Testing High-Performance Pipelined Circuits with Slow-Speed Testers

MUHAMMAD NUMMER and MANOJ SACHDEV
University of Waterloo

This article presents a methodology for testing high-performance pipelined circuits with slow-speed testers. The technique uses a clock timing circuit to control data transfer in the pipeline in test mode. The technique adds no extra hardware in the data path of the pipeline and therefore has virtually no performance penalty. A clock timing circuit capable of achieving a timing resolution of 50 ps in 0.18 μm CMOS technology is presented. The design provides the ability to test the clock timing circuit itself. The effectiveness of the technique is demonstrated using a 16-bit pipelined multiplier as a test vehicle. Simulations show that we are able to detect delay faults as small as 50 ps at an input clock frequency of 100 MHz.

Categories and Subject Descriptors: B.6.2 [**Logic Design**]: Reliability and Testing—*testability*; B.7.3 [**Integrated Circuits**]: Reliability and Testing—*testability*

General Terms: Design, Reliability

Additional Key Words and Phrases: Delay-fault testing, high-performance testing, design for delay testability

1. INTRODUCTION

The 2001 edition of the International Technology Roadmap for Semiconductors (ITRS) expects that the clock frequency of high-performance state-of-the-art CMOS VLSI circuits will exceed 6 GHz by year 2007 [ITRS 2001]. According to the ITRS, potential manufacturing yield loss associated with the at-speed functional test methodology is related to the growing gap between automatic test equipment (ATE) performance and the ever increasing device I/O speed. In the last two decades, while the clock frequencies of VLSI circuits have improved at an average rate of 30% per year, the tester accuracy has improved only at a rate of 12%. If this trend continues, tester timing accuracy will soon approach the cycle time of high-performance devices making at-speed test almost impossible.

This article is an extended version of the authors' work published in *Proceedings of the Design, Automation and Test in Europe Conference 2003* (*DATE03*), vol. 3 (March), 212–217.
Authors' address: Electrical and Computer Engineering, University of Waterloo, 200 University Avenue West, Waterloo, ON, N2L 3G1, Canada.

Due to the slow advances and the high cost of ATEs, we might not be able to test future high-performance VLSI circuits. Therefore, it will be essential to design these circuits with design-for-testability/built-in-self-test (DFT/BIST) techniques to reduce the reliance on traditional, high-cost, full-feature testers. The requirements of ATEs designed to work with DFT/BIST techniques are much simpler than the traditional testers.

In this article, we propose a methodology for testing high-performance pipelines using slow-speed testers. The technique depends on test mode clock shifting. The technique adds no extra hardware in the data path of the pipeline and therefore has virtually no performance penalty.

2. BACKGROUND

The size of a defect determines whether the defect affects the logic functionality of a circuit. Normally, smaller defects, which are likely to cause partial shorts or opens, have a higher probability of occurrence. Such defects often cause timing failures without altering the logic functionality of the circuit. A number of recent studies show concerns about new failure mechanisms in scaled geometries that are harder to detect with conventional means. Nigh et al. [1997] reported a significantly large number of timing-only failures that did not affect the steady-state logic functionality. Similarly, for Intel's manufacturing processes, Needham et al. [1998] reported an increasing shift towards soft defects as technology moved from 0.35 to 0.25 μm. These defects do not always cause failures at all temperature and voltage conditions and are considered to be major long term reliability threats.

Using DFT/BIST techniques that allow testing high-performance circuits with slow-speed testers is one way to tackle the problem of high-performance circuit testing. The creation of a low frequency test mode in digital circuits was first introduced by Agrawal and Chakraborty [1995]. In their proposal, a quantifiable, externally controlled delay is added such that high-performance testing can be carried out with relatively slow-speed testers. They used a pulse-triggered flip-flop in which a dynamic latch is introduced inside a traditional master-slave flip-flop. Shashaani and Sachdev [1999] proposed the controlled-delay flip-flop (CDFF) as an alternative to the pulse-triggered flip-flop. In this technique, an additional test mode clock is used to control the delay of the flip-flop. The main advantages of the CDFF over the pulse-triggered flip-flop are the stable operation and improved performance in normal mode. Kerkhoff et al. [2001] suggested a BIST environment for detecting delay faults based on the use of CDFFs. Nummer and Sachdev [2001, 2003] proposed an on-chip clock generation methodology that allows the test mode clock frequency of circuits using CDFFs to be reduced arbitrarily.

