Missouri University of Science and Technology
Scholars' Mine

Electrical and Computer Engineering Faculty Research & Creative Works

Electrical and Computer Engineering

01 Apr 2007

Implementation of Static and Semi-Static Versions of a 24+8x8 Quad-rail NULL Convention Multiply and Accumulate Unit

R. Sankar

V. Kadiyala

S. Kumar

S. Mohan

et. al. For a complete list of authors, see https://scholarsmine.mst.edu/ele_comeng_facwork/1675

Follow this and additional works at: https://scholarsmine.mst.edu/ele_comeng_facwork

Part of the Electrical and Computer Engineering Commons

Recommended Citation

R. Sankar et al., "Implementation of Static and Semi-Static Versions of a 24+8x8 Quad-rail NULL Convention Multiply and Accumulate Unit," *Proceedings of the IEEE Region 5 Technical Conference, 2007*, Institute of Electrical and Electronics Engineers (IEEE), Apr 2007.
The definitive version is available at https://doi.org/10.1109/TPSD.2007.4380351

This Article - Conference proceedings is brought to you for free and open access by Scholars' Mine. It has been accepted for inclusion in Electrical and Computer Engineering Faculty Research & Creative Works by an authorized administrator of Scholars' Mine. This work is protected by U. S. Copyright Law. Unauthorized use including reproduction for redistribution requires the permission of the copyright holder. For more information, please contact scholarsmine@mst.edu.

A Framework for the Detection of Crosstalk Noise in FPGAs

Sindhu Kakarla, Waleed K. Al-Assadi, senior Member, IEEE
Department of Electrical and Computer Engineering, University of Missouri–Rolla
1870 Miner Circle, Rolla, MO 65409 USA
Email: {sk9qd, waleed } @umr.edu

Abstract -**In recent years, crosstalk noise has emerged a serious problem because more and more devices and wires have been packed on electronic chips. As Integrated Circuits are migrated to more advanced technologies, it has become clear that crosstalk noise is the important phenomenon that must be taken into account. Despite of being more immune to crosstalk noise than their ASIC (Application Specific Integrated Circuit) counterparts, the dense interconnected structures of FPGAs (Field Programmable Gate Arrays) invite more vulnerabilities with crosstalk noise. Due to the lack of electrical detail concerning FPGA devices it is quite difficult to test the faults affected by crosstalk noise. This paper proposes a new approach for detecting the effects such as glitches and delays in transition that are due to crosstalk noise in FPGAs. This approach is similar to the BIST (Built-in Self Test) technique in that it incorporates the test pattern generator to generate the test vectors and the analyzer to analyze the crosstalk faults without any overhead for testing.**

Index- **Crosstalk noise, FPGA, Test Pattern Generator, Wire under Test (WUT).**

I. Introduction

As Integrated Circuits continue to migrate to more advanced technologies, crosstalk noise due to inter-wire capacitance of FPGA has become a critical concern for electronic designers due to the following factors: (1) interconnect scaling in one dimension, (2) continuous reduction in a device feature size, and (3) an increase in the domination of interconnect capacitance to the total interconnect capacitance [1]. Crosstalk noise occurs when a change in voltage on one trace causes a corresponding change in voltage on a nearby trace. If the affected trace (victim) is a global signal such as a clock or reset then the induced pulse might cause the circuit to incorrectly change the state, which in effect would manifest it into a functional error or change the switching time of the victim. A large switching time may lead to critical timing failures.

In recent years, FPGAs have become popular design fabrics because of their faster time-to-market, reprogrammability, low non-recurring engineering costs, and easy debugging. A typical FPGA structure is composed of Configurable Logic Blocks (CLBs), I/O blocks, and programmable interconnects. Each CLB consists of one or more basic logic elements (BLEs) each of which contains a look-up table (LUT), a flip-flop, and a multiplexer. The interconnect programmability is implemented by switches inside switch boxes and connection boxes [1]. Connection boxes allow CLB pins to connect to tracks, and switch boxes are used to build connections of appropriate lengths from prefabricated wire segments along the tracks.

