Minimizing the Number of Test Configurations for FPGAs

Erik Chmelař
Center for Reliable Computing
Stanford University*

***Abstract*— FPGA test cost can be greatly reduced by minimizing the number of test configurations. A test technique is presented for FPGAs with multiplexer-based routing architectures in which multiple logical paths through each multiplexer is enabled instead of only one path. It is shown that for Xilinx Virtex-II and Spartan-3 FPGAs only 8 test configurations are required to achieve 100% stuck-at, PIP stuck-on, and PIP stuck-off fault coverage.**

Index Terms— **FPGA, test configuration generation.**

I. Introduction

A *Field-programmable Gate Array* (FPGA) is a configurable integrated circuit able to implement an arbitrary logic design. An FPGA must be tested for defects, requiring many iterations of: (1) program the device with a test configuration, and (2) apply test stimuli and observe the response. However, the time spent programming the device with each configuration is several milliseconds [1], much larger than the several microseconds required to apply test stimuli [2]–[5]. Thus, test cost is dominated by configuration time.

Test time can be decreased while maintaining fault coverage by reducing (1) the number of configurations, or (2) the configuration time. The presented configuration generation technique addresses the former, enabling multiple logical paths through each switch matrix multiplexer instead of only one path. Unlike normal operation, during this test mode multiple inputs of a given switch matrix multiplexer drive its output. By applying appropriate test stimuli to each multiplexer, all stuck-at faults on interconnects and logic element inputs and outputs, stuck-on PIPs, and stuck-off PIPs can be detected (PIPs are defined in Sec. III). Signal contentions result; however, since they are limited to a single clock cycle, they are not damaging to the FPGA. It is shown that only 8 device configurations are required for 100% stuck-at, PIP stuck-on, and PIP stuck-off fault coverage in Xilinx Virtex-II and Spartan-3 FPGAs.

The organization of this paper is as follows. Relevant configuration generation techniques are surveyed in Sec. II. A generic FPGA structure is given in Sec. III. The presented configuration generation technique is developed in Sec. IV. Finally, the paper concludes in Sec. V.

II. Previous Work

Although most configuration generation techniques consider only faults in the logic resources [6]–[8], some do consider faults in the more challenging-to-test interconnection network [9]–[12]. Those addressing faults in the interconnection network derive a reduced set of test configurations using interconnect modeling and graph traversal algorithms [9], [10].

A technique using a maximum flow algorithm is developed in [9] that achieves 100% interconnect stuck-at, stuck-open, and pairwise bridge fault coverage in Virtex FPGAs. However, the 8 test configurations developed consider only those PIPs that join interconnects to other interconnects; PIPs that join interconnects to logic blocks and the transistors that implement the routing multiplexers are not considered (unlike newer architectures, multiplexers in Virtex are outside of the switch matrices and contain at least as many transistors).

The minimum number of configurations to test all routing resources of an FPGA using graph traversal algorithms corresponds to the size of the largest multiplexer in the device. Because Virtex has 28:1 multiplexers [13], 28 configurations are required, one per logical path through a given 28:1 multiplexer (multiple multiplexers can be tested in parallel). Because switch matrices of contemporary FPGAs are completely multiplexer-based, this limitation is especially important.

A *Built-in Self Test* (BIST) technique is developed in [10] that addresses this limitation, employing a maximum flow algorithm to generate 59 test configurations that detect stuck-at, pairwise bridge, and interconnect stuck-open faults on a large subset of the routing resources in Virtex FPGAs.

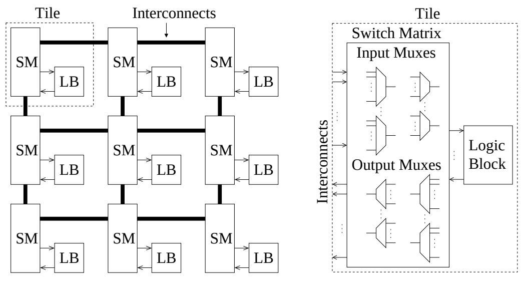
III. FPGA Structure

An FPGA contains both logic and routing resources. The logic resources are the hardware within the basic building blocks. The primary building blocks are *Logic Blocks* (LBs) and *Input/Output Blocks* (IOBs); contemporary FPGAs now contain some multiplier blocks and RAM blocks as well.

