

# Test Pattern Generation for Signal Integrity Faults on Long Interconnects

Amir Attarha  
Advanced DSP Development  
LSI Logic, Corporation  
Plano, TX 75074  
aattarha@lsil.com

Mehrdad Nourani  
Center for Integrated Circuits & Systems  
The Univ. of Texas at Dallas  
Richardson, TX 75083  
nourani@utdallas.edu

## Abstract

*In this paper, we present a test pattern generation algorithm aiming at signal integrity faults on long interconnects. This is achieved by considering the effect of inputs and parasitic RLC elements of the interconnect. To enhance the performance of test generation process, model order reduction methodology is employed. This strategy significantly improves the simulation time with slight loss of accuracy.*

## 1 Introduction

### ■ Motivation

Advance in VLSI technology has played a key role to continuously improve clock frequency and system density. However, it has generated new test challenges, so-called signal integrity, for high speed systems. Signal integrity is defined as any disturbance (noise) which occurs in VLSI systems as the result of undesired parasitic elements or the effects of environment. The frequency is the dominant factor in causing such signal disturbance. Thus, as the speed of SoCs goes higher, we expect to experience more integrity problems in interconnects.

Signal integrity issues arise from long on-chip interconnects where the effect of parasitic elements may jeopardize the functionality and reliability of high performance SoCs. Coupling capacitances, ground capacitances, mutual inductances, self inductances, and wire resistances cause signal disturbance (noise) on long interconnects. Such noise eventually may appear as excessive delay (performance degradation), overshoot/undershoot (reliability drop), and positive or negative glitch (functional error). In this work, we present a test pattern generation method for long interconnects targeting integrity loss. It exploits model order reduction methodology [1] to reduce the computation required for obtaining efficient test patterns.

### ■ Prior Work

Many researchers have addressed the crosstalk issue at the gate-level. Several analyses of crosstalk effects and causes have been reported in [2][3]. Various fault models and test pattern generation strategies for detecting crosstalk effects at gate-level were also presented [4][5]. In [6], the authors presented a pattern generation technique for delay testing and dynamic timing analysis capable of considering the impact of power supply noise on the performance of the circuit. In [7], an on-line detection of realistic failures including crosstalk,

delay, and transient faults was presented.

Designers have discovered that large portion of the signal delay is being attributed to the parasitic effects of the interconnection [8]. Despite the dominant effect of long interconnects on performance and testing, few research groups have investigated the challenges of test pattern generation for long interconnects. Maximum Aggressor (MA) fault model was presented in [9], which abstractly models crosstalk effects on the interconnects with a linear number of faults. Based on the MA fault model a semi-BIST circuitry and a defect simulation method were developed [10]. Although MA significantly simplifies the problem for interconnects modeled as RC circuits, it suffers from lack of precision needed for accurate RLC interconnect models [11]. In [12], the authors presented a BIST-based methodology for testing interconnects for noise and skew in gigahertz SoCs and discuss the adverse effects of overshoot on SoC reliability.

The accuracy of a test pattern generation depends heavily on the interconnect model. Researchers have presented various interconnect models [14]. Since inductance effects on on-chip interconnects have become increasingly important in high frequency, we use the interconnect model similar to [14], in which all capacitive and inductive couplings are considered.

Due to the complexity of the interconnect model, an efficient simulation method is essential for test pattern generation. We have adopted model order reduction methods to alleviate computation complexity with slight loss of accuracy. Model order reduction methods were developed as an alternative for circuit-level simulators to approximate the behavior of long interconnect, power and clock networks [15]. They attempt to approximate a high-order system with a much lower order system with smaller number of poles and zeros (residues), where poles and zeros are the roots of denominator and numerator of the system transfer function, respectively [16]. Generally speaking, the poles specify the behavioral characteristic of the system and the zeros carry the input effects on the system behavior. Using small number of poles and zeros, the simulation can be made significantly efficient with slight loss of accuracy when a large linear interconnect network is replaced by its corresponding reduced order model. Elmore delay model [17], as the first reduced order model, is the most common technique for approximating the delay of RC networks. Elmore model calculates the first time-moment of impulse response of a RC network and correspondingly approximates delay. The first

time-moment can be expressed as the difference of the summations of reciprocal values of poles and zeros, which can be approximated by the reciprocal value of the dominant pole.

