

# Programmable Logic BIST for At-speed Test

Yu Huang

*Mentor Graphics Corp.  
300 Nickerson Road  
Marlborough, MA 01752  
yu\_huang@mentor.com*

Xijiang Lin

*Mentor Graphics Corp.  
8005 SW Boeckman Rd  
Wilsonville, OR 97068  
xijiang\_lin@mentor.com*

## Abstract

*In this paper, we propose a novel programmable logic BIST controller that can facilitate at-speed test for the design with multiple clock domains and multiple clock frequencies. Moreover, a static analysis method is also proposed to optimize the BIST test pattern allocation for testing the timing faults in different intra/inter clock domains when the maximum number of applied BIST test patterns is specified. Experimental results show the effectiveness of the proposed method on achieving higher test coverage than the method with test patterns evenly distributed among different test sessions.*

## 1. Introduction

The scan based at-speed testing has become a very important means to detect timing defects that cannot be ignored at 90nm technologies and below [1]-[4]. When doing test pattern generation for the delay defects, two types of fault models, transition fault model and path-delay fault model, have been used extensively in industry. To catch delay defects during test application by using ATE, it is important to apply the launch and capture clocks in the same frequencies as in the normal functional operations. When the at-speed clocks are driven directly from ATE, a couple of problems arise [3]:

(1) To reduce the percentage of test cost over manufacturing cost, older testers or lower-cost testers with lower-limited clock frequencies may be still in use to test high-speed designs.

(2) At frequencies above 100 MHz, it becomes more and more difficult to deal with the interactions between tester skew, pad delays, and internal-clock insertion delays.

To solve the problems mentioned above, programmable phased-locked loop (PLL) embedded in the design has been used to provide the at-speed test clocks during test application [4][5][6]. When applying logic Build-in-Self-Test (BIST) [7]-[10], it may not have the same problems since all the at-speed clocks are typically generated internally by the BIST controller and ATE only provides the control data with slow clocks.

Even if BIST controller clock was provided by ATE, it can use internal clock multipliers to get internal clocks with higher speed. However, for designs with multiple clock domains and multiple clock frequencies, it requires BIST controller to be capable of running different capture clock sequences to detect the defects in different intra/inter clock domains. Typically, BIST controller is customized and hardwired for a set of pre-defined capture clock sequences [11] such that it limits the flexibility for applying additional capture clock sequences to achieve better defect coverage.

It is normal that a design and its later revisions stay long enough to be manufactured with different technology nodes. With rapid changes in technology, it becomes extremely difficult to predict the critical paths that are necessary to be tested. It is known that wiring delay exceeds gate delay at 180 nm and below in aluminum processes. If using copper processes, wiring delay exceeds gate delay at 130 nm and below. At 90 nm, wiring delay will account for about 75% of the overall delay [12]. Therefore, with the changing technology nodes, new critical paths with new capture clock sequences are expected to emerge. Under such circumstances, capture clock sequences that would work well for a certain technology for which the controller was designed may no longer work when manufactured using the new technology node. Consequently, there is an increasing need for a programmable logic BIST solution that would support flexibility to modify the capture clock sequences even after the BIST logic insertion and synthesis are done.

Moreover, at 90 nm and below, performance of a VLSI is unknown prior to the detailed routing, which makes the traditional pre-layout timing analysis methods such as simulation based technologies, static timing analysis and emulation systems inaccurate. Although some post-layout timing analysis tools are available in the industry, the problem of finding out all true critical paths cannot be completely solved at pre-silicon phase because the problem mentioned above is further exacerbated by signal integrity problems, design integrity issues and process variations. The signal integrity issues

include crosstalk, IR drop, power and ground bounce, etc. The design integrity issues include electron migration, hot electron, wire self-heating, etc. In this scenario, it is likely that some true critical paths may not be targeted for test. From our industrial practices, we did see some designs passed all stuck-at and at-speed tests, but failed in system at some specific functional clock sequences. If the BIST controller is programmable, it will help silicon debug process by adding new capture clock sequence that caused the failure.

