, log any information about the testing (part identifier for example), perform the tests and view the results with just a few button presses. We plan to include support for logging and presenting test coverage metrics in the future.

IFT currently supports the Xilinx 7-Series architectures (Virtex7, Kintex7, Artix7, Zynq700). Additional Xilinx ISE-based architectures can be added with a one-time porting effort. Support for any given architecture includes all devices within that architecture. IFT has been tested on a Xilinx Kintex 7 xc7k70t FPGA to prove validity of the approach. Testing includes 100% of slices and block RAMs and a more than 65% of the interconnect. Full testing of the slices, including RAM and shift register capabilities was accomplished in under 8 minutes using JTAG running at 6MHz. Full interconnect testing will take substantially longer. Most of this time was spent programming the device and the run time could be significantly improved by using the faster SelectMap interface.

IV. Conclusion and Future Plans

The Independent FPGA Functional Testing tools enable testing of Xilinx FPGAs. We have demonstrated the capabilities on a Kintex 7 FPGA. Future work includes covering more resources, such as the DSP multipliers, clock networks and clocking resources. We are in the process of porting IFT to more devices including Virtex 4. Further support for families prior to Series 7 should be straightforward using the current approach. We believe supporting Ultrascale and later families will be possible but will require new tools for performing interfacing with Xilinx Vivado.

We also include to improve our metrics for reporting test coverage. Metrics we will report on include percent of bitstream bits and percent of interconnect covered. As part of this, we have developed a tool to trace a routing path from the source to the sink to report of PIPs tested along the path. Coverage metrics will be reported in the reports section of the GUI.

Acknowledgement

This material is based research sponsored by the Air Force Research Labs (AFRL) and the Defense Advanced Research Projects Agency (DARPA) under agreement number FA8560-18-1-7817. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views, and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Research Labs (AFRL), the Defense Advanced Research Projects Agency (DARPA), or the U.S. Government.

References

[1] Xilinx, *UG474: 7 Series FPGAs Configurable Logic Block User Guide*, September 2016. [Online]. Available: https://www.xilinx.com/support/documentation/user_guides/ug474_7Series_CLB.pdf

[2] A. J. V. D. Goor, "Using March Tests to Test SRAMs," *IEEE Design Test of Computers*, vol. 10, no. 1, pp. 8–14, March 1993.

[3] J. L. Dailey, B. R. Garrison, M. D. Pulukuri, and C. E. Stroud, "Built-In Self-Test of Embedded Memory Cores in Virtex-5 Field Programmable Gate Arrays," in *2011 IEEE 43rd Southeastern Symposium on System Theory*, March 2011, pp. 220–225.

[4] A. J. van de Goor, G. N. Gaydadjiev, V. G. Mikitjuk, and V. N. Yarmolik, "March LR: A Test for Realistic Linked Faults," in *Proceedings of 14th VLSI Test Symposium*, April 1996, pp. 272–280.

[5] V. D. Goor, I. B. S. Tlili, and S. Hamdioui, "Converting march tests for bit-oriented memories into tests for word-oriented memories," 1998.

[6] S. Hamdioui and A. J. van de Goor, "Efficient tests for realistic faults in dual-port srams," *IEEE Transactions on Computers*, vol. 51, no. 5, pp. 460–473, May 2002.

[7] N. Steiner, A. Wood, H. Shojaei, J. Couch, P. Athanas, and M. French, "Torc: Towards an Open-Source Tool Flow," in *Proceedings of the 19th ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, ser. FPGA '11. New York, NY, USA: ACM, 2011, pp. 41–44. [Online]. Available: http://doi.acm.org/10.1145/1950413.1950425