3. TESTING PIPELINES BY CLOCK SHIFTING

The basic idea in the proposed technique is demonstrated in Figure 1. In this study, we assume that the registers in the pipeline use a single phase common clock and are built using flip-flops as the storage element. A clock timing circuit is used to control clocks driving all registers in the pipeline. The idea is to use

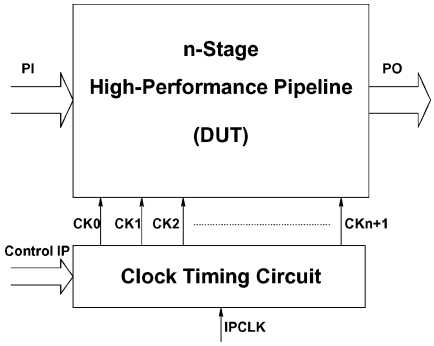

Fig. 1. Block diagram of the DUT and the clock timing circuit.

shifted version(s) of the input clock, IPCLK, to control the timing of data flow through the pipeline in the test mode. In order to achieve that, each register in the pipeline has to have a separately routed clock, as shown in Figure 1. This results in higher complexity of clock generation and propagation. The clock net of a pipelined circuit normally consists of a tree of buffers. Special care should be given to balance the load of the different clock tree branches in order to keep the skew between the different clock signals within acceptable limits. In our technique, only those buffers close to the clock timing circuit require extra design and layout effort. Due to the small number of these upstream buffers, the extra effort imposed by the technique would not be substantial.

The normal and test mode operation of the circuit is illustrated in Figure 2, showing stage i in a pipeline ($0 \leq i \leq n$). In normal mode, a single phase high-frequency clock is used for all registers in the pipeline. This is shown in Figure 2(b). As a result, the operation of the circuit depends on the period of this clock. The delay of stage i , t_{d_i}, can be expressed using

$$t_{d_i} = t_{prop_i} + t_{comb_i} + t_{setup_{i+1}} + \left(t_{CK_i} - t_{CK_{i+1}}\right), \tag{1}$$

where t_{prop_i} is the propagation delay of register i, t_{comb_i} is the delay of the ith stage combinational block, $t_{setup_{i+1}}$ is the setup time of register $i+1$, and $(t_{CK_i} - t_{CK_{i+1}})$ is the difference between the delays through the clock driving networks of registers i and $i+1$. For the pipeline to function correctly, the normal mode clock period, T_{NM}, has to be at least equal to the largest stage delay, that is,

$$T_{NM} \geq MAX_{i=0}^{n}\left(t_{d_i}\right). \tag{2}$$

In the test mode, a delayed version of the clock is used to test the pipeline. This is illustrated in Figure 2(c). In order to test stage i, a delayed version of the input clock, IPCLK, with delay T_d is applied to register $i+1$, while the original clock is used for all other registers. The test mode clock period, T_{TM}, has to be at least equal to the normal mode clock period. It is clear that using a larger value of T_{TM} means that we can use a slow-speed tester in the test mode. Setting T_d to be equal to t_{d_i} allows the ith stage to operate within its normal mode timing constraints while the whole circuit is running at a lower frequency. As a result, a slow-speed tester can be used for performance binning and delay fault testing. In the test mode, the tester supplies a slow-speed input clock as well