An interconnect wire not only serves as a conductor of electrons but also behaves as a parasitic resistor, capacitor, or an inductor [2]. Process variations and manufacturing defects worsen this problem of noise and delay effects leading to unexpected unpredictability and increase in coupling capacitances and mutual inductances between interconnects [3]. Because of the present dense interconnect structure in FPGAs; coupling capacitance dominates result in crosstalk noise in FPGAs. This might cause neighboring switching nets (aggressors) to introduce noise pulses to their quiet neighbors (victims), resulting in a functional failure if the noise-induced incorrect value is latched. As Crosstalk noise in FPGAs has become a serious problem, this paper proposes a new approach similar to the BIST technique to detect crosstalk faults among interconnects in FPGAs.

This paper is organized as follows; Section 2 describes the nature of capacitive crosstalk, and Section 3 reviews the previous work that has been done on techniques used to detect crosstalk noise in FPGAs and crosstalk reduction techniques. Section 4 proposes a new FPGA crosstalk noise test architecture and section 5 presents a new version of interconnect routing cost function. Finally Section 6 concludes the paper.

II. Nature of capacitive crosstalk

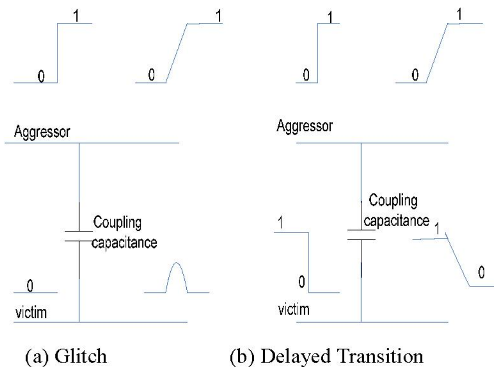
Crosstalk in FPGAs is mainly due to coupling capacitance coming from the physical adjacency between nets, possibly introducing noise pulses and delay variations on the quiet neighbors (victims). The effects of crosstalk can be modeled by an increase or decrease in the wiring capacitance of the victim net, which has a significant effect on the delay of a net [4]. Coupling between a pair of interconnects can result in two different crosstalk effects: a glitch, or a delayed transition, depending on the nature of the signal transitions in interconnects as shown in Fig. 1(a) and 1(b), respectively [5,6]. The glitches due to crosstalk noise may be either positive or

1-4244-1280-3/07/$25.00 ©2007 IEEE 2007 IEEE Region 5 Technical Conference, April 20-21, Fayetteville, AR

Authorized licensed use limited to: IEEE Xplore. Downloaded on January 20, 2009 at 14:35 from IEEE Xplore. Restrictions apply.

negative on the quiet neighbors, and the delay transition may be either rise time delay transition or fall time delay transition.

In addition to these effects, damped oscillations may be imposed on the top of a glitch. However, if the damping is large enough to alter the state of the circuit, then the oscillations can be approximated as either a glitch or a delayed transition [5]. As far as FPGAs are concerned the effect of crosstalk noise among interconnects is more observable as a delay variation (i.e., significant effect on delay of the net) or spurious transition [4].



(a) Glitch (b) Delayed Transition

Fig 1. Effects of crosstalk

This paper proposes a novel FPGA crosstalk test architecture to detect the crosstalk effects such as positive and negative glitches, as well as the rise and fall time delays among the interconnects in FPGAs.

III. Previous work

The major difficulties in testing the FPGA system for faults induced due to crosstalk noise are mainly a result of the almost complete absence of electrical detailing concerning an FPGA device. Crosstalk can become a real problem in two conditions: (1) If an aggressor switches at a specific time window with respect to the transition of a victim so as to have a significant effect on a victim's transition, called temporal correlation; (2) If an aggressor switches in an opposite or same direction with respect to a victim's transition [1]. Most of the previous work on crosstalk noise in FPGAs is focused on the crosstalk-induced delay, developing routing algorithms for crosstalk reduction for the applications that are eventually downloaded onto the FPGAs.

