Testing Interconnects of Dynamic Reconfigurable FPGAs

Chi-Feng Wu

Dept. of Electrical Engineering
National Tsing Hua University
Hsinchu, Taiwan 300
e-mail: cfwu@slugger.ee.nthu.edu.tw

Cheng-Wen Wu

Dept. of Electrical Engineering
National Tsing Hua University
Hsinchu, Taiwan 300
e-mail cww@ee.nthu.edu.tw

Abstract— Field Programmable Gate Arrays (FPGAs) are an increasingly popular choice for fast prototyping and for products whose time to market is relatively short. Testing FPGAs before programming them is thus becoming a major concern to the manufacturers as well as the users. In this paper we propose a universal test for the interconnects of typical dynamic reconfigurable FPGAs. The proposed test configurations and corresponding test patterns for the Xilinx XC6200 FPGAs are shown to cover all interconnect faults. In our test, the total number of test configurations is only 7, which is independent of the FPGA size. The test time for XC6216 is less than 5ms.

I. Introduction

With the advent of deep-submicron VLSI technology, system-on-a-chip is no longer a dream. However, as the integration density and design complexity of system chips keep increasing, the time required for design verification is becoming extremely long. Field Programmable Gate Arrays (FPGAs) thus are receiving a growing attention. They provide a good hardware prototyping and emulation platform, resulting in short turn-around time for product development. They also can be used in first-generation products that need to get into the market soon.

A typical FPGA consists of an array of function units and interconnect channels/matrices. The function units and interconnect switches are *programmable*, i.e., they can be configured to perform different logic functions. For a RAM-based FPGA, the configuration data (normally generated by software tools) is downloaded to its control memory before it can be used as a chip designed to the specified function. The FPGAs can be reprogrammed for virtually unlimited times.

There are many FPGA architectures developed in the past for different applications. A look-up table (LUT) based FPGA contains several *look-up tables* in a function unit, which is also called a *configurable logic block (CLB)*. It normally uses surrounding segmented wiring channels and switch matrices for interconnect. LUT-based FPGAs are widely used for hardware prototyping because of its flexible function and interconnect configuration; e.g., Xilinx XC4000-series [1] and Altera FLEX10K-series [2]. On the other hand, with a hierarchical architecture, dynamic reconfigurable FPGAs employ multiplexers in both function and interconnect configuration; e.g., Xilinx XC6200-series [3]. Unlike the LUT-based FPGAs which must complete their configuration in one pass and can not be partially reconfigured, the configuration memory of a dynamic reconfigurable FPGA can be changed partially. A section of the device can be reconfigured without disturbing circuits already configured in other sections. The enhanced programmability is dedicated for custom computing [4], so they are also called *reconfigurable processing units (RPUs)*.

FPGA Testing can be done in two ways: testing the unprogrammed FPGA (the universal test) and testing the programmed FPGA. The latter is normally done by a user with test patterns generated for the target circuit configured into the FPGA; however, such user patterns are not efficient even for faults in the configured circuit because of the technology mapping problem [5]. An FPGA can realize a great amount of different functions, so testing all possible configurations to verify the correctness of the FPGA is not feasible. However, by proper fault modeling and careful selection of configurations, the FPGA can still be fully tested. A test sequence that fully test the FPGA without exercising all possible configurations (i.e., with only a limited amount of test configurations) is called a *universal test* [6]. It is universal because it has nothing to do with the target circuit. Note that a universal test still requires different *test configurations* (TCs) and their corresponding *test patterns* (TPs). TC generation is a very time-consuming process; moreover, TC downloading occupies most of the testing time, i.e., time(TC)$\gg$time(TP) for each TC. To speed up the universal testing process, we must reduce the number of TCs while still able to cover all programmable resources of the FPGA, i.e., function units and interconnects.

Methodologies for testing interconnects in LUT-based FPGAs have been proposed in [7–9]. Also, the problem of testing function units in dynamic reconfigurable FPGAs is addressed in [10]. In this paper, we focus on testing interconnects in dynamic reconfigurable FPGAs. The next section gives an overview of the FPGA architecture. Fault models and the proposed multiplexer test strategy are presented in Sec. III. Based on the fault models and test strategy, we present a complete test for basic interconnects in the FPGA, in Sec. IV. Conclusions are given in Sec. V.