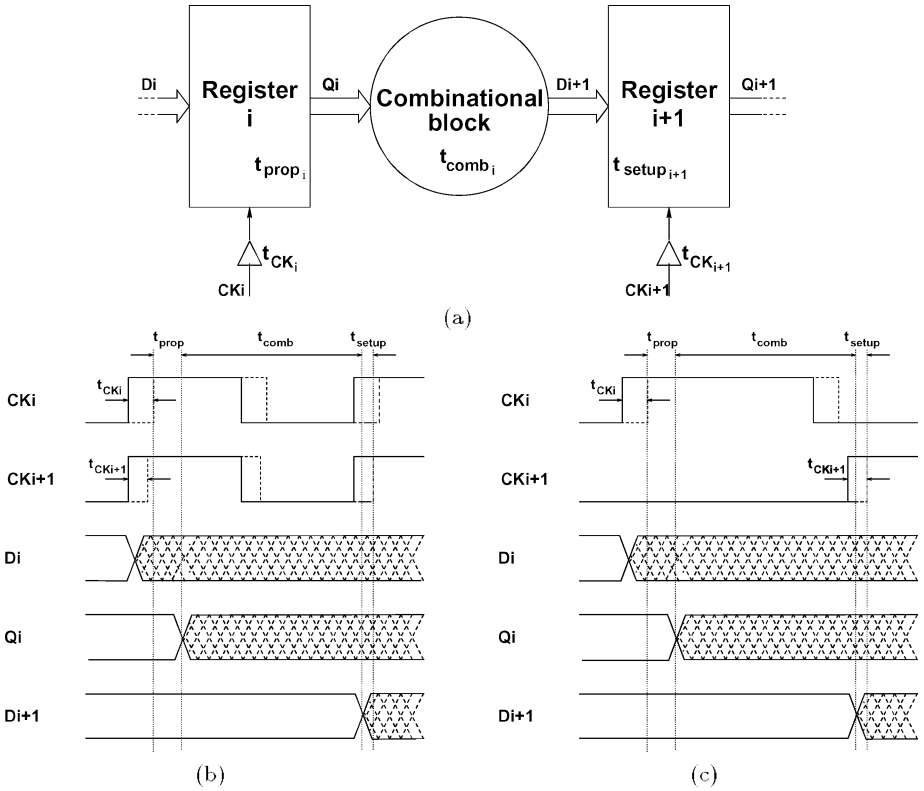

Fig. 2. Proposed technique. (a) Circuit model. (b) Normal mode. (c) Test mode.

as test vectors for the target stage. These vectors are supplied at the rate of the slow-speed clock. After a predetermined number of clock cycles, the tester reads the results at the same rate of the slow-speed clock. This technique does not require any changes in the design of the registers or the combinational blocks and therefore has virtually no performance penalty.

For the proposed technique to achieve its goals, it is essential for the value of T_d to be controllable within state-of-the-art timing accuracy. Furthermore, we should have the ability to program T_d in order to bin the device under test (DUT) for performance. This feature also allows different stages in the pipeline to be tested even if they have unbalanced delays. For these reasons, an on-chip clock timing circuit is used to generate the delayed version of the input clock and to control the clocks propagated to the different stages in the pipeline.

4. TEST VEHICLE: 16-BIT PIPELINED MULTIPLIER

In order to verify the benefits of the proposed technique, a 16-bit pipelined multiplier is designed and utilized as a test vehicle. Parallel multiplication is done in three steps. In the first step, the two operands are used to generate partial products whose number depends on the type of encoding algorithm used.

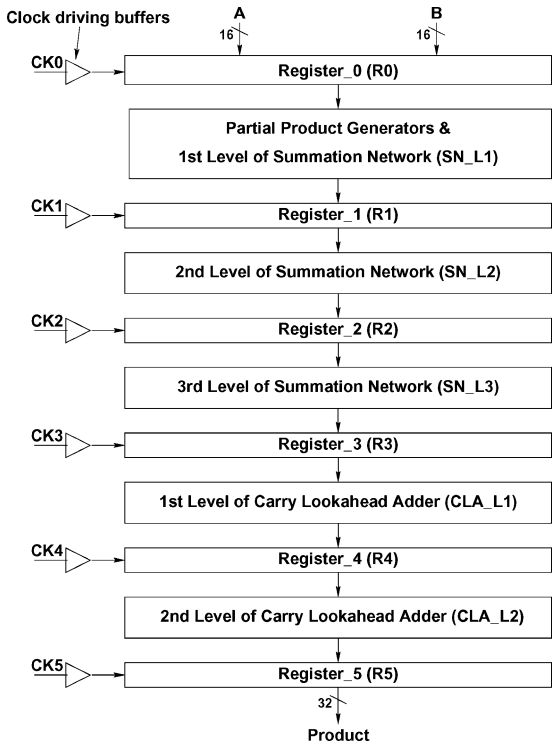

Fig. 3. Block diagram of the pipelined multiplier used as a test vehicle.