Among the techniques currently used for minimizing crosstalk effects are efforts to explore ways to reduce capacitive coupling. Coupling capacitance can be dramatically reduced if the spacing between the adjacent wires is increased. A reduction in the total loading capacitance can also be obtained by the wire spacing technique wherein the circuit's timing and power characteristics can be improved as well. The reduction of coupling capacitance will be directly converted into the reduction in delay variation due to crosstalk. The effectiveness of the wire spacing technique depends on the available routing space in an FPGA. If the total area of an FPGA structure is decided by the transistors, and if the extra spacing between wires does not cause any area penalty, then the wire spacing proves effective. However, for the architectures in which wires decide the total area, the wire spacing technique is ineffective. The other commonly used technique is power or ground shielding [1]. In this technique the VDD or GND track between two signal wires is placed to eliminate the effects of neighbors switching on the victim explicitly. Shielding is not as effective in improving the timing as is wire spacing with the same area overhead. The advantage of shielding is that it provides a good delay prediction, which introduces more confidence to noise control. This technique is mostly used for long lines crossing the entire FPGA structure and also for some dedicated interconnects such as clock and reset. The authors in [1] stated that on long parallel wires, crosstalk-induced delay can be easily accumulated and should be avoided. They proposed a new switch box design such that long parallel wires are reduced. In this paper's new approach of detecting crosstalk effects in FPGA, preference is given to testing the interconnect wires before they hit a switch box. In [4], a crosstalk-aware router is proposed, which is the enhancement of the Versatile Place and Route [VPR] timing router by assuming a simple model, in which crosstalk between two traces is modeled by a change in the effective capacitance seen by both traces. The VPR timing router is enhanced by modifying the cost function which specifies the complexity of routing process and timing model optimized for the delay that may be induced due to crosstalk [7]. The work in [8] proposes a new approach for crosstalk reduction through area routing using multiple possible connections between two points that are to be connected in this approach the crosstalk effects between the interconnects are avoided by routing the segments in various layers either as L-shaped or as Z-shaped where the L-shape or Z-shape indicates the direction of the routes. In [9], an online testing of non-logical faults such as crosstalk faults is proposed using an N-bit two rail code checker and functional block based on a finite state machine. This, however, limits the detection of crosstalk faults to the faults in CLBs. The functional block consists of N-basic cells, each connected to a monitored interconnect line and able to concurrently detect an undesired transition of the monitored line [9, 10]. The work in [11] presents an iterative logic array (ILA) method of finding delay faults among interconnects wherein the FPGA under test is configured to have one or more independent iterative logic arrays, on each of which propagates a signal transition. An ILA is a series of logic blocks and the associated interconnects under test. Considering the crosstalk noise in FPGA devices the detection of interconnects that may be affected by the crosstalk noise should be done at the manufacturing level. Based on this fashion, the newly proposed test architecture can be connected in a scan chain fashion.

IV. FPGA Crosstalk noise detection

Considering all the factors resulting in an increase in coupling capacitance among interconnects in an FPGA, this paper proposes a novel test architecture for the detection of

Authorized licensed use limited to: IEEE Xplore. Downloaded on January 20, 2009 at 14:35 from IEEE Xplore. Restrictions apply.

crosstalk-affected interconnects at the manufacturing level. The proposed approach concentrates mainly on MOTP (Manufacturing oriented testing procedure) of FPGA, where the FPGA device is tested for the crosstalk noise irrespective of the application that is being downloaded onto the device. The proposed approach is based on the Maximum Aggressor Fault model, wherein among the bunch of interconnects one interconnect is a victim and all other interconnects are aggressors. The Maximum Aggressor Fault Model is a high-level representation of all physical defects and process variations that lead to crosstalk errors such as positive glitches, negative glitches, and delays in transitions [5]. This study's main objective is to develop a technique similar to BIST to detect the crosstalk faults among interconnects in an FPGA device by connecting the CLBs of an FPGA in a scan chain fashion based on the boundary scan architecture [3]. As the cross-coupling of interconnects mainly results in effects such as glitches, or delayed transition, this study concentrates on detecting these effects among the interconnect structure in FPGA devices. An interconnect can act either as a victim or an aggressor. Efficiency of detecting the crosstalk noise depends on the test vectors being applied on the interconnects because the test pattern generator, which generates the test patterns, cannot be a regular counter for detecting the effects due to crosstalk, as the usage of a regular counter leads to a computationally expensive approach. Fig. 2 shows an example of interconnect system in which the transitions can detect the glitches, rise time, and fall time delay transitions due to the crosstalk noise in an interconnect system.

As shown in Fig. 2, the second interconnect line is assumed to be a victim, and the other lines act as aggressors. This is shown as an example and the test pattern generator in our design generates all the patterns assuming that each interconnect can act as a victim and the same interconnect can act as an aggressor for other interconnects.