Asymptotic Waveform Evaluation (AWE) [18] was introduced as another method based on moment-matching allowing a linear circuit to be analyzed for its dominant poles and corresponding zeros. AWE calculates a certain number of time-moments for the circuit model and then matches these moments to the reduced order model. It was successfully used on the verification of the clock-distribution network on the DEC Alpha chip [18]. Despite its spectacular success, AWE suffers from a number of fundamental numerical limitations. To overcome these numerical limitations of moment-matching methods, researchers have developed reduction methods based on bi-orthogonalization algorithms such as Pade Via Lanczos (PVL) [1] or orthogonalized Krylov subspace methods [15].

In our approach, we use the PVL method [1] to evolve a test pattern generation approach for detecting intermittent failures due to integrity fault (loss) on long interconnects. The PVL method was chosen because of its high accuracy, numerical stability and ability to compute an error bound and identify the true poles and zeros of the original system. The main contribution of our work is in generation of test patterns aimed at signal integrity faults. To the best of our knowledge, this work is the first proposing a systematic method to find test patterns for detecting unacceptable integrity loss.

## 2 Integrity Fault: Concept and Model

True characteristics of a signal is reflected in its waveform. Recent interconnect simulation, design and optimization methods not only consider peak voltage and delay, but also take into account the signal waveform [19] [20]. Informally, integrity of a signal indicates how clean the signal waveform is in terms of delay and level of voltage. In reality, electronic components can tolerate certain level of noise. For example, a CMOS gate interprets any voltage in the  $[V_{Hmin}, V_{dd}]$  range as logic "1" and any voltage in the  $[V_{ss}, V_{Lmax}]$  range as logic "0". Generally, digital circuits are designed to tolerate certain amount of skew delay, i.e.  $T_{SI\_R}$  for rising delay and  $T_{SI\_F}$  for falling delay (see Figure 1).

In practice, circuits have *noise-immune* regions that tolerate certain level of voltage swing and *skew-immune* regions that tolerate certain level of delay. Any portion of signal that exits these immune regions indicates the integrity loss. This concept has been shown graphically in Figure 1 in which the shaded and unshaded (white) strips show the immune and vulnerable regions, respectively. Note that two terms *integrity loss* and *integrity fault* are used interchangeably in this paper.

### 2.1 Integrity fault model

In essence, integrity faults are manifested as voltage and delay violations. The main source of integrity faults is parasitic RLC elements associated with the interconnect network. In addition, manufacturing defects can also contribute in worsening the faulty behavior of interconnect network in terms of signal integrity.

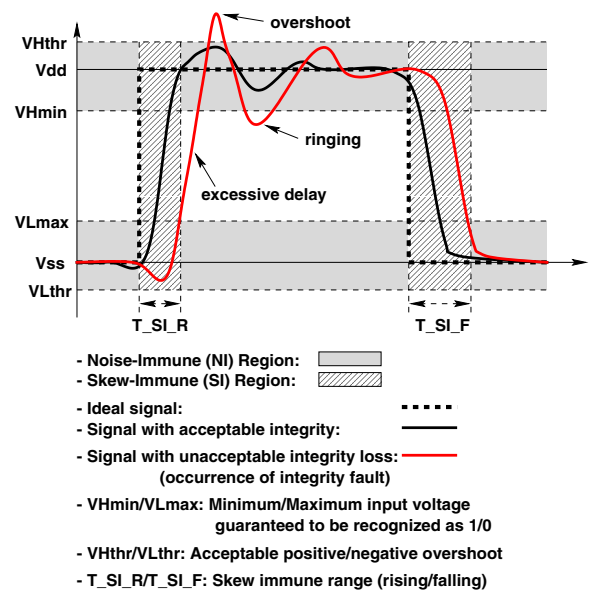


Figure 1. The concept of integrity loss.