In [13], a programmable Test Waveform Generator (TWG) driven by on-chip PLL is proposed to improve the flexibility of generating capture clock sequences. For each functional clock, a TWG including 2 shift-registers is used to create the desired clock waveforms. Before applying next test pattern, the shift-registers must be initialized with control data through scan load. In this paper we propose a novel 2-dimensional scan programmable logic BIST architecture that can generate the desired at-speed capture clock sequences for multiple clocks. For each test session, i.e., applying  $N$  BIST test patterns with the same capture clock sequence continuously, the programmable controller only need to be initialized once.

The remainder of the paper is organized as follows. In Section 2, we review how a traditional logic BIST controller generates at-speed clocks. In Section 3, we propose a programmable logic BIST controller that can generate any specified capture clock sequences. We also illustrate how to extend the proposed method to utilize the embedded PLL for at-speed testing. In section 4, we propose an effective test pattern allocation algorithm based on static analysis. The proposed algorithm optimizes the test pattern count allocated for each test session with a unique capture clock sequence in order to achieve high test coverage when the maximum number of applied BIST test pattern count is specified. Section 5 gives experimental results for several industrial circuits and Section 6 concludes the paper.

## 2. Traditional Logic BIST Controller

Traditional logic BIST controller [14] implements a generic clock gating mechanism for controlling clocks driving the core design under test. One typical example illustrated in Figure 1 shows a conceptual view of a logic BIST controller that drives three clock domains ( $clk_1$ ,  $clk_2$  and  $clk_3$ ) in the design. During normal functional operations,  $bist\_run$  is turned off such that the original clocks drive the core design through the MUXes directly. During logic BIST testing,  $bist\_run$  is turned on and the clocks from logic BIST controller are used to drive the core design. When  $scan\_enable$  is on, the slow  $shift\_clock$  drives the clocks of the core design to shift-in/shift-out the test stimuli/test responses to/from scan chains. The  $shift\_clock$  is derived from the BIST controller by using a clock divider. When  $scan\_enable$  is off the circuit goes into capture mode and the core design

is driven by the waveform generator that produces capture windows including different capture clock sequences based on three BIST capture clocks,  $bist\_cap\_clk_1$ ,  $bist\_cap\_clk_2$ , and  $bist\_cap\_clk_3$ . All three BIST capture clocks are derived from the same reference clock,  $bist\_clk$ , which is in turn derived from the fastest top-level clock input.

To test the designs with multiple clock domains and multiple clock frequencies, users can define multiple capture windows, and generate both fast and slow clock waveforms within the capture windows by controlling the waveform generator. The capture clock sequence shown in Figure 2 gives an example of using a capture window to test at-speed faults, including intra-clock domain paths and inter-clock domain paths. Typically, the waveform generator is hardwired to produce a set of predefined capture clock sequences and it is not reconfigurable.

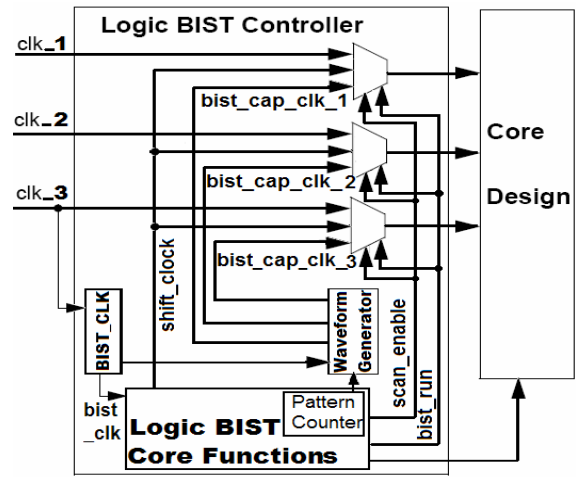


Figure 1: Traditional logic BIST controller

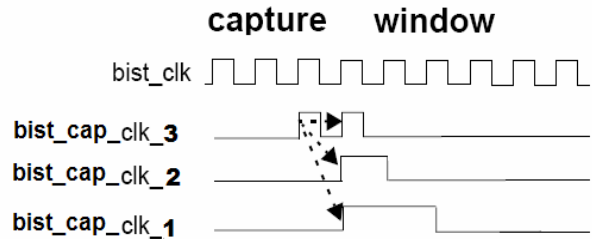


Figure 2: Example of a capturing window

The *Pattern Counter* block shown in Figure 1 is used to control number of BIST patterns applied at each test session. When the pattern counter reaches the pattern limit of a test session, the BIST controller starts a new test session with a new capture clock sequence. The number of test sessions and the pattern counter size are typically predetermined before synthesizing BIST controller. In general, they are unable to be changed after the synthesis.