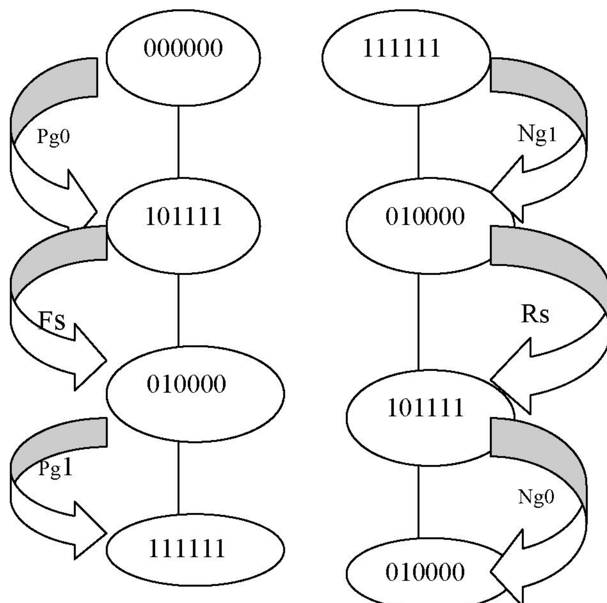


Fig. 2. Test vectors generated by Test Pattern Generator

The victim interconnects may be either at static high or static low logic values, which may be affected by the transition of the aggressors. Pg0, Pg1 and Ng0, Ng1 are positive and negative glitches, respectively. Rs and Fs are rising and falling skews in the victim line. Pg0 is the glitch that occurs on the victim line at static low values due to the transition in the aggressors from '0' to '1'. Pg1 is a glitch that may be obtained on the victim line, which is static high due to transition in the aggressors from '0' to '1', and similarly Ng0, Ng1 are the glitches on the victim line with the aggressors transitioning from '1' to '0'. Based on this model of different crosstalk effects new test architecture is proposed to detect the crosstalk noise among FPGA interconnects.

A. *The Test Architecture*

A careful observation of Fig. 2 shows that the victim line undergoes transition for every two clock cycles, whereas the aggressors undergo transition for every clock cycle. Based on this, a test pattern generator is obtained by configuring each CLB at the source side of the wire under test (WUT). This CLB is configured with the test architecture as shown in Fig. 3. To evaluate the presence or absence of crosstalk effects as stimulated by the test pattern generator on the source side of the WUT, present a new block called an "analyzer" is presented at the destination side of the WUT. The CLBs on the destination side of the concerned WUT are configured with the architecture of the analyzer as shown in Fig. 4.

The test architecture for the test pattern generator shown in Fig. 3 is configured in the CLBs at interconnects input sides. The test pattern generator is controlled by two external primary inputs, SE (Shift Enable), and CT (Crosstalk). For the input SE at logic'0', the initial vectors for the test pattern generator are scanned in. To shift the vectors among the CLBs in the scan-chain logic '1' is applied at pin SE. Flip-flop FF1 stores the victim-select data which indicate the line that is to be assumed as a victim in a bunch of interconnects. The AND gate controls the transitions on the victim interconnects and aggressor interconnects based on the victim select data and other input pin CT. Flip-flop FF2 stores the initial vectors that are scanned in before the actual testing procedure begins. Flip-flop FF3 is used to delay the transitions in interconnects that are assumed to be victims to two clock cycles based on the observations from Fig. 2. The multiplexer at the output of FF3 is used to set the frequency of the clock in flip-flop FF2 depending on the output from the AND gate (i.e., depending on the interconnect line whether it is assumed to be a victim or as an aggressor). The output of the flip-flop FF2, Q, is connected to the interconnect wire under test (WUT) and the other output of flip-flop FF2, is fed back to one of the inputs of the multiplexer followed by the FF1.

B. *Operation*

The primary inputs SE and CT control the overall operation of the test pattern generator. The testing procedure starts with the victim select data scanned into FF1 from the external inputs. Assuming a six-interconnects system and considering the

1-4244-1280-3/07/$25.00 ©2007 IEEE 2007 IEEE Region 5 Technical Conference, April 20-21, Fayetteville, AR

Authorized licensed use limited to: IEEE Xplore. Downloaded on January 20, 2009 at 14:35 from IEEE Xplore. Restrictions apply.

second interconnect as a victim net appropriate test patterns are generated as shown in Fig. 2. After the generation of test patterns for the second victim line, the line rotates with one-hot encoded data obtained by just scanning in a '0'. For example, if the second line is a victim, the victim select datum is "010000"; for the third interconnect to be a victim, '0' is scanned in to shift the victim select data one position to the right as "001000". In this manner all the interconnect lines in a bunch are assumed to be a victim at least once.