Signal integrity faults may not only harm the system functionality (e.g. positive or negative glitches) but also they may deteriorate system's lifetime due to time-dependent dielectric breakdown (TDDB) [21]. More importantly, repeated overshoots are known to inject high-energy electrons and holes (also called *hot-carriers*) into the gate oxide that ultimately cause permanent degradation of MOS transistors' performance and reliability [22].

Conventional test pattern generation algorithms [23] cannot be used for signal integrity faults due fundamental differences between the nature of classic and integrity faults. Classic faults such as stuck-at, bridging, and open faults, are often assumed to cause permanent adverse effects on system in terms of logic level of signals. On the other hand, integrity faults are intermittent, appear mostly as disturbances, heavily depend on the working frequency and are hard to categorize.

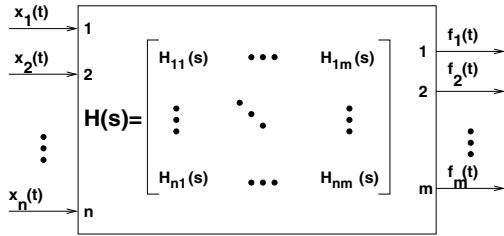
In our method, we consider single defect assumption in test pattern generation process. Any defect locally affects only one interconnect line at a time. It means the behavior of the other interconnect lines in the network will not be affected. With this assumption, we will find the test patterns causing maximal integrity loss in terms of voltage and delay on the interconnect network. When a fault occurs, the adverse effects of integrity loss (e.g. delay or overshoot) are magnified due to the characteristics of the fault. In other words, our patterns are chosen carefully by analysis of fault-free interconnect. However, the selection process guarantees to detect possible integrity faults which deteriorate (violating acceptable delay or overshoot) the behavior of a single interconnect line.

## 3 Interconnect System Representation

In this Section, we present a new formulation for interconnect network which facilitates test pattern generation. The most important novelty of this formulation is the fact that the effects of inputs are parameterized into the behavior of the

**Table 1. Input representation**

$a_i$	$b_i$	$x_i(t)$
-1	0	Falling step
-1	1	Not applicable
0	0	Quiescent at 0
0	1	Quiescent at 1
1	0	Rising step
1	1	Not applicable

**Figure 2. S-domain representation of an interconnect network**

interconnect model. This leads to a unique formula which annotates the effects of input patterns within the behavior of the interconnect network.

### 3.1 Input Representation

In a System-on-Chip (SoC), the interconnect drivers are designed very strong to diminish any possible signal disturbance [24]. Therefore, the transitions on input signals can be reckoned as step functions. On the other hand, some of the signals are quiescent at '1' or '0' depending on the driver circuitry. we define an input  $x_i(t)$  of the interconnect network as:

$$x_i(t) = a_i u(t) + b_i$$

where  $u(t)$  is a step function;  $a_i$  and  $b_i$  are constants such that  $a_i \in \{-1, 0, 1\}$ ,  $b_i \in \{0, 1\}$  and if  $|a_i| = 1$  then  $b_i = 0$ . Table 1 demonstrates how different values of  $a_i$  and  $b_i$  cover all possible transitions on the interconnect line.

### 3.2 System Representation

An interconnect network can be represented by a set of transfer functions in the S-domain [16]. As shown in Figure 2, an interconnect network with  $n$  inputs and  $m$  outputs is represented by transfer function  $H(s)$ .  $H(s)$  itself contains  $n \times m$  partial transfer functions relating inputs to outputs. Note that, in general for an interconnect network, we have  $m \geq n$  due to the possibility of fanout on some wires.

Using the transfer function matrix of the system, the transfer function of a specific output  $f_r$  ( $1 \leq r \leq m$ ) will be:

$$H_{f_r}(s) = \sum_{i=1}^n X_i(s) H_{ir}(s) \quad (1)$$

where  $X_i(s)$  is the Laplace transform of  $x_i(t)$  (the  $i$ th input).