The second step is to add the partial products together in a summation network which reduces the partial products to two operands. The product is generated in the final step by adding the resulting two operands using a carry propagate adder.

In our design, no encoding is used to generate the partial products. This results in a number of partial products equal to the size of the multiplier (16). These partial products are added in the summation network using 4-2 compressors as the main component. A 4-2 compressor accepts four partial sums and reduces them to two [Mehta et al. 1991]. In order to reduce the 16 partial products to 2 operands, this has to be done in three levels of 4-2 compression. A carry-lookahead adder with conditional sum select [Ohkubo et al. 1995] is used to generate the product from these two operands.

The multiplier is implemented with five pipelined stages, as shown in Figure 3. The first stage is used to generate the 16 partial products and reduce them to 8 partial sums after the first level of the summation network. The second and third levels of the summation network are implemented in the second and third stages of the pipeline. The final addition is done in the last two stages.

Pipeline stages are separated by registers to control the timing of data flow through the multiplier. Static flip-flops are used as the storage elements in all registers. As shown in Figure 3, each register is controlled by a separate clock

Table I. Critical Path Delays through Multiplier Stages

Pipeline stage	Critical path delay (ps)
SN_L1	715
SN_L2	655
SN_L3	655
CLA_L1	665
CLA_L2	615

provided through the clock timing circuit. Details of the design of this circuit are given in the next section.

Performance characterization of the multiplier is carried out in order to find its maximum operating frequency and the critical path through each stage of the pipeline. These results are shown in Table I. The delays shown include the propagation delay of the register feeding the stage, the setup time of the register accepting the output of the stage, and the difference in delays of the clock driving networks of the input and output registers (refer to Eq. 1). As shown in Table I, the first stage (SN_L1) has the largest delay and the operating frequency of the multiplier is determined by this stage. This delay is equal to 715 ps which translates to a maximum operating frequency of 1.4 GHz.

5. CLOCK TIMING CIRCUIT

This section provides design details of the clock timing circuit used for the test vehicle. A number of issues have to be taken into account in the design of the clock timing circuit in order to ensure proper delay-fault testability of the pipeline.

(1) *Programmability and Resolution.* The delay, T_d, should be programmable for two reasons: (i) to allow DUT performance binning , and (ii) to allow paths of different delays to be tested for delay faults. The resolution with which T_d is varied should be high enough to allow the detection of small delay faults.

(2) *Timing Accuracy.* Process, temperature, and supply voltage variations have a large effect on the delays of the circuit. The design should compensate for these effects such that the value of T_d will be as desired regardless of these variations.

(3) *Tester Edge Placement Inaccuracy* (*Jitter*). A slow-speed tester is generally a low-specification device. It has a low timing resolution and a high edge placement inaccuracy, or jitter. The design should have the ability to suppress the tester's jitter in order to minimize its effect on the timing resolution and accuracy of the clock timing circuit.

(4) *Testability.* Adding hardware to enhance testability begs the question: How can we ensure the functionality of the extra hardware? Is it testable? Answering these questions is very important and is a challenge to most design-for-testability techniques.