Logic blocks contain the combinational and sequential elements needed to perform logic functions: *SRAM Look-up Tables* (LUTs) implement combinational functions, and bistables are used in sequential designs. Input/output blocks pass signals between an FPGA and an external device.

Blocks are interconnected by the routing resources, or interconnection network—interconnects, *Switch Matrices* (SMs), *Programmable Interconnect Points* (PIPs), multiplexers, buffers, and vias. Blocks and interconnects are grouped together into tiles, which are replicated throughout the FPGA to create the entire layout, shown in Fig. 1a.

*Now working at LSI Logic in Milpitas, CA.



(a) 3×3 Tiled FPGA Layout (b) Tile

Fig. 1. FPGA Structure

In contemporary FPGAs—Virtex-II and Spartan-3—switch matrices are composed of multiplexers: input multiplexers route signals from interconnects to blocks and output multiplexers route signals from blocks to interconnects, shown in Fig. 1b. These switch matrix multiplexers are implemented as two levels of NMOS pass transistors. Fig. 2 shows a 16:1 switch matrix multiplexer (16 inputs, 1 output) and its level-restore circuitry. One pass transistors and its associated SRAM control cell is called a PIP. In Virtex-II and Spartan-3 FPGAs the maximum-sized switch matrix multiplexers are 32:1.

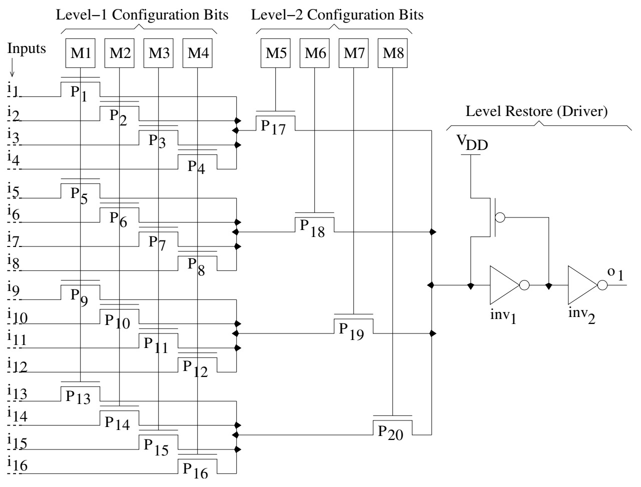


Fig. 2. 16:1 Two-level Switch Matrix Multiplexer

IV. Multiplexer-based Testing

A. Overview

All FPGA test techniques to date are based on programming the FPGA with several legal configurations and applying test stimuli for each. A *legal* configuration is one that causes no signal contentions. Under this normal operation, any given switch matrix multiplexer is configured such that only one logical path from one of its input to its output is active. A legal configuration for the 16:1 multiplexer of Fig. 2 requires that only one bit in the level-l configuration SRAM cell group ($M1 \dots M4$) and one bit in the level-2 group ($M5 \dots M8$) are set to logic-1, enabling a path from one input ($i_1 \dots i_{16}$) to the output (o_1). A multiplexer configuration in which all bits are set to logic-0 is also legal if that multiplexer is not used.

By setting more than one configuration bit in each group to logic-1—an *illegal* configuration—several logical paths through the multiplexer can be enabled. Since the constraints of graph traversal configuration generation algorithms no longer apply (more than one logical path through a multiplexer is allowed), the number of configurations required to test an entire FPGA can be significantly reduced, to only 8 for the Virtex-II and Spartan-3 FPGAs, as explained in Sec. IV-B.

B. Test Configurations

Enabling multiple logical paths through a multiplexer causes several inputs to be shorted. Because switch matrix multiplexers are implemented using NMOS pass transistors, pull-down paths (logic-0) are much stronger than pull-up paths (logic-1). By correctly choosing the logical paths (configuration of the multiplexer) and the input stimuli (pull-down and pull-up paths), all stuck-at faults and all PIP stuck-on and stuck-off faults can be detected (discussed in Sec. IV-B.1). Since all signals are routed via these multiplexers, testing all multiplexer inputs and outputs is both necessary and sufficient to test all interconnects and logic element inputs and outputs.