Using an order reduction method (e.g. PVL [1] or ENOR [25]), the transfer function of output  $f_r$  becomes of order  $q$ :

$$H_{f_r}(s) = \sum_{i=1}^n \sum_{j=1}^q \frac{k_{ij}}{(s - p_{ij})} \quad (2)$$

where  $k_{ij}$  and  $p_{ij}$  are zeros and poles of the output  $f_r$ , respectively. Note that  $H_{f_r}(s)$  is obtained based on the superposition of the effects of all  $n$  inputs on  $f_r$ . The output of system can be computed in the frequency domain by multiplying the transfer function and an input function. After transferring it to time domain to observe the timing behavior of the output for  $t > 0$ ,  $f_r(t)$  is calculated:

$$f_r(t) = \sum_{i=1}^n \left( a_i \sum_{j=1}^q \frac{k_{ij}}{p_{ij}} (e^{p_{ij}t} - 1) + b_i \sum_{j=1}^q k_{ij} e^{p_{ij}t} \right) \quad (3)$$

The practical use of Equation 3 is that if poles and zeros are available, for a given set of inputs  $f_r(t)$  can be obtained very quickly using numerical methods. Therefore, it can be used to find patterns that cause maximal delay or overshoots. In providing this equation, we need to estimate  $q$  (the order of model reduction). Determining  $q$  is beyond the scope of this paper. We just point out that it plays an important role in the accuracy of the equation and computational complexity of our method. A key feature of the PVL [1] algorithm is that the accuracy of approximation can be guaranteed for a special frequency range. The authors in [1] have shown that a quality measure ( $Q$ ) for the poles of  $H(s)$  can be defined to determine  $q$ . Calculation of  $Q$  is based on defining bounds for eigenvalues of the reduced model. Finally,  $Q$  is used for determining the optimum order of reduction ( $q$ ) so that to satisfy the desired accuracy. Details can be found in [27][1].

#### 3.2.1 Finding Patterns for Maximal Delay

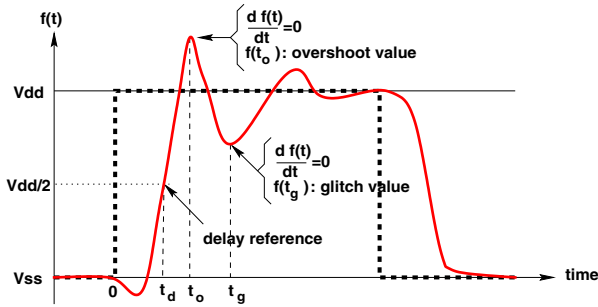
In Equation 3, all the timing information of the output signal is available and the impact of inputs  $x_i(t)$  are considered within  $a_i$  and  $b_i$  coefficients. As a result, by defining delay as the time taken to reach certain voltage (e.g. 50% of  $V_{dd}$  as shown in Figure 3), the delay can be computed from the following equation:

$$f_r(t) - 0.5V_{dd} = 0 \quad (4)$$

Using numerical methods, we can solve Equation 4 to obtain the maximal delay  $t_d$  associated with an input set.

#### 3.2.2 Finding Patterns for Maximal Overshoot

Overshoot occurs when a signal momentarily exceeds  $V_{dd}$ . In other words, overshoot is the global maximum of the output waveform of the system. Finding the global maximum of a nonlinear equation (e.g. Equation 3) is a complicated problem in general. However, overshoot is the maximum voltage value taking place in the neighborhood of the root of Equation 4. More specifically, the neighborhood information (i.e. a point



**Figure 3. Calculating delay and overshoot.**

around  $t_d$ ) has already been obtained by solving Equation 4. The following equation finds when the overshoot happens::

$$\frac{df_r(t)}{dt} = 0 \ni t \in (t_d, t_d + \Delta] \quad (5)$$

By solving Equation 5,  $t_o$  is found and the overshoot value ( $f_r(t_o)$ ) is computed. Figure 3 graphically demonstrates the calculation of overshoot value.

### 3.2.3 Finding Patterns for Glitches

Glitches (ringings) occur when a signal momentarily falls below  $V_{Hmin}$ . Similar to what we explained for overshoots, glitches taking place in the neighborhood of  $t_o$  obtained already by solving Equation 5. The following equation finds when the glitch happens:

$$\frac{df_r(t)}{dt} = 0 \ni t \in (t_o, t_o + \Delta] \quad (6)$$

By solving Equation 6,  $t_g$  is found and the glitch voltage value ( $f_r(t_g)$ ) is computed for further processing. Figure 3 demonstrates this graphically.