II. Architecture

The Xilinx XC6200-series FPGAs are dynamic-reconfigurable FPGAs [3]. Based on a fine-grain hierarchical

architecture, its smallest building block is called the *basic cell*. A large array of cells are connected in the form of *sea of gates* at the lowest hierarchy. Routing in this level is implemented by the nearest-neighbor interconnect (or *basic interconnect*). Neighboring cells are further grouped into 4×4 blocks. Each block has length-4 wires connecting cells in the same row and column and even to the neighboring blocks. Likewise, length-16 wires connect the 4×4 array of such blocks (i.e., 16×16 array of basic cells). Finally, an XC6216 FPGA consists of a 4×4 array of 16×16 blocks and length-64 wires, as shown in Fig. II. The capacity is increased by hierarchically integrating more blocks in a similar way.

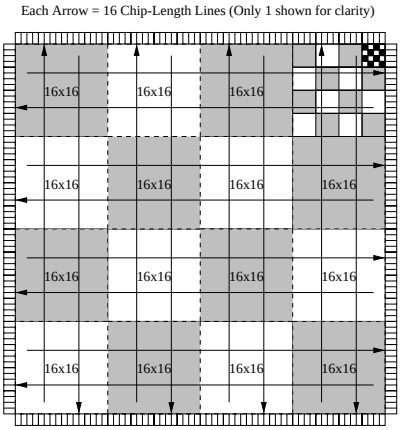

Fig. 1. XC6216 routing hierarchy.

The basic cell (BC) implements simple functions and provides basic interconnects between neighboring cells. As illustrated in Fig. 2, a BC consists of a function unit and several routing multiplexers. The logic functions of the BC are generated by the function unit. The function unit can be configured to implement any two-input gate, a 2:1 multiplexer, constant 0 or 1, single input functions (buffer or inverter), or any of these in addition to a D-type flip-flop. Note that the implicit select lines of the multiplexers are controlled by bit-values of the configuration memory.

III. Fault Model and Test Strategy

Multiplexer is the elementary component of XC6200. As illustrated in Fig. 2, testing the interconnects is tantamount to testing the routing multiplexers and wires at each level of the hierarchy. Functional model and assumptions of the multiplexer are defined below before we introduce the test strategy.

A multiplexer is a group of switches which pass one of the inputs to the output according to the configuration of switches: the switch for the selected input is on and all others are off. The multiplexers (switches) are actually controlled by the configuration data. Therefore, we consider the multiplexer as configurable switches (see Fig. 3). Exactly one switch is on at any time. Two assumptions for multiplexer switches are as follows:

- If all switches are off (open), the output is either stuck-at-1 or stuck-at-0.

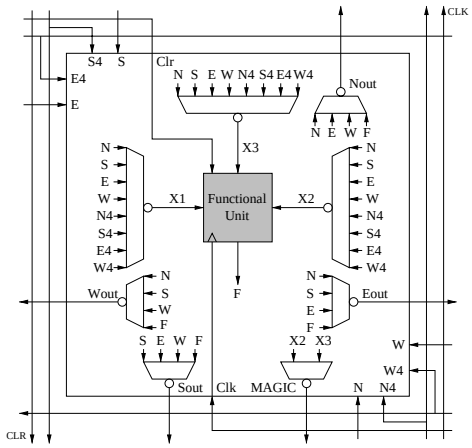

Fig. 2. Basic cell.

- If two switches are on (closed) simultaneously, the output is equivalent to the wired-OR of the two selected inputs.

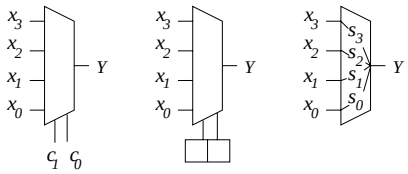

Fig. 3. Multiplexer in the FPGA.

When a switch is always on regardless of the configuration, it is a stuck-on switch; similarly, a stuck-off switch is always off. Note that a test that detects stuck-on/off faults of a multiplexer also detects stuck-at faults of its I/O wires.