To activate and observe faults on all multiplexers using *Automatic Test Equipment* (ATE), the multiplexers are joined together into *Iterative Logic Arrays* (ILAs): the output of one multiplexer is joined to an input of another via an interconnect or logic element. Therefore, the configuration generation process must be addressed at two levels: (1) the multiplexer level, to determine the logical paths and input stimuli of a single multiplexer, and (2) the chip level, to determine how to join the multiplexers together to form ILAs.

1) Multiplexer Level Configuration: SPICE simulations on a 90 nm, 1.2 V technology—the same technology node of Spartan-3—were conducted to find a minimum set of configurations to achieve 100% stuck-at, PIP stuck-on, and PIP stuck-off fault coverage for a single multiplexer. Although Virtex-II and Spartan-3 FPGAs contain some 32:1 switch matrix multiplexers (also two-levels of NMOS pass transistors), a 16:1 multiplexer was analyzed to keep Table I small: extension to 32:1 multiplexer configurations is provided in Sec. IV-B.2.

Table I summarizes each of the 6 configurations that test all inputs, all PIPs, and the output of a 16:1 multiplexer. The first 4 configurations test for all stuck-at 0 and stuck-at 1 faults, all PIP stuck-off faults, and PIP stuck-on faults for PIPs $P_1 \dots P_{16}$ (see Fig. 2); the last 2 configurations test for PIP stuck-on faults for PIPs $P_{17} \dots P_{20}$. Because minimum-sized transistors are used (as done in practice), the results are independent of input driver size (driver of the preceding multiplexer or logic element) and output driver size (level-restore circuitry).

The first configuration of Table I sets the configuration bits to $M_1 \dots M_8$ =1000 1111, enabling 4 logical paths through the multiplexer, from inputs i_1, i_5, i_9, and i_{13} to output o_1. A stuck-at 0 fault on any of these 4 inputs can be detected by driving these inputs to logic-1 (other inputs don't matter). A stuck-at

TABLE I

TEST PARAMETERS FOR 16:1 SWITCH MATRIX MULTIPLEXER

Cfg. ($M_1 \dots M_8$)	Inputs[†] ($i_1 \dots i_{16}$)	Output[†] (o_1) Fault-free	Output[†] (o_1) Faulty	Faults[‡] Inputs ($i_1 \dots i_{16}$)	Faults[‡] PIPs ($P_1 \dots P_{20}$)	Tolerance ($k\Omega$) Input	Tolerance ($k\Omega$) PIP
1000 1111	1··· 1··· 1··· 1···	1	0	0··· 0··· 0··· 0···	···· ···· ···· ···· ····	≤2.2	–
	0··· 1··· 1··· 1···	0	1	1··· ···· ···· ····	0··· ···· ···· ···· 0···	≤2.2	≥2.2
	1··· 0··· 1··· 1···	0	1	···· 1··· ···· ····	···· 0··· ···· ···· ·0··		
	1··· 1··· 0··· 1···	0	1	···· ···· 1··· ····	···· ···· 0··· ···· ··0·		
	1··· 1··· 1··· 0···	0	1	···· ···· ···· 1···	···· ···· ···· 0··· ···0		
	1000 1000 1000 1000	1	0	···· ···· ···· ····	·111 ·111 ·111 ·111 ····	–	≤0.5
0100 1111	·1·· ·1·· ·1·· ·1··	1	0	·0·· ·0·· ·0·· ·0··	···· ···· ···· ···· ····	≤2.2	–
	·0·· ·1·· ·1·· ·1··	0	1	·1·· ···· ···· ····	·0·· ···· ···· ···· 0···	≤2.2	≥2.2
	·1·· ·0·· ·1·· ·1··	0	1	···· ·1·· ···· ····	···· ·0·· ···· ···· ·0··		
	·1·· ·1·· ·0·· ·1··	0	1	···· ···· ·1·· ····	···· ···· ·0·· ···· ··0·		
	·1·· ·1·· ·1·· ·0··	0	1	···· ···· ···· ·1··	···· ···· ···· ·0·· ···0		
	0100 0100 0100 0100	1	0	···· ···· ···· ····	1·11 1·11 1·11 1·11 ····	–	≤0.5
0010 1111	··1· ··1· ··1· ··1·	1	0	··0· ··0· ··0· ··0·	···· ···· ···· ···· ····	≤2.2	–
	··0· ··1· ··1· ··1·	0	1	··1· ···· ···· ····	··0· ···· ···· ···· 0···	≤2.2	≥2.2
	··1· ··0· ··1· ··1·	0	1	···· ··1· ···· ····	···· ··0· ···· ···· ·0··		
	··1· ··1· ··0· ··1·	0	1	···· ···· ··1· ····	···· ···· ··0· ···· ··0·		
	··1· ··1· ··1· ··0·	0	1	···· ···· ···· ··1·	···· ···· ···· ··0· ···0		
	0010 0010 0010 0010	1	0	···· ···· ···· ····	11·1 11·1 11·1 11·1 ····	–	≤0.5
0001 1111	···1 ···1 ···1 ···1	1	0	···0 ···0 ···0 ···0	···· ···· ···· ···· ····	≤2.2	–
	···0 ···1 ···1 ···1	0	1	···1 ···· ···· ····	···0 ···· ···· ···· 0···	≤2.2	≥2.2
	···1 ···0 ···1 ···1	0	1	···· ···1 ···· ····	···· ···0 ···· ···· ·0··		
	···1 ···1 ···0 ···1	0	1	···· ···· ···1 ····	···· ···· ···0 ···· ··0·		
	···1 ···1 ···1 ···0	0	1	···· ···· ···· ···1	···· ···· ···· ···0 ···0		
	0001 0001 0001 0001	1	0	···· ···· ···· ····	111· 111· 111· 111· ····	–	≤0.5
1000 0011[§]	0··· 0··· 1··· 1···	1	0	···· ···· ···· ····	··· ···· ···· ···· 11··	–	≤3.7
1000 1100[§]	1··· 1··· 0··· 0···	1	0	···· ···· ···· ····	···· ···· ···· ···· ··11	–	≤3.7