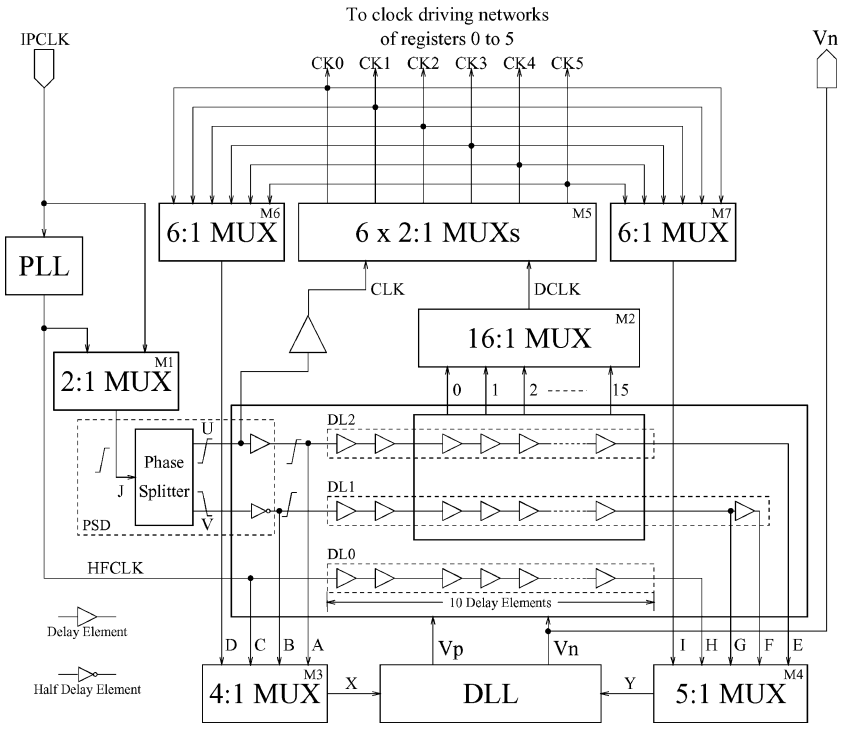

Fig. 4. Clock Timing Circuit (multiplexer select inputs are not shown).

Figure 4 shows the clock timing circuit used in our design (multiplexer select inputs are not shown). This circuit is designed using CMOS 0.18 μm technology provided through TSMC. It comprises a phase-locked-loop (which could be part of the clock generation circuit for the chip, as is the case for high-performance microprocessors, for example), three delay lines, a phase splitter & delay circuit, a DLL, and a number of multiplexers. The design allows T_d to vary between 250 ps and 1000 ps with 50 ps increments (resolution). These are design variables and it is up to the designer to choose the values suiting a specific circuit or application. For our test vehicle, the range of T_d is reasonable to allow us to do performance binning and test for delay faults. A 50 ps timing resolution is high enough considering the maximum operating frequency of the multiplier. The design and operation of the different blocks used in the clock timing circuit is described below.

(1) *Phase-locked-loop (PLL).* The PLL works as a frequency multiplier. It is used to generate a high-frequency clock from a low-frequency input clock in the test mode. It is assumed that the normal mode input clock is a high-frequency clock, and hence the PLL is not active in that mode (the design could be such that the same PLL is used to generate the normal mode high-frequency clock from a slower input clock). Our design assumes that the input clock from the tester (IPCLK) has a frequency of 100 MHz (this is a design variable and could be different). The PLL generates a high-frequency

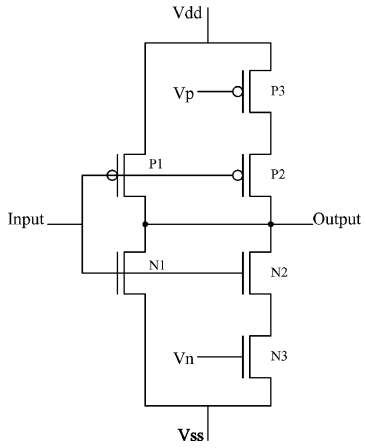

Fig. 5. Schematic diagram of half the delay element.

clock (HFCLK) with a frequency of 1 GHz (10 times the input clock frequency). HFCLK is used only to calibrate the delay lines used to generate T_d. Using a PLL to generate the high-frequency clock used for calibration serves two purposes: (i) it helps suppress tester jitter and reduce its effect on the accuracy of the design, and (ii) it allows us to use smaller number of delay elements in the delay lines as will be shown later.