### 3.2.4 Numerical Methods to Solve Equations

Theoretically, we are able to analyze the output for any possible indication of signal integrity loss such as delay, overshoot, or undershoot. However, Equations such as 3, 4 and 5 must be solved efficiently. Fortunately, there are several efficient numerical methods for solving nonlinear equations such as Bisection, Secant, and Newton-Raphson [26]. We have selected Newton-Raphson method to solve these Equations. In this method, care must be given to the proper choice of a starting point. Fortunately, in all of our equations we deal with a narrow neighborhood in which the starting point can easily be determined. With a careful selection of the starting point, Newton-Raphson method converges quadratically to the root, as proven in [26].

## 4 Test Pattern Generation Strategy

Interconnects behave as simple conductors at low frequency which can easily be tested for classical fault models such as bridge, open, and stuck-at faults. Nevertheless,

they need to be treated as transmission lines in high frequency which creates new challenges for SoC testing.

To closely monitor the true behavior of interconnect lines, it entails to test them at-speed which requires very expensive ATEs. Due to the ATE time/cost expenses, even for limited number of interconnect lines routed closely, applying exhaustive test patterns is not justified. Another approach which does not carry high test pattern generation cost is random pattern strategy. Although random patterns can be generated easily, they eventually prolong test session to achieve acceptable level of testing. Furthermore, weighted random patterns intended to detect signal integrity faults is equally challenging [11]. Large number of parasitic elements, irregularity of the routed wires, and contribution of working frequency make it difficult to derive some common properties in interconnect to define weights and design a reasonable cost hardware [13]. Therefore, an efficient test patterns causing the worst case of signal integrity loss are essential for decreasing test cost and enhancing the test quality of high speed SoCs.

### 4.1 Test Pattern Generation Algorithm

Considering the property of the interconnect network and the effect of parasitic elements, the search space of possible patterns can be restricted without decreasing the chance of finding effective patterns. To be more specific, we need to elaborate on the roles of different parasitic elements in the behavior of an interconnect network.

In spite of coupling capacitances between all wires, the effect of capacitive coupling is considered local, in the sense that the coupling effects of adjacent wires are quite dominant compared to the capacitive coupling effects of far off wires [14]. However, the inductance has larger range effect and thus the effect of mutual inductance could be significant. Furthermore, the effect of coupling inductances and capacitances on a wire oppose each other [14]. When the signal on a wire switches in one direction, the noise due to capacitive coupling affects other nearby signals in the same direction as that of switching. However, the noise due to inductive coupling is in the opposite direction.

To be systematic in limiting the search space, we define the *locality* metric. Locality  $k$  is a parameter showing the proximity of neighboring wires in each side that actively affect one wire due to coupling capacitances and inductances. While larger  $k$  increases the accuracy, the simulation and pattern generation will require longer search. If the interconnect network contains  $n$  input lines, the total number of patterns (search space, also called *candidate set*) will reduce from  $4 * 4^n$  to  $4 * 4^{n-2k-1}$ , where 4 reflects all possible states for an input (i.e. falling, rising, quiescent at 0, and quiescent at 1). Note that depending upon the targeted integrity fault (overshoot/delay), the input to a specific line subject to fault will be determined. We can further restrict the search space (candidate set) of patterns by exploiting the fact that the effect of mutual inductances oppose coupling capacitances.