### 3. Proposed Logic BIST Architecture

In this section, we described a programmable waveform generator (PWG) to produce any complex clock waveforms for at-speed test.

#### 3.1. Programmable Waveform Generator

The structure of PWG is shown in Figure 3. It is composed of a control block (the left box) and a 2-dimensional (2-D) scan register bank (the right box). The control block has four input signals, `test_session_setup`, `shift_clock`, `bist_clk` and `scan_enable`. The last 3 signals are the same as the traditional BIST controller, while `test_session_setup` is used to load the programmable control data into the 2-D scan register bank before a test session starts. The three output signals,  $Clk_d$ ,  $Clk_s$  and `Sel` of the control block are used to control the various operations of the 2-D scan register bank, which will be explained later. The 2-D scan register bank has  $n$  rows and  $k$  columns, which can generate any complex clock waveforms for  $k$  clocks. The number of rows and the number of columns of a 2-D scan register bank are called *depth* and *width*, respectively. The depth of a 2-D scan register bank determines the maximum frequency range for all active clocks in the capture window and the maximum number of pulses for each clock. The width of a 2-D scan register will determine the maximum number of active clocks in all the capture windows. The details about how to select the depth and the width when designing a 2-D scan register bank will be given later.

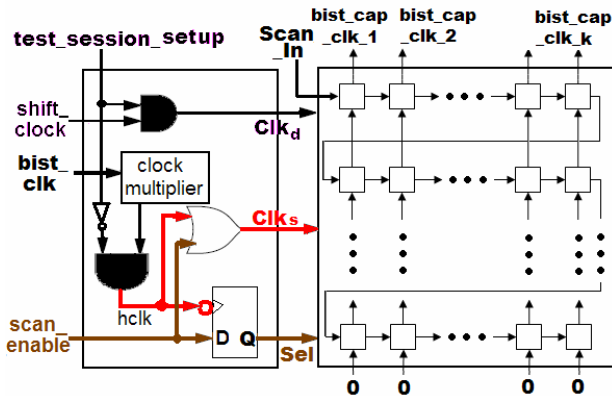


Figure 3: Programmable waveform generator

The 2-D scan register bank is composed of a set of 2-D scan cells, represented by the small rectangles in Figure 3. The structure of a 2-D scan cell and its connections with the control signals from the control block are shown in Figure 4. Here, we use cell  $(i, j)$  to represent the 2-D scan cell at the  $i^{\text{th}}$  row and the  $j^{\text{th}}$  column in a 2-D scan register bank. As shown in Figure 4, a 2-D scan cell includes two flip-flops, DFF and its shadow SDFD. DFF is used to store the waveform control data for each test session while SDFD is used to apply the capture clock sequence for each BIST pattern.