(2) *Delay-locked-loop (DLL).* The main function of the DLL is to calibrate the delay lines used to generate the delayed version of the input low-frequency clock. This is particularly important in order to ensure correct timing regardless of process, temperature, and/or supply voltage variations. Its outputs, V_p and V_n, are used to control the delays of the delay elements. The values of these control voltages depend on the phase difference between the two inputs of the DLL (X and Y). When locked, V_p and V_n should be such that the signal at Y is delayed by one complete cycle from the signal at X. The DLL serves another important purpose. It allows great flexibility for testing the clock timing circuit itself. As a result, most of the extra hardware in our design can be tested simply by comparing values of the DLL outputs (V_p and V_n) when different signals are used to close the loop.

(3) *Delay lines and phase splitter & delay circuit.* Three delay lines, DL0, DL1, and DL2, are used in the design. Each delay line comprises a number of delay elements. The schematic diagram of half the delay element is shown in Figure 5. It consists of a static inverter connected in parallel to a current starved inverter [Moyer 1996]. The circuit delay is controlled by the two bias voltages V_p and V_n. When transistors P3 and N3 are turned off, the delay is maximum and depends on the sizes of transistors P1 and N1. On the other hand, when V_p and V_n are equal to V_{ss} and V_{dd}, respectively, maximum current is allowed to flow through the current starved inverter resulting in minimum delay.

Initially, our objective was to design a delay element with a delay of 50 ps (the resolution of T_d). We were able to achieve that only for one edge of the input clock. The delay for the other edge was much higher. Using the DLL

to calibrate the delay lines requires that the delay for any edge to be within a certain limit, otherwise the signal will be lost before reaching the end of the delay line. As this is not feasible using the technology at hand (0.18 μm CMOS technology), we designed the delay elements to have a delay of 100 ps and used two delay lines with a 50 ps delay in between to get a resolution of 50 ps. It is worth noting that sizing of the transistors in the delay element should be such that there are values of the control voltages that result in a delay of 100 ps under worst case conditions (slow-PMOS and slow-NMOS transistor models, $T = 100^{\circ}$C, and V_{dd} is 10% lower than its nominal value) as well as best-case conditions (fast-PMOS and fast-NMOS models, room temperature, and V_{dd} is 10% higher than its nominal value).

DL0 consists of 10 delay elements. It is used to calibrate the other two delay lines. This is achieved by closing the DLL loop using DL0 and applying HFCLK from the PLL to the input of this delay line. The period of HFCKL is 1 ns for an input clock frequency of 100 MHz. Once locked, the output voltages from the DLL should be such that the total delay across DL0 is equal to one clock period. Since these voltages control all three delay lines, the delays of all delay elements in the circuit are adjusted to 100 ps. If IPCLK were to be used to calibrate the delay lines, DL0 would have 100 delay elements rather than only 10. This explains the benefit of generating HFCLK using the PLL.

As shown in Figure 4, the phase splitter & delay (PSD) circuit accepts either IPCLK (low-frequency clock, in the test mode) or HFCLK and generates two clocks, A and B. The PSD circuit is designed such that the clock at A is delayed by half delay-element delay with reference to the clock at B. Therefore, when V_p and V_n are such that the delay of the delay element is 100 ps, the delay from B to A is 50 ps. These two clocks are fed to the other two delay lines, DL1 and DL2. DL1 and DL2 consist of 11 and 10 delay elements, respectively. The main function of these two delay lines is to generate DCLK with programmable delays with respect to CLK (Figure 4). It is worth noting that the input clock jitter has virtually no effect on the timing accuracy of DCLK and hence on the high-performance delay-fault testability. Any edge placement inaccuracy at the Input, propagates through DL1 and DL2. As a result, the same amount of jitter is added to both CLK and DCLK keeping the delay between these two clocks at the desired value of T_d.