For example, when more topographical and layout information of interconnects is available, we can determine the dom-

```

TPG ()
Input: 1) Interconnect information extracted from layout
        2) Acceptable delay ( $T_d$ ), overshoot ( $V_o$ ) and glitch ( $V_g$ )
        3) Locality number ( $k$ ).
Output: Test pattern set  $P = P_d \cup P_o \cup P_g$  for integrity faults.

01: Provide the candidate set using  $k$ 
02: Select the appropriate  $q$ 
03: Perform model order reduction
04:  $P_d = \{\}, P_o = \{\}, P_g = \{\}, P = \{\}$ 
05: for ( $r=1$  to  $n$ ) do
06: {
07:     Find neighbors and candidate set for the  $r$ th line
08:     Form  $f_r(t)$  for numerical analysis
09:     for (pattern  $p$  in candidate set) do
10:     {
11:         Calculate integrity loss by using Equations 3, 4, 5, 6
12:         If (delay >  $T_d$ )
13:              $P_d = P_d \cup \{p\}$ 
14:         If (overshoot >  $V_o$ )
15:              $P_o = P_o \cup \{p\}$ 
16:         If (glitch <  $V_g$ )
17:              $P_g = P_g \cup \{p\}$ 
18:     }
19: }
20: return  $P = P_d \cup P_o \cup P_g$ 

```

**Figure 4. Test pattern generation algorithm.**

inancy of coupling capacitances versus mutual inductances. Then, the patterns which stimulate only one of those two groups of parasitic elements become of more interest. Such information and choice can be reflected in the locality factor  $k$  such that eventually the total number of patterns will be reduced to:  $(n - 2k) * 4^{(n-2k-1)} + k \sum_{i=0}^{k-1} 4^{(n-2k+i)}$ , where, the summation term covers the asymmetry of  $k$  lines at the sides of interconnect network. Briefly, depending on the available information, test time budget and desired accuracy the candidate set of patterns will be chosen. Eventually, the test pattern generation algorithm will select a very small subset of the candidate set.

For generating the test patterns, accurate interconnect model is required which can be obtained using extraction tools such as NETAN [28]. Figure 4 describes our test pattern generation algorithm. First, the locality factor  $k$  and the order of model order reduction  $q$  need to be decided. Then, we use the PVL algorithm [1] to get the reduced order model and find the poles and zeros. For a selected line, Equation 3 is formed. Then, the integrity loss regarding is calculated for an input pattern chosen from the candidate set. If the pattern causes the integrity loss which exceeds the acceptable range, the pattern will be stored in set  $P_d$  (test for delay) or set  $P_o$  (test for overshoot), it will be selected. Otherwise another pattern will be tried until the whole candidate set is exhausted.

## 5 Simulation Results

The accuracy of our method heavily depends on the model order reduction accuracy. In this section, first we use a small interconnect network (to be solved by SPICE quickly) to show that our method has sufficient accuracy compared to SPICE. Then, we exercise a practical interconnect network.

**Table 2. Comparing SPICE and our method.**

Test Patterns	SPICE		Order Reduction	
	Delay [ps]	O.S. [Volt]	Delay [ps]	O.S. [Volt]
00 → 01	32.43	2.53	31.97	2.59
00 → 11	39.98	2.65	38.62	2.64
01 → 10	44.65	2.89	45.06	2.91

**Table 3. Simulation of integrity fault – delay**

Line	Avg. Iterations	Delay [ps]
1	424.6	25.4
2	320.0	32.9
3	129.0	25.8
4	106.3	31.3
5	13.7	52.7
6	276.3	33.5
7	180.3	41.5
8	168.3	28.3
9	1000	Not Found

As a small interconnect, two parallel lines with 2 mm length in 0.35  $\mu\text{m}$  technology are considered. Each of them are modeled by RLC segments and coupling capacitances. The maximum delay and peak overshoot (O.S.) are reported in Table 2 for three arbitrary transitions. As this table shows, our method identifies the same test patterns to stimulate the maximal integrity loss as SPICE finds but runs much faster. The running time for each run for SPICE and our order reduction method are 410 and 3 seconds (wall clock time on SPARC ULTRA 10 with 64 MByte RAM), respectively. MATLAB [29] was used for generating the reduced order model and performing analytical computations.

For more practical experimentation, we have used an interconnect network containing nine parallel lines with RLC segments, coupling capacitances, and mutual inductances. Before starting the test pattern generation, we need to restrict the search space of possible test patterns for signal integrity faults. For instance, if maximum delay of  $n = 9$  interconnect lines are targeted, the total number of possible patterns is equal to  $9 * (4^9) = 2359296$  as we explained in previous section. While if we consider locality factor of  $k = 2$ , as described in Section 4, the number of patterns in candidate set will reduce to  $(5 * 4^4) + (2 * 4^5) + (2 * 4^6) = 11520$ .