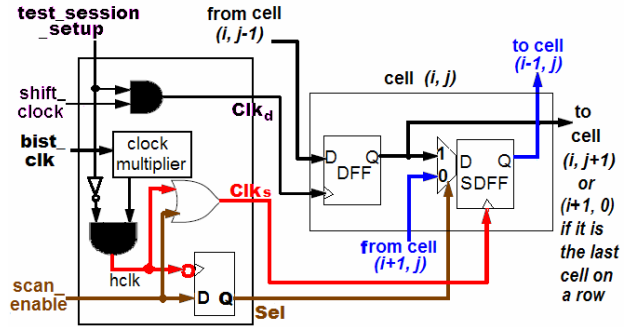


Figure 4: Structure of a 2-D scan cell

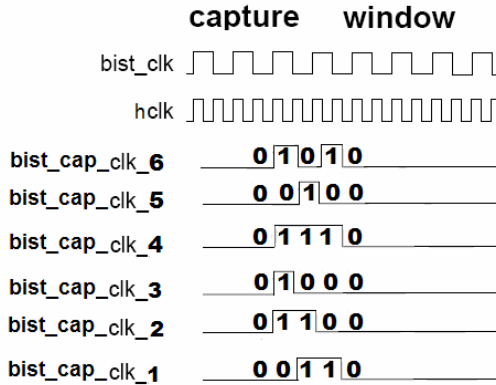
At the beginning of a new test session, the PWG is reprogrammable by loading the control data generating desirable capture clock sequence into the DFFs of every 2-D scan cell. To shift the control data into the DFFs through `Scan_in`, `test_session_setup` is asserted to make  $Clk_d$  driven by `shift_clock`.  $Clk_d$  is used to shift the control data into the DFFs. The `Scan_in` pin can be controlled by ATE, boundary scan or even an on-chip flash/EPROM that stores the clock waveforms. Once the control data is loaded, `test_session_setup` is de-asserted to start the test session. The control data stored in the DFFs will not be changed in current test session.

The clock multiplier block doubles the frequency of `bist_clk`. Its output drives `hclk` when `test_session_setup` is de-asserted. At the end of loading next BIST pattern into the scan chains, `scan_enable` is de-asserted and `hclk` drives  $Clk_s$  to update the value at SDFDs during capture. Right after `scan_enable` is de-asserted, `Sel` is 1 and the data stored in DFFs are copied into SDFDs at the leading edge of the next `hclk`. At the trailing edge of the `hclk` in the same cycle, `Sel` becomes 0 such that the SDFDs at the  $i^{\text{th}}$  row of the 2-D scan register bank are able to be updated by the values stored in the  $(i+1)^{\text{th}}$  row in the following `hclk` clock cycles. After  $m$  `hclk` clock cycles, all the SDFDs will have value 0, where  $m$  is the depth of the 2-D scan register bank, and the responses captured into the scan cells are unloaded into MISR after `scan_enable` is asserted. Above steps are repeated for every BIST pattern in the current test session. The copy operation from DFFs to SDFDs guarantees the same capture clock sequence is used for every BIST pattern in the current test session. The test session stops when the pattern counter reaches to a predefined limit and the `test_session_setup` is asserted to program the PWG for the next test session.

An example of capture window by using the proposed PWG is shown in Figure 5. To produce the capture clock sequence shown in Figure 5, the size of the 2-D scan register bank is  $5 \times 6$ . The control data loaded into the 2-D scan register bank is also shown in Figure 5.

As mentioned earlier, the width of a 2-D scan register bank is determined by the maximum number of clock domains that are required to be active during

capture and the depth of the 2-D scan register bank is determined by the number of slowest clock cycles to be applied in the capture window. For example, let us assume that `bist_clk` is 200MHz, the slowest clock is 50MHz, and the capture window should be able to apply two consecutive slowest clock pulses. To generate two consecutive pulses of 50MHz, it requires  $(1+8+8)=17$  waveform bits to be shifted out by using 400MHz `hclk` clock. That is to say the depth is at least 17 in this example. Note that the first waveform bit is added to set all the capture clocks in off-state at the beginning of the capture window. Since the data input of the SDFF at the last row is driven by constant 0, the capture clock will be returned to the off state at the end of the capture window and no additional waveform bit needs to be added.



**Figure 5: Capture clock sequence generated by 5x6 2-D scan register bank**

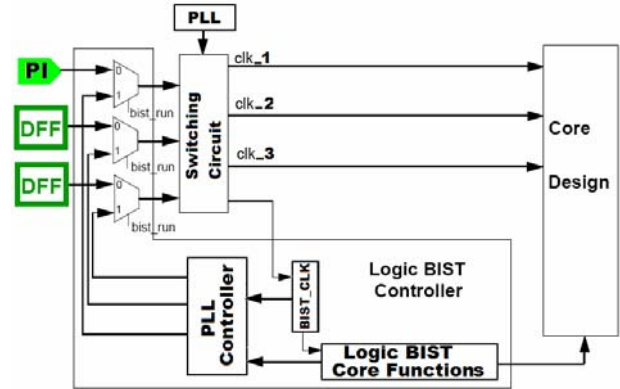
To program the multiple clocks with big frequency difference, a large depth has to be used when designing the 2-D scan register bank, which may incur large area overhead. If the design includes on-chip PLLs, the BIST controller may take the advantage of them to reduce the area overhead and it is described next.