(4) *Multiplexers M1–M7.* The multiplexers shown in Figure 4 can be divided into two groups. The first group includes M2 and M5. Outputs from 16 delay elements in DL1 and DL2 are tapped and fed to M2. According to the select inputs of M2, one of the 16 inputs is selected to be DCLK. This allows the delay between CLK and DCLK to be varied between 250 ps and 1000 ps with 50 ps increments (note that CLK is buffered to compensate for the delay through M2). M5 consists of six 2:1 multiplexers. It is used to control the clocks feeding all registers in the pipeline (CK0 to CK5). Depending on the mode of operation and the pipeline stage to be tested, M5 sets the clock of each register to either CLK or DCLK.

The second group of multiplexers include M1, M3, M4, M6, and M7. These multiplexers are used only to ensure the functionality of the clock timing circuit. Details on the operation of these multiplexers are given in the next section.

The proposed clock timing circuit takes into account all the issues mentioned at the beginning of this section. Programmability is achieved through DL1, DL2, and M2. Using a DLL allows us to achieve the same timing accuracy regardless of process, temperature, and/or supply voltage variations. The DLL with a number of multiplexers help ensure the functionality of the clock timing circuit itself. With the help of the PLL, tester clock jitter is not allowed to propagate to DL0. It also has no effect on the timing accuracy of signals generated form DL1 and DL2. The area overhead due to this design is estimated to be 100 gates per pipeline stage. This should be acceptable for medium to large pipelined circuits. It is important to note that matching between the different components in the design is essential to ensure correct timing even with small local process, temperature, and/or supply voltage variations. This can be achieved through circuit layout techniques similar to those used for analog circuits.

6. MODES OF OPERATION

In this section, we put the different blocks of the clock timing circuit together and show how they function in the different modes of operation. The clock timing circuit operates in three different modes. In *normal mode*, the pipeline is used to perform the function it is designed for. In the *DUT test mode*, the clock timing circuit is used to verify the performance of the DUT and to test it for delay faults. In the *clock timing circuit test mode*, the clock timing circuit is tested to ensure its ability to give a correct image about the performance of the pipeline.

6.1 Normal Mode

In normal mode, the control inputs of multiplexers M1 and M5 are set such that IPCLK propagates to all registers in the pipeline. As a result, the whole pipeline runs using the same clock ensuring normal operation. The remainder of the clock timing circuit serves no function in normal mode, and can be turned off. This is important to reduce the effect the clock timing circuit has on normal mode power consumption of the chip.

6.2 DUT Test Mode

This mode is used to verify the performance of the DUT and test it for delay faults. The pipeline is tested one stage at a time. For every target path, two vectors are used to test the circuit. The first vector initializes the DUT, while the second vector activates the target path of the stage to be tested. Using multiplexer M2, T_d is set according to the expected delay of that path. Depending on the stage to be tested, multiplexer M5 is configured to set one of the register's clocks to DCLK and all other clocks to CLK. For example, to test stage 3 in the pipeline, clocks CK0, CK1, CK2, CK3, and CK5 are set to CLK while CK4 is

set to DCLK. As a result, data flows between registers 3 and 4 within normal mode timing constraints. For all other stages, the low frequency clock allows operation under relaxed timing. This is important to ensure that delay faults in these stages do not affect the target path and hence the stage under test. This procedure is repeated for every path to be tested until the DUT is tested completely.

6.3 Clock Timing Circuit Test Mode

In this mode, the DLL, with the help of multiplexers M1, M3, M4, M6, and M7, is used to characterize and verify the performance of the clock timing circuit. Multiplexer M1, is set to allow the high-frequency clock, HFCLK, to propogate through DL1 and DL2. The clock timing circuit is tested in three phases as shown in Figure 6. In the first phase, all three delay lines (excluding the last delay element in DL1) are tested. The second phase is used to test the PSD circuit. This is done by verifying that the delay between nodes A and B is equal to half delay-element delay. The third phase is used mainly to test multiplexer M5. This phase also covers multiplexers M6, M7, and a small part of M2. Each of the three phases covers part of multiplexers M3 and M4, such that these two multiplexers are fully testable. In our design, only M1 and most of M2 are not testable, which is a very small area compared to the rest of the clock timing circuit and the DUT.