As shown from Fig. 2 aggressors transition once every clock cycle whereas victim undergoes transition once for every two clock cycles. This is achieved with a simple AND gate controlled by the victim select data from FF1 and the crosstalk- enabled primary input (CT). If the output of the AND gate is '1', then the data in FF2 are complemented for every two clock cycles, if it is '0' then the data in FF2 are complemented for every clock cycle. This is the main requirement for the test pattern generator in which the victim line undergoes transition for every two clock cycles and aggressor undergoes transition for every clock cycle.

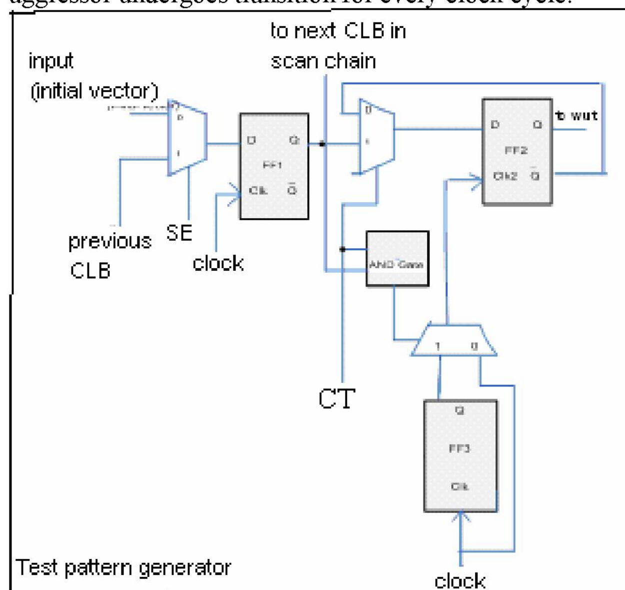


Fig. 3. Test Pattern Generator of one cell (CLB) in scan chain

For the generation of test patterns only the initial vectors are to be scanned into FF2 and the rest of the patterns are generated by the pattern generator itself. This structure is configured in a scan chain for all the CLBs that are connected to the WUTs and the scan procedure is enabled by the input pin SE. The main advantage of this pattern generator is that only initial vectors "000000" and "111111" need to be scanned in and the remaining test vectors will be generated by the testing architecture.

After the generation of patterns and detection of the faults assuming that one of interconnects in a bunch is a victim, a '0' is scanned into FF1 to change the victim, and the procedure repeats. As explained, the CLBs at the destination side of the WUTs are configured with the test architecture called the "analyzer" as shown in Fig. 4

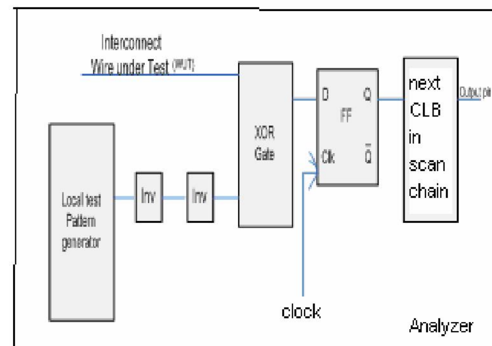


Fig 4 Block diagram of Analyzer

The analyzer shown in Fig. 4 mainly consists of a simple XOR gate for which one of the inputs is from the interconnects under test and the other input is from the local test generator, which generates the same test patterns as that of the original pattern generator. However, the local test generator is delayed by placing the basic inverters "inv". The number of inverters that should be placed depends on the specifications of the wire delay of the type of interconnects that are under test. The output of the XOR gate is given to flip-flop FF. If the output of the FF is '1' then it indicates that WUT is affected by the crosstalk. The output of the flip-flop from the analyzer is connected to the next CLB configured with same architecture shown in Fig. 4 to form a scan chain; i.e., all the outputs of flip-flops from CLBs at the destination side of WUTs are connected as a scan chain.