### 3.2. Programmable PLL

In modern designs, on-chip PLLs are widely utilized for high speed functional operations. To make the at-speed testing more "true-to-life", the same clock paths and timing as the normal functional operations should be exercised. As a result, the PLL switching circuits are designed to be programmable by controlling some registers and/or PIs in order to facilitate with at-speed testing [6]. When using logic BIST to do at-speed testing, it is better to utilize on-chip PLLs to provide the at-speed clocks rather than generating them by BIST controller itself. The PWG proposed in the previous sub-section can be used to provide the control data for the PLL switching circuit. If the number of PLL controlling bits is smaller than the total number of clocks generated from PLL, it requires smaller width when designing the 2-D scan register bank. Similarly, if the number of clock cycles to control the PLL switching circuit is less than

the length of waveform bits to generate consecutive pulses of the slowest clocks, it requires smaller depth when designing the 2-D scan register bank.

The structure of using 2-D scan register bank to control PLL switching circuit during BIST run is shown in Figure 6. In this example we assume the PLL switching circuit is controlled by 3 bits, one from PI and the other two from DFFs.



**Figure 6: Directly control programmable PLL**

As shown in Figure 6, three MUXes are inserted between the PLL controlling bits and the PLL switching circuit. When `bist_run` is asserted, PLL controller takes over the control of PLL switching circuit and the PLL control bits are driven by the outputs of the 2-D scan register bank embedded in the PLL controller block. Note that the `bist_clk` is derived from PLL. It is not generated by BIST controller locally. To produce a desired capture clock sequence, the PLL control data is loaded into 2-D scan register bank before starting a BIST test session.

When designing the 2-D scan register bank, its depth is determined by the sequential depth to control the original PLL control bits during capture and its width is determined by the number of original PLL controlling bits.

To choose the BIST capture clock generation scheme between the two methods proposed in this sub-section and in the previous sub-section, we may estimate the area overhead for both schemes by calculating the number of 2-D scan cells included in the 2-D scan register bank. The scheme with less area overhead is selected. When the area overhead is similar for both schemes, directly controlling PLL is always preferable since it does not introduce additional delays in the existing functional clock tree.

## 4. Test Pattern Allocation for the Design with Multiple Clock Domains

When using logic BIST to do the self-test, the maximum number of test patterns to be applied is typically determined in advance in order to restrict the

test application time as well as to design the test pattern counter embedded in the logic BIST controller. For the design with multiple clock domains, it requires running the at-speed test for each intra clock domain and each inter clock domain to achieve the best test quality for delay defects. The problem we are facing is how to allocate limited number of test patterns to test intra/inter clock domains in order to achieve maximal test coverage. The simplest solution is to distribute the total test pattern count evenly among all test sessions. The total number of test sessions is determined by all possible capture clock sequences. In general, the number of faults in different intra/inter clock domains is not distributed uniformly. To achieve higher test coverage, the clock domain with larger number of faults typically requires more test patterns. Finding the optimal solution for the test pattern allocation problem is a NP-hard problem. We proposed a heuristic next to solve the test pattern allocation problem for the transition faults. The same heuristic can be extended to other fault models in a straightforward way.

Let us assume that all the capture clock sequences to test different inter/intra clock domains are already determined and  $TP$  denotes the maximum number test patterns to be applied during entire BIST run. We use the following procedure to calculate the test pattern count for each test session.

**Procedure *Test\_Pattern\_Allocation*( $TP$ )**

1. For each clock sequence  $CS_i$ , do static analysis to calculate the number of faults  $NF_i$  that can be potentially detected by  $CS_i$ .
  - (a) Create an ATPG window by implying the capture clock sequence  $CS_i$ . The size of the ATPG window is equal to the length of  $CS_i$ .
  - (b) Starting from all the observation points in the last frame of the ATPG window, traverse each frame in the ATPG window backward in order to mark the gates that are potentially observed.
  - (c) For each frame in the ATPG window, identify the primary inputs and the pseudo primary inputs that can potentially create a transition when moving time frame from the previous frame to the current frame. Starting from all the identified primary inputs and pseudo primary outputs, traverse circuit forward in order to mark all the combinational gates in the current frame that can potentially launch a transition.
  - (d) If a transition fault at any frame in the ATPG window is in the intersection cones marked by Steps (b) and (c), the fault is potentially detectable by  $CS_i$ .
2. Let  $N = \sum_{i=1}^{N_{CS}} NF_i$ , where  $N_{CS}$  is the total number of capture clock sequences.
3. For each capture clock sequence  $CS_i$ , the number of test patterns allocated for it is calculated as follows:

$$TP_{CS_i} = \frac{NF_i * TP}{N}.$$

During test application, each BIST test session is applied sequentially. If the signature stored at MISR is unloaded at the end of applying a test session, a defective chip may be identified earlier if the capture clock sequences are applied in the decreasing order of the number of faults detected by each capture clock sequence. This is because the clock domain with larger number of faults has higher probability of introducing defects during manufacturing. Meanwhile, to speed-up fault simulation, we apply the following two techniques:

- Run fault simulation for different clock sequences according to the decreasing order of the number of potentially detected transition faults. This may help to drop faults that are possible to be detected by multiple capture clock sequences earlier from the fault list.
- When running fault simulation for a capture clock sequence, we only consider the faults that are potentially detected by this particular clock sequence.

## 5. Experimental Results

We apply the proposed test pattern allocation method to six industry circuits and the experimental results are shown in Table 1. In Table 1, following the circuit names, we give the number of transition faults and number of clocks in each circuit under the column *#Flts* and *# Clks*. When selecting the capture clock sequences to test different clock domains, we restrict our consideration on the clock sequences that pulse the same clock continuously for two clock cycles. The clock sequences that test the faults in the inter clock domains are not considered in this experiment since majority faults are within the intra clock domains.

Three BIST runs are simulated with the maximum number of test pattern count set to be 512K, 768K, and 1024K, respectively. For each BIST run, two test pattern allocation methods are used, one is based on the proposed procedure and the other uses the even distribution method. The test coverage achieved by each method is shown under the sub-columns *Opt.* and *Even.*, respectively. For all the circuits under the experiments, the proposed test pattern allocation method achieves higher test coverage than the even distribution method when simulating the same number of BIST test patterns. For example, more than 3% of test coverage is achieved by the proposed method for ckt1 with 512K BIST test patterns. Among four out of six circuits, the proposed method with 512K BIST patterns can achieve higher test coverage than the even distribution method with 1024K BIST test patterns. The experimental results shown in Table 1 demonstrate the effectiveness of the proposed test pattern allocation method.

**Table 1: Transition fault test coverage with different test pattern allocation methods**

Ckt	# Flts (Mil.)	# Clks	512K		768K		1024K	
			Opt. (%)	Even (%)	Opt. (%)	Even (%)	Opt. (%)	Even (%)
ckt1	0.75	6	57.87	54.77	59.37	56.02	60.30	57.23
ckt2	0.82	2	71.82	71.49	72.49	72.19	72.91	72.64
ckt3	1.06	11	78.93	78.48	79.25	79.03	79.45	79.30
ckt4	1.82	12	59.94	59.30	60.09	59.53	60.19	59.69
ckt5	2.28	7	59.33	57.18	60.24	58.07	60.86	58.70
ckt6	5.55	10	71.82	70.01	72.63	70.96	73.20	71.57

**Table 2: Area Overhead Evaluation**

Depth	# Clocks			
	6	8	10	12
5	14.1%	18.5%	22.9%	27.1%
10	26.8%	35.1%	43.7%	51.7%
15	39.5%	52.1%	65.4%	77.6%

To evaluate the area overhead of the proposed programmable logic BIST controller, we use one industrial design with traditional logic BIST. The core design is a black box so we have no knowledge about the core. We only take out the RTL code of the BIST controller for our experiments. The original RTL BIST controller has only a single clock domain. We first modify the controller to handle N clocks (N=6, 8, 10 12). We then use *Design Compiler* to synthesize the BIST controller RTL code and get the areas for each modified version. The proposed 2-D scan register bank is inserted into each modified RTL BIST controller next. For the case with N clocks, the width of the 2-D scan register is also N. The depths of the 2-D scan register vary from 5, 10 to 15. In Table 2, we report the percentage of area overhead over the corresponding non-programmable BIST controller for each synthesized programmable BIST controller. For example, when driving 6 clocks, the area overhead for the proposed programmable BIST is 14.1% with depth 5. It can be seen from Table 2 that the area overhead is almost linearly increased with the width or depth of a 2-D scan register bank. In the worst case among all the experiments we did, the area overhead is 77.6% when handling 12 clocks in a capture window with depth 15. The area overhead is acceptable since the area of the original non-programmable BIST controller is typically a small percentage (less than 5% in general) of the entire design.