[†]0 = logic-0, 1 = logic-1, · = don't care.

[‡]0 = input stuck-at 0 or PIP stuck-off, 1 = input stuck-at 1 or PIP stuck-on, · = no detectable fault.

[§]Last two configurations and input stimuli are not unique, for example $M_1 \dots M_8 =$0001 1010 and $M_1 \dots M_8 =$0100 0101 are valid.

0 fault creates a pull-down path that dominates the remaining 3 pull-up paths and causes the output o_1 to transition to logic-0, shown in Fig. 3. Detection is possible even if the stuck-at 0 fault is modeled by a resistive short to ground less than or equal to 2.2 kΩ (shown as tolerance in Table I).

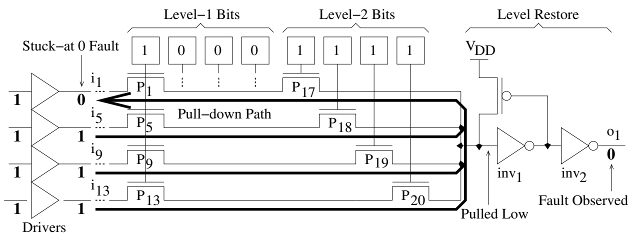


Fig. 3. Stuck-at 0 Fault Detection (row 1 of Table I)

With the same configuration, $M_1 \dots M_8 =$1000 1111, a stuck-at 1 fault on input i_1, i_5, i_9, or i_{13} can be detected by individually driving each to logic-0 (static power dissipation is minimized by limiting the resulting signal contention to one clock period). If an input is stuck-at 1, then a pull-down path cannot be created and the output o_1 remains at logic-1. Detection is possible even if the stuck-at 1 fault is modeled by a resistive short to V_{DD} less than or equal to 2.2 kΩ.

All stuck-off PIPs (non-conducting) along a pull-down path can be detected, since it breaks the pull-down path and causes the output o_1 to remain at logic-1. Even a partially stuck-off PIP with resistance greater than or equal to 2.2 kΩ can be detected (typical on resistance is several hundred ohms).

To detect PIP stuck-on faults, inputs i_1, i_5, i_9, and i_{13} are simultaneously driven to logic-1 while the rest are driven to logic-0. Thus, if any off PIP is stuck-on, a pull-down path is created that dominates the remaining 4 pull-up paths and causes the output o_1 to transition to logic-0. Even a partially stuck-on PIP with resistance less than or equal to 0.5 kΩ can be detected (the smaller resistive tolerance is because the pull-down path is fighting 4 pull-up paths rather than 3).