Theorem 1 *A test which detects all switch stuck-on/off faults of a multiplexer also detects its I/O stuck-at faults.*

Proof: The test which detects a switch stuck-off fault must exercise a transition on its data path. Obviously it can also detect stuck-at faults on the corresponding data input and output. A stuck-at fault on a configuration (control) input results in multiple stuck-off faults. For example, in Fig. 3, a stuck-at-0 on c_0 results in stuck-off faults on s_1 and s_3. Therefore, stuck-at faults on configuration inputs are covered by the test for stuck-off faults. □

Consider a multiplexer with data inputs $X = \{x_{k-1}, \ldots, x_1, x_0\}$, switches $S = \{s_{k-1}, \ldots, s_1, s_0\}$, and output $Y = x_c$. Let the configuration input (i.e., switch control input) $C = c$, where c is a binary number and $0 \leq c \leq k-1$, then in the fault-free case

$$s_i = \begin{cases} \text{on} & \text{if } i = c; \\ \text{off} & \text{otherwise.} \end{cases}$$

To test the multiplexer for switch stuck-on and stuck-off faults, we define two test patterns (X vectors) for any configuration

vector c:

$$MP_c^0 = \langle mp_{k-1}^0, \ldots, mp_1^0, mp_0^0 \rangle$$

and

$$MP_c^1 = \langle mp_{k-1}^1, \ldots, mp_1^1, mp_0^1 \rangle,$$

where

$$mp_i^0 = \begin{cases} 0 & \text{if } i = c; \\ 1 & \text{otherwise,} \end{cases}$$

and $mp_i^1 = \overline{mp_i^0}$. Also, for ease of discussion, let $MP^1 = \{MP_{k-1}^1, \ldots, MP_1^1, MP_0^1\}$, $MP^0 = \{MP_{k-1}^0, \ldots, MP_1^0, MP_0^0\}$, and finally $MP = MP^1 \cup MP^0$.

It is obvious that MP activates any switch stuck-on and stuck-off fault, and the fault effect can be observed from the output Y. For example, when $c = 0$, MP_0^1 and MP_0^0 together activate the stuck-off fault of s_0, since if the switch is always off then it will fail to pass at least one of the values 0 and 1. Additionally, MP_0^0 activates the stuck-on faults of all switches except s_0, because each of these faults results in a faulty output value (i.e., 1) according to the second assumption of the multiplexer model. Note that when wired-AND logic is assumed instead of wired-OR, MP still activates all switch stuck-on and stuck-off faults, where the stuck-on faults are activated by MP^1 instead of MP^0.

Bridging (short) faults on input nets of the multiplexer are also covered by MP^0 if the fault behavior is equivalent to wired-OR logic, or by MP^1 if wired-AND is assumed.

IV. Testing Basic Interconnects

Basic interconnects are implemented by four four-input multiplexers in the basic cell, as shown in Fig. 4. Parallel testing of these multiplexers is achieved by three TCs, as shown in Fig. 5. In the figure, we show only a 2×2 array for clarity. It can be directly extended to any $N \times N$ array and tested with the same approach.

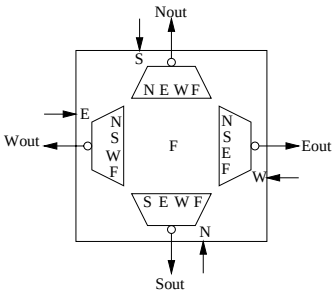

Fig. 4. Basic routing switches in BC.

Multiplexers whose outputs are Nout, Wout, Sout, and Eout are denoted as M_N, M_W, M_S, and M_E, respectively. In the test configuration $TC = (c_1, c_2, c_3, c_4)$, c_1 defines the switch control inputs for M_N, c_2 for M_W, c_3 for M_S, and c_4 for M_E, respectively. For example, if $TC = (E, S, W, N)$, it means that switches E, S, W, and N are turned on in M_N, M_W, M_S, and M_E, respectively. Also, the orthogonal test configuration as shown in Fig. 5 is $TC_o = (N, W, S, E)$.