C. *Overall Test Architecture*

The overall testing procedure is done for horizontal parallel wires and vertical parallel wires in an FPGA. The testing is mainly done for long parallel running wires before they hit a switch box where the crosstalk can easily accumulate. The overall test architecture for a set of six horizontal interconnects is shown in Fig. 5 as an example. The pattern generators in Fig. 5 at the source side of WUTs are the CLBs configured with architecture as shown in Fig. 3 and the analyzers at the destination side of the WUTs are the CLBs configured with architecture as shown in Fig. 4. At the destination side of WUTs the output of the flip-flops from the CLBs configured with the architecture in Fig. 4 is connected in a scan-chain fashion, and the output of the scan-chain is directly fed to the primary output pin of the FPGA. After the testing procedure is done, data from the output pins of the FPGA should be stored in a parameter file and the routing procedure should use the parameter file for any application that is to be routed in the FPGA. If a particular interconnect is affected by crosstalk then using the parameter file that interconnects is avoided for routing the given application by modifying the routing cost function conversely, shielding (inserting VDD or GND between those two signal wires) may be done for those interconnects.

Authorized licensed use limited to: IEEE Xplore. Downloaded on January 20, 2009 at 14:35 from IEEE Xplore. Restrictions apply.

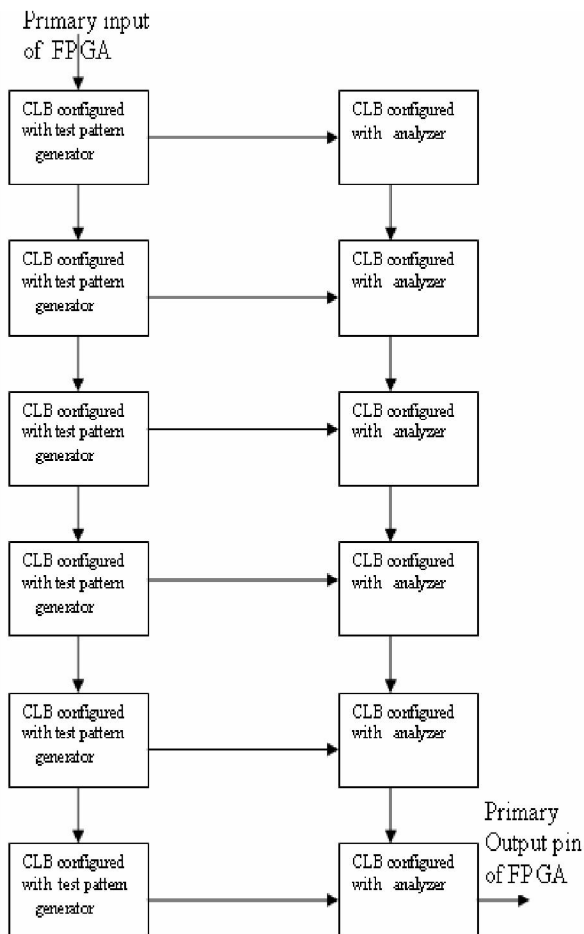


Fig 5. Overall Test architecture for set of interconnects

V. Modified Cost Function

The maze-routing algorithm can be used for routing the nets in an FPGA [7]. In the maze-routing algorithm, the fitness of any segment that might be added to the net is evaluated using the cost function. The cost function is used to obtain the most appropriate path between the CLBs to be connected among all the possible paths.

The parameters are defined as follows:

Cost (i) = Cost for routing a particular segment i
Crit = Criticality of the currently routed net
Delay (i) = Elmore delay of the segment i
b (i) = Base cost of using segment i
h (i) = Historical congestion cost of using segment i
p (i) = Present congestion cost of using segment i

The cost function is defined as

$$\text{Cost}(i) = \text{Crit}*\text{delay}(i)+(1-\text{Crit})*b(i)*h(i)*p(i) \quad (1)$$

The criticality of a net is close to 1 if the net is close to the critical path of the circuit [4]. The Base cost is the basic cost of using a resource. The historical congestion cost is to prevent the nets to be routed to tracks that have led to bad routing solutions in the past. The present congestion cost indicates the congestion cost of using the segment for a given application. In the routing procedure, if the parameter file indicates that a particular interconnect line is affected due to crosstalk noise, then a penalty function (which indicates an increase in the cost of routing a segment) is added to the original cost function. Using the new approach that is proposed for detection of crosstalk noise the cost function for routing the nets in an FPGA can be modified as follows:

$$\text{Cost}(i) = \text{Crit}*\text{delay}(i)+(1-\text{Crit})*b(i)*h(i)*p(i)+\text{Penalty} \quad (2)$$

Where Penalty = 0 if $i \neq j$

$$\text{Penalty} = \sum_{j=0}^{n} (1-\text{Crit})*b(j)*h(i)*p(j) \quad \text{if } i = j \quad (3)$$

In these equations, ‘j’ is the track number or segment from a parameter file obtained after a testing procedure and ‘n’ is the number of times that particular segment appears in a parameter file.