A total of 6 such configurations can achieve 100% stuck-at, PIP stuck-on, and PIP stuck-off fault coverage. Note that the resistive tolerance of the last two configurations of Table I, 3.7 kΩ, is larger than that of the others because the pull-down path is only fighting 2 pull-up paths rather than 3 (2.2 kΩ resistive tolerance) or 4 (0.5 kΩ resistive tolerance).

2) Larger Multiplexers: Contemporary FPGAs contain switch matrix multiplexers of varying sizes: Virtex-II and Spartan-3 FPGAs contain several 32:1 multiplexers. The simplest way to extend the 16:1 multiplexer configurations to 32:1 multiplexer configurations is to test the two halves of the 32:1 multiplexer independently, say the top half, inputs $i_1 \dots i_{16}$, and the bottom half, inputs $i_{17} \dots i_{32}$. For example, the first configuration of Table I, $M_1 \dots M_8 = 10001111$, extended to a 32:1 multiplexer with configuration SRAM cells $M_1 \dots M_{12}$, where $M_1 \dots M_4$ are the level-l bits and $M_5 \dots M_{12}$ are the level-2 bits, becomes $M_1 \ldots M_{12}$=1000 11110000 (top half) and $M_1 \ldots M_{12}$=1000 00001111 (bottom half).

The first 4 configurations of Table I can be extended in this manner, creating a total of 8 configurations. Fortunately, the last 2 configurations of Table I can be extended without increasing the total number of configurations: $M_1 \ldots M_8 =$1000 0011 becomes $M_1 \ldots M_{12} =$1000 00001111 and $M_1 \ldots M_8 =$1000 1100 becomes $M_1 \ldots M_{12} =$1000 11110000 (duplicates of the first and second extended configurations). Note that the resistive tolerance of the extended versions of the last two configurations in Table I is reduced from 3.7 kΩ to 2.2 kΩ because in each case the pull-down path is fighting 3 pull-up paths instead of 2 (tolerances for all other configurations remain the same). Table II summarizes the 8 configurations.

TABLE II
CONFIGURATIONS FOR 32:1 SWITCH MATRIX MULTIPLEXER

16:1 Mux ($M_1 \ldots M_8$)	32:1 Mux ($M_1 \ldots M_{12}$)	
1000 1111	1000 00001111	1000 11110000
0100 1111	0100 00001111	0100 11110000
0010 1111	0010 00001111	0010 11110000
0001 1111	0001 00001111	0001 11110000
1000 0011	1000 00001111 (see row 1 cfg. 1)	
1000 1100	1000 11110000 (see row 1 cfg. 2)	

3) Chip Level Configuration: Because all multiplexer inputs and outputs are tested, by definition all interconnects and logic element inputs and outputs are tested, (see Fig. 1b). To activate and observe all faults, the multiplexers are joined via the interconnects and logic elements into ILAs (not necessarily independent). The inputs and outputs of an ILA are primary inputs and outputs, controlled or observed by ATE. Each multiplexer is configured with one of the 8 configurations outlined in Table II (configurations for smaller multiplexers are easily derived from those of larger ones), yielding a total of 8 device configurations. After an FPGA is programmed with a test configuration, (1) test stimuli is applied to ILA inputs, (2) each ILA is clocked for its sequential depth (number of bistables), and (3) each ILA output is observed (the output of shorter ILAs is observed before that of longer ILAs).

The routing of ILAs (directions) and the configuration of each multiplexer is arbitrary, as long as the output of each multiplexer is propagated down the ILA to a primary output. Propagation is accomplished by (1) configuring at least one fault-free logical path through each multiplexer, and (2) configuring all bistables and LUTs as transparent elements.

First, a *fault-free logical path*—a path from an input of a multiplexer, through several turned-on (conducting) PIPs, to the output—must exist for the output of each multiplexer to propagate down the ILA to ATE. The output of a multiplexer, say mux_i, must be routed to the input to a fault-free logical path of the next multiplexer in the ILA, mux_{i+1}. For example, a configuration for a 16:1 multiplexer of $M_1 \ldots M_8 =$0000 1111 is useless since no signal can propagate through it unless one of its PIPs, $P_1 \ldots P_{16}$, is stuck-on. All configurations in Tables I and II meet this fault-free logical path criterion.