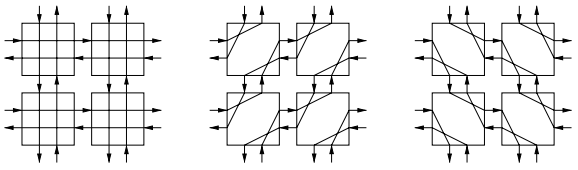

Fig. 5. TCs for basic interconnects.

The test pattern is denoted as $TP = \langle t_S, t_E, t_N, t_W \rangle$, where t_S is the two-bit south-bound test sequence, t_E the east-bound test sequence, t_N the north-bound test sequence, and t_W the west-bound test sequence, respectively. According to the test strategy presented in the previous section, applying MP^0 and MP^1 to multiplexers in parallel is our goal. For the orthogonal test configuration, we apply the two-pattern tests $TP_o^0 = \langle 01, 10, 10, 01 \rangle$ and $TP_o^1 = \langle 10, 01, 01, 10 \rangle$ to achieve this goal. Note that F is the output of the function unit which is not shown for simplicity. By proper hierarchical routing and configuration of the function unit as a buffer or an inverter, the required value of F in each cell can be assigned. As a result, TP_o^1 and TP_o^0 deliver MP^1 or MP^0 to all cells in parallel with two sets of F values as shown in Figs. 6 and 7, respectively. In these figures we show symbolic maps on the right, where the triangles represent the multiplexers at the corresponding locations in the cells. Inside each triangle, there is a circle if MP^0 is applied to the corresponding multiplexer, and a cross ($\times$) if MP^1 is applied instead. Combining these maps we see that MP is successfully applied to each and every multiplexer. Similarly, the $+45^\circ$ diagonal TC is $TC_{d+} = (E, S, W, N)$, and its two-pattern tests are $TP_{d+}^0 = \langle 00, 00, 11, 11 \rangle$ and $TP_{d+}^1 = \langle 11, 11, 00, 00 \rangle$. For the -45° diagonal TC, $TC_{d-} = (W, N, E, S)$, the two-pattern tests are $TP_{d-}^0 = \langle 00, 11, 11, 00 \rangle$ and $TP_{d-}^1 = \langle 11, 00, 00, 11 \rangle$.

All stuck-on and stuck-off faults of the basic interconnect switches are covered except the stuck-off faults of the F switches. Testing the F switches requires four configurations: TC_{F_N}, TC_{F_W}, TC_{F_S}, and TC_{F_E}, where, e.g., TC_{F_W} is illustrated in Fig. 8. It is clear that if w receives $\langle 01 \rangle$ then the F switch stuck-off fault in M_W is detected. The other three F switch stuck-off faults are covered in a similar way.

In any of the orthogonal, 2 diagonal, and 4 F configurations, fault effects are propagated in the respective directions to the primary outputs, so given their corresponding test patterns, the 7 TCs test all stuck-on and stuck-off faults in M_S, M_E, M_N, and M_W. By Thm. 1, all multiplexer I/O stuck-at faults are covered. The test MP also covers bridging faults between data inputs lines. Bridging faults between adjacent interconnect wires are detected by the orthogonal TC and the corresponding patterns, which guarantees that adjacent wires carry complementary values during test. In summary, the basic interconnects are fully tested by 7 TCs.

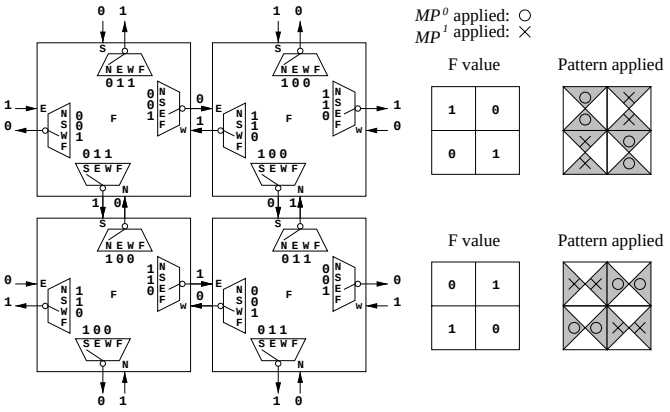

Fig. 6. Orthogonal configuration with TP_o^0.