After defining  $k$ , the test pattern generation procedure is used.  $q$  is calculated along with the model order reduction. In our experimentation the selected order  $q = 14$ , while the original model is order of 100 since it contains more than 300 memory elements such as capacitances and inductances. The acceptable delay and overshoot are  $T_d = 25\text{ps}$  and  $V_o = 2.62\text{volt}$ , respectively. The frequency and the source voltage are  $f = 1\text{GHZ}$  and  $V_{dd} = 2.5\text{V}$ .

Table 3 and 4 demonstrate the simulation results for capturing time-dependent and voltage-dependent integrity faults, respectively. As seen, the average number of MATLAB iterations to solve the equations numerically vary depending upon the layout information extracted and position of the line. The

**Table 4. Simulation of integrity fault – overshoot**

Line	Avg. Iterations	Overshoot[Volt]
1	208.7	2.76
2	132.3	2.69
3	159.3	2.91
4	84.4	3.04
5	61.0	3.19
6	131.7	2.73
7	70.7	2.85
8	365.4	2.63
9	330.7	2.71

lines in the middle are more subject to coupling capacitance and mutual inductance interferences compared to the lines at (or close to) the side of interconnect network. Therefore, as a general trend, more efforts are required to find the desired test pattern at (or close to) sides because the candidate set is smaller. For instance, as Table 3 shows, MATLAB finds the test patterns for line 5 (in the middle) after only 14 iterations on the average while 424 iterations on average were needed to find the test patterns for line 1. There might be also some cases that no pattern can exceed  $T_d$ ,  $V_o$  or  $V_g$ . We stopped MATLAB after 1000 iterations.

### Acknowledgements

This work was supported in part by the National Science Foundation CAREER Award #CCR-0130513.