Next, all logic elements are configured as transparent logic. Each bistable is configured as a transparent D flip-flop. Each LUT is configured with a Boolean function whose value changes if at least one of its inputs changes, $F = A1_{fault-free} \cdot A2_{fault-free} \cdot A3_{fault-free} \cdot A4_{fault-free}$ (or F' if a negated LUT output is desired). Fig. 4 shows an example where $F = (\overline{A_1} \cdot \overline{A_2} \cdot \overline{A_3} \cdot \overline{A_4})'$. If any ILA inputs a faulty value to the LUT, all 4 ILAs propagate the value to a primary output. Note that each logic-0 input is really a transition from logic-1 to logic-0 for one clock period as stated in Sec. IV-B.1).

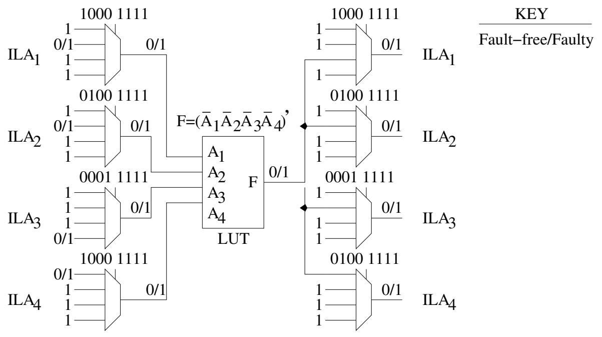


Fig. 4. 4 ILAs Each Joined by a LUT (don't care inputs not shown)

V. CONCLUSION

All FPGA test techniques to date are based on programming the FPGA with several legal configurations in which only one logical path is enabled through each switch matrix multiplexer. This paper shows that enabling multiple logical paths can minimize the number of test configurations, only 8 to achieve 100% stuck-at, PIP stuck-on, and PIP stuck-off fault coverage in the Xilinx Virtex-II and Spartan-3 FPGAs.

REFERENCES

[1] E. Chmelar, *FPGA Test and Diagnosis*. PhD thesis, Stanford University, June 2004.

[2] Xilinx, Inc., *Virtex Series Configuration Architecture User Guide*. Xilinx, Inc., San Jose, CA, 2003.

[3] Xilinx, Inc., *Virtex FPGA Series Configuration and Readback*. Xilinx, Inc., San Jose, CA, 2002.

[4] A. Doumar and H. Ito, “Testing the logic cells and interconnect resources for FPGAs,” *Proc. Eighth Asian Test Symp.*, pp. 369–374, 1999.

[5] S. Toutounchi and A. Lai, “FPGA test and coverage,” *Proc. Int. Test Conf.*, pp. 599–607, 2002.

[6] M. Renovell, J. Portal, J. Figueras, and Y. Zorian, “Test pattern and test configuration generation methodology for the logic of RAM-based FPGA,” *Sixth Asian Test Symp.*, pp. 254–259, Nov. 1997.

[7] M. Renovell, J. Portal, J. Figueras, and Y. Zorian, “Minimizing the number of test configurations for different FPGA families,” *Proc. Eighth Asian Test Symp.*, pp. 363–368, Nov. 1999.

[8] M. Renovell, J. Portal, J. Figueras, and Y. Zorian, “Test configuration minimization for the logic cells of SRAM-based FPGAs: A case study,” *Proc. European Test Workshop*, pp. 146–151, May 1999.

[9] M. Tahoori and S. Mitra, “Automatic configuration generation for FPGA interconnect testing,” *Proc. 21st VLSI Test Symp*, pp. 134–139, 2003.

[10] D. Fernandes and I. Harris, “Application of built-in self test for interconnect testing of FPGAs,” *Proc. Int. Test Conf.*, pp. 1248–1257, 2003.

[11] X. Sun, A. Alimohammad, and P. Trouborst, “Modeling of FPGA local/global interconnect resources and its derivation of minimal test configurations,” *Proc. 17th IEEE Int. Symp. Defect and Fault Tolerance*, pp. 284–292, Nov. 2002.

[12] X. Sun, J. Xu, A. Alimohammad, and P. Trouborst, “Minimal test configurations for FPGA local interconnects,” *Canadian Conf. Computer and Electrical Engineering*, vol. 1, pp. 427–432, May 2002.

[13] Xilinx, Inc., *Virtex 2.5 V Field Programmable Gate Arrays*. Xilinx, Inc., San Jose, CA, 2001.