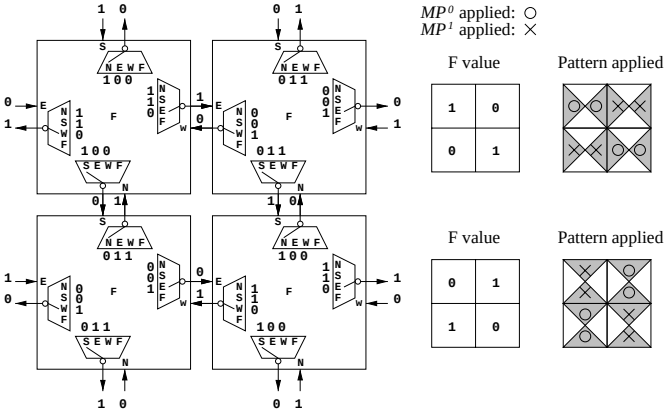

Fig. 7. Orthogonal configuration with TP_o^1.

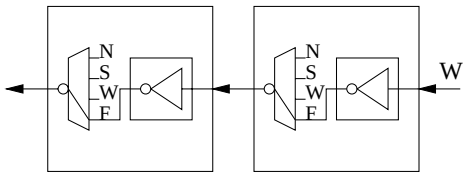

Fig. 8. TC for F switch stuck-off fault in M_W.

V. Conclusions

We have proposed a universal test for basic interconnects of Xilinx dynamic reconfigurable FPGAs. We started from testing the elementary components (the multiplexers) by the test pattern MP, which covers all stuck-on and stuck-off faults of the multiplexer switches, stuck-at faults of the multiplexer I/Os, and shorts between the data inputs. A universal test for the basic interconnects of the Xilinx XC6200-family FPGAs have been proposed. The test contains 7 TCs—1 orthogonal, 2 diagonal, and 4 F configurations. This configuration count is independent of the array size. As each configuration takes about 0.66ms to download for the XC6216 FPGA [3], its interconnect testing time is less than 5ms.

References

[1] Xilinx, *The Programmable Gate Array Data Book*. San Jose, California: Xilinx, Inc., 1996.

[2] Altera, *FLEX 10K Embedded Programmable Logic Family Data Sheet, Ver. 3*. Altera, Co., Jan. 1998.

[3] Xilinx, *XC6200 Field Programmable Gate Arrays*. San Jose, California: Xilinx, Inc., Apr. 1997.

[4] D. A. Buell, J. M. Arnold, and W. J. Kleinfelder, *Splash 2: FPGAs in a custom computing machine*. Los Alamitos, CA: IEEE Computer Society Press, 1996.

[5] K. Kwiat, W. Debany, and S. Hariri, "Effects of technology mapping on fault-detection coverage in reprogrammable FPGAs," *IEE Proc.-Comput. Digit. Tech.*, vol. 142, pp. 407–410, Nov. 1995.

[6] T. Inoue, H. Fujiwara, H. Michinishi, T. Yokohira, and T. Okamoto, "Universal test complexity of field-programmable gate arrays," in *Proc. Fourth Asian Test Symp. (ATS)*, (Bangalore), pp. 259–265, Nov. 1995.

[7] H. Michinishi, T. Yokohira, T. Okamoto, T. Inoue, and H. Fujiwara, "A test methodology for interconnect structures of LUT-based FPGAs," in *Proc. Fifth Asian Test Symp. (ATS)*, (Hsinchu), pp. 68–74, Nov. 1996.

[8] W.-K. Huang, X.-T. Chen, and F. Lombardi, "On the diagnosis of programmable interconnect systems: Theory and application," in *Proc. 14th IEEE VLSI Test Symp. (VTS)*, pp. 204–209, 1996.

[9] M. Renovell, J. M. Portal, J. Figueras, and Y. Zorian, "Testing the interconnect of RAM-based FPGAs," *IEEE Design & Test of Computers*, vol. 15, Jan.-Mar. 1998.

[10] C.-F. Wu and C.-W. Wu, "Testing function units of dynamic reconfigurable FPGAs," in *Proc. 9th VLSI Design/CAD Symp.*, (Nantou), pp. 189–192, Aug. 1998.