### References

- [1] Peter Feldmann, and Ronal W. Freund, "Efficient Linear Circuit Analysis by Pade Approximation via the Lanczos Process," *IEEE Trans. on CAD*, vol. 14, no. 5, pp. 639-649, May 1995.
- [2] S. Voranantakul, J. Prince, and P. Hsu, "Crosstalk Analysis for High-speed Pulse Propagation in Lossy Electrical Interconnections", *IEEE Trans. Components, Hybrids, and Manufacturing Tech.*, vol. 16, no. 1, pp. 127-136, 1993.
- [3] Y. Eo, W. Eisenstadt, J. Jeong, and O. Kwon, "A New On-Chip Interconnect Crosstalk Model and Experimental Verification for CMOS VLSI Circuit Design," *IEEE Trans. On Electron Devices*, vol. 47, no. 1, Jan. pp. 129-140, 2000.
- [4] W. Chen, S. Gupta and M. Breuer, "Test Generation in VLSI Circuits for Crosstalk Noise," in Proc. *Intern. Test Conf. (ITC'98)*, 1998, pp. 641-650.
- [5] K. Lee, C. Norquittst, and J. Abraham, "Automatic Test Pattern Generation for Crosstalk Glitches in Digital Circuits," in Proc. *VLSI Test Symposium*, pp. 34-39, 1998.
- [6] A. Krstic, Y. Jiang and K. cheng, "Pattern Generation for Delay Testing and Dynamic timing Analysis Considering Power-Supply Noise Effects," *IEEE Trans. on CAD*, vol. 20, no. 3, pp. 416-425, March 2001.
- [7] C. Metra, M. Favalli, and B. Ricco, "On-Line Detection of Logic Errors due to Crosstalk, Delay and Transient Faults," *Int. Test Conf.* pp. 524-533, 1998.
- [8] C. L. Ratzlaff and Lawrence T. Pillage, "RICE: Rapid Interconnect Circuit Evaluation Using AWE," *IEEE Trans. on CAD*, vol. 13, no. 6, pp. 763-776, June 1994.
- [9] M. CuvIELLO, S. Dey, X. Bai and Y. Zhao, "Fault Modeling and Simulation for Crosstalk in System-on-Chip Interconnects," in Proc. *Intern. Conf. on Computer Aided Design (ICCAD'99)*, 1999, pp. 297-303.
- [10] X. Bai, S. Dey and J. Rajski, "Self-Test Methodology for At-Speed Test of Crosstalk in Chip Interconnects," in Proc. *Design Automation Conf. (DAC'00)*, 2000, pp. 619-624.
- [11] M. Nourani and A. Attarha, "Built-In Self-Test for Signal Integrity," in Proc. *Design Automation Conf. (DAC'01)*, June 2001.
- [12] Amir Attarha and Mehrdad Nourani, "Testing Interconnects for Noise and Skew in Gigahertz SoCs," in Proc. *Intern. Test Conf. (ITC'01)*, pp. 305-314, 2001.
- [13] P. Chang, B. Keller and S. Paliwal, "Design and Implementation of a Parallel Weighted Random Pattern and Logic Built in Self Test Algorithm," in the Proceedings of the 1999 IEEE International Conference on Computer Design, pp. 238-243, 1999.
- [14] C. Cheng, J. Lillis, S. Lin and N. Chang, *Interconnect Analysis and Synthesis*, John Wiley & Sons, 2000.
- [15] A. Odabasioglu, M. Celik, and L. Pileggi, "PRIMA: Passive Reduced-Order Interconnect Macromodeling Algorithm," *IEEE Conf. on CAD*, 1997.
- [16] Alan V. Oppenheim, *Signals and systems*, second Edition, Prentice Hall, 1997.
- [17] W. C. Elmore, "The Transient Response of Damped Linear Networks With Particular Regard to Wideband Amplifiers," *J. of Appl. Phys.*, vol. 19, no. 1, pp. 155-163, 1948.
- [18] Lawrence T. Pillage, and Ronald A. Rohrer, "Asymptotic Waveform Evaluation for Timing Analysis," *IEEE Trans. on CAD*, vol. 9, no 4, pp. 352-366, April 1990.
- [19] J. Cong and L. He, "Theory and Algorithm of Local-Refinement-Based Optimization with Application to Device and Interconnect Sizing," *IEEE Trans. on CAD*, vol. 18, no. 4, pp. 406-420, April 1999.
- [20] P. Restle, A. Tuehli and S. Walker, "Dealing with Inductance in High-Speed Chip Design," in Proc. *Design Automation Conf. (DAC'99)*, pp. 904-909, 1999.
- [21] W. Hunter, "The Statistical Dependence of Oxide Failure Rates on Vdd and Tox Variations with Applications to Process Design, Circuit Design and End Use," in Proc. *Reliability Physics Symposium*, pp. 72-81,
- [22] P. Fang, J. Tao, J. Chen and C. Hu, "Design in Hot Carrier Reliability For High Performance Logic Applications," in Proc. *IEEE Custom Integrated Circuits Conf.*, pp. 25.1.1-25.1.7, Oct. 1998.
- [23] Abramovici, M. Breuer and A. Friedman, *Digital Systems Testing and Testable Design*, Computer Science Press, 1990.
- [24] J. Rabaey, *Digital Integrated Circuits*, Prentice Hall, 1996.
- [25] Bernard N. Sheehan, "ENOR: Model Order Reduction of RLC Circuits Using Nodal Equations for Efficient Factorization," *Design Automation Conference*, pp. 17-21, 1999.
- [26] E. W. Cheney, *Numerical Mathematics and Computing*, Brooks Cole publishing Company, 1985
- [27] Peter Feldmann, and Ronal W. Freund, "Small-signal Circuit Analysis and Sensitivity Computations with the PVL Algorithm," *IEEE Trans. on Circuits and systems II: Analog and Digital Signal Processing*, vol. 43, no. 8, pp. 577-585, Aug. 1996.
- [28] NET-AN: Multi-net three-dimensional field solver extraction tool, Users reference manual, OEA International, Inc., 2001.
- [29] MATLAB User's and reference manual, 2000 *The MathWorks, Inc., Version 6.0.0.88*, 2000.