This study draws the following important conclusions from equation 3 that for interconnects affected by the crosstalk noise, the base cost and the present congestion cost of the segment will be increased with the penalty function. This increases the cost of routing particular segment. Using this vital information, the route which carries more probability of being affected by crosstalk may be avoided for the application by the routing algorithm (maze routing algorithm), which essentially depends on the cost function.

VI. Conclusion

This paper proposed a new framework for detecting crosstalk noise among interconnects in FPGAs where the testing is done irrespective of the intended application. The proposed framework presents test architectures for the CLBs to be configured at the source and destination side of WUTs to detect the effects of crosstalk noise. The main advantage of the approach is that the testing architecture includes crosstalk effects such as positive and negative glitches on the victim interconnects along with delays in transition. An improved version of the existing cost function for routing in FPGAs is also presented, which simultaneously incorporates the crosstalk noise effects such as glitches and delays in the same expression. The data from a parameter file (described in section V) are used to modify the cost function for routing any application on an FPGA such that it is unaffected by crosstalk noise. The information obtained from the parameter file can be used by the application configuring process so that the routing can be done with minimal crosstalk effects. The experimental results for the proposed approach to detect crosstalk noise effects in FPGAs are yet to be observed. In future work, this testing procedure will be correlated to the BIST technique usually used to find logical or interconnect faults in FPGAs in a way that the same technique can be used to detect stuck-at faults along with crosstalk faults.

Authorized licensed use limited to: IEEE Xplore. Downloaded on January 20, 2009 at 14:35 from IEEE Xplore. Restrictions apply.

VII. References

[1] Yajun R., Marek–Sadowska, "Crosstalk Noise in FPGAs", *DAC '03,* pp.944-949, 2003.

[2] Nourani M, Attarha A., "Built-In Self-Test for Signal Integrity" DAC'01, pp: 792-797, 2001.

[3] Ahmed N., Tehranipour M., Nourani M., "Extending JTAG for Testing Signal Integrity in SOCs", proceedings of the Design Automation and Test in Europe Conference and Exhibition (DATE'03), 1530- 1591/03, 2003.

[4] Wilton S., "A Crosstalk- Aware Timing- Driven Router for FPGAs", FPGA *'01,* pp.21- 28, 2001.

[5] Bai X., Dey S., Rajski J., "Self – Test Methodology for At-Speed Test of Crosstalk in Chip Interconnects", *DAC, '01,* pp: 619-624, 2000.

[6] Chen W., Gupta S. K., Breuer M. A., "Test Generation in VLSI Circuits for Crosstalk Noise", *Proceedings of IEEE International Test Conference p*p: 641-650, 1998.

[7] Hur W. S., Jaganathan, Lillis J., "Timing Driven maze routing." In pr*oc. Int. Symp. On Physical Design*, Pages 208- 213, 1999.

[8] Smey, M. R., Swartz B., Madden H.P, "Crosstalk Reduction in Area Routing"*, DATE '03,* 2003.

[9] Metra C., Pagano A., Ricco B., "On-line Testing of Transient and Crosstalk Faults Affecting Interconnections of FPGA Implemented Systems", *ITC International Test Conference,* pp:939-947, 2001.

[10] Abramovici M. and Stroud C.,"BIST-Based Detection and Diagnosis of Multiple faults in FPGAs", in *Proc of IEEE Int. Test Conf.,* pp.785- 794,2000.

[11] Chmelar E., "FPGA Interconnect Delay Fault Testing" *ITC International Test Conference,* pp: 1239-1247, 2003.

[12] Cuviello M., Dey S., Bai X., "Fault Modeling and Modeling for Crosstalk in System on Chip Interconnects", proceedings of the International Conference on Computer-Aided Design, pages 297-303, 1999.

[13] Haung W. K., Lombardi F., "An approach for Testing programmable configurable Field Programmable Gate Arrays", in *Proc of IEEE VLSI Symp.*, pp.450-455,1996.

Authorized licensed use limited to: IEEE Xplore. Downloaded on January 20, 2009 at 14:35 from IEEE Xplore. Restrictions apply.