## 6. Conclusions

In this paper, a novel programmable logic BIST controller was proposed to facilitate at-speed test for the design with multiple clock domains and multiple clock frequencies. In addition, a static analysis method is also

proposed to optimize the BIST test pattern allocation for testing the at-speed faults in different intra/inter clock domains. Preliminary experimental results showed that the proposed programmable BIST controller has acceptable area overhead while the propose pattern allocation algorithm is effective to achieve higher at-speed test coverage.

## References

- [1] K. Kim, S. Mitra, and P.G. Ryan, "Delay Defect Characteristics and Testing Strategies," in *IEEE D&T of Computers*, Sept.-Oct., 2003, pp. 8-16.
- [2] B.R. Benware, R. Madge, C. Lu, and R. Daasch, "Effectiveness Comparisons of Outlier Screening Methods for Frequency Dependent Defects on Complex ASICs," in *Proc. of VTS*, 2003, pp. 39-46.
- [3] X. Lin, R. Press, J. Rajski, P. Reuter, T. Rinderknecht, B. Swanson, and N. Tamarapalli, "High-Frequency, At-Speed Scan Testing," in *D&T of Computers*, Sept.- Oct. 2003, pp. 17-25.
- [4] N. Tendolkar et al., "Novel Techniques for Achieving High At-Speed Transition Fault Test Coverage for Motorola's Microprocessors Based on PowerPC Instruction Set Architecture," in *Proc. of VTS*, 2002, pp. 3-8.
- [5] M. Beck, O. Barondeau, M. Kaibel, F. Poehl, X. Lin, and R. Press, "Logic Design for On-chip test Clock Generation – Implementation Details and Impact on Delay Test Quality," in *Proc. DATE*, 2005, pp. 56-61.
- [6] R. Press and J. Boyer, "Easily Implement PLL Clock Switching for At-Speed Test - By Taking Advantage of Pattern-Generation Features, A Simple Logic Design can Utilize Phase-Locked-Loop Clocks for Accurate At-speed Testing," in *Chip Design Magazine*, Feb.-Mar. 2006.
- [7] B. Koenemann, "LFSR-Coded Test Patterns for Scan Design," in *Proc. European Test Conf.*, 1991, pp. 237-242.
- [8] M. A. Miranda and C. A. Lopez-Barrio, "Generation of optimized Single Distributions of Weights for Random Build-in Self-test," in *Proc. of ITC*, 1993, pp. 1023-1300.
- [9] S. Hellebrand, J. Rajski, S. Tarnick, S. Venkataraman, and B. Courtois, "Build-in Test for Circuits with Scan Based on Reseeding of Multiple-polynomial Linear Feedback Shift Register," in *IEEE Trans. Computer*, vol. C-44, Feb. 1995, pp. 223-233.
- [10] P. Wohl, J.A. Waicukauski, S. Patel; F. DaSilva; T.W. Williams, and R. Kapur, "Efficient Compression of Deterministic Patterns into Multiple PRPG Seeds," in *Proc. of ITC*, 2005, paper 36.1.
- [11] K. Hatayama, M. Nakao, and Y. Sato, "At-Speed Built-in Test for Logic Circuits with Multiple Clocks," in *ATS 2002*, pp.292-297, 2002.
- [12] P. Chao and L. Lev, "Down to the Wire Requirements for Nanometer Design Implementation," in *EE Design*, August 15, 2002.
- [13] V. Iyengar, G. Grise, and M. Taylor, "A flexible and scalable methodology for GHz-speed structural test," in *Proc. DAC 2006*, pp. 314 - 319.
- [14] LBISTArchitect™ Process Guide, Mentor Graphics Corporations.