6.3.1 *Phase 1: Testing DL0, DL1, and DL2.* In this phase, delay lines DL0, DL1, and DL2 are tested between nodes C and H, B and G, and A and E, respectively. Table II gives the number of delay elements between the different nodes in the clock timing circuit. The first three entries in the table have the same number of delay elements. As a result, closing the DLL loop with these node pairs one at a time should result in very close values of V_n (monitored off-chip) for all three configurations. As shown in Figure 6, the test in this phase is done in four steps:

(1) Close the DLL loop using C and H ($M3 : C \rightarrow X; M4 : H \rightarrow Y$). Measure V_n and record it as V_{n_0}.
(2) Close the DLL loop using B and G ($M3 : B \rightarrow X; M4 : G \rightarrow Y$). Measure V_n and record it as V_{n_1}.
(3) Close the DLL loop using A and E ($M3 : A \rightarrow X; M4 : E \rightarrow Y$). Measure V_n and record it as V_{n_2}.
(4) Compare V_{n_0}, V_{n_1}, and V_{n_2}. If the differences between the three values are within acceptable limits, the three delay lines are considered to be free of delay faults.

Ideally, the three values should be equal. Mismatch between the delay elements and the different components in the DLL would result in some differences. Characterization is necessary to define how much difference due to mismatch and process variations is acceptable. In this context, we assume that only single delay fault exists in the circuit. It is highly unlikely to have the

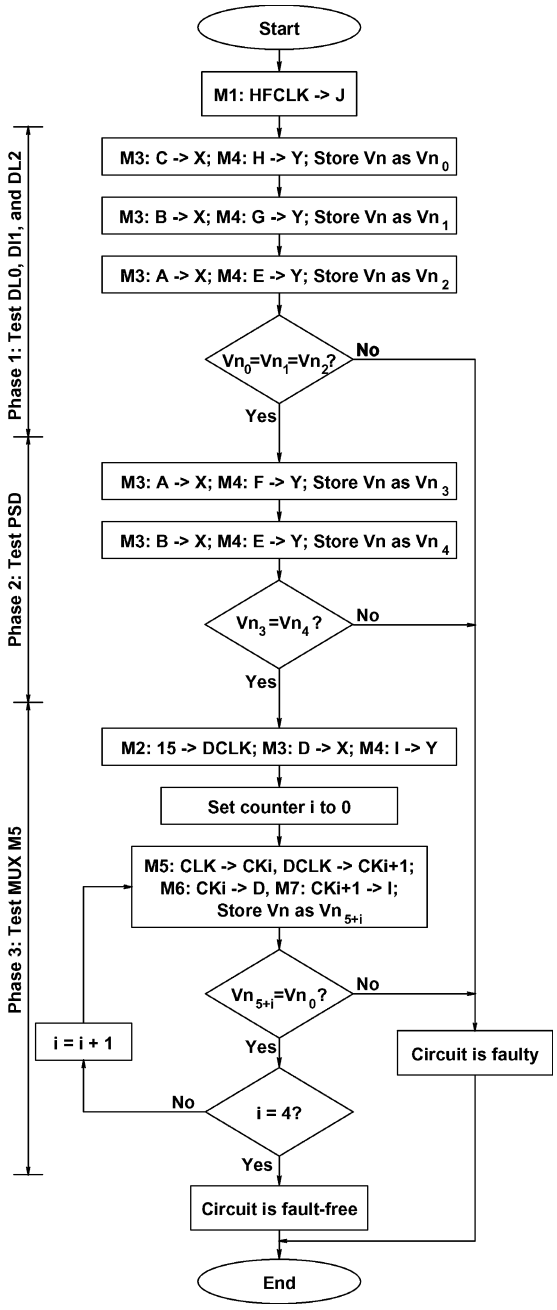

Fig. 6. Procedure for testing the clock timing circuit.

exact same amount of delay fault in two of or all three delay lines. In addition to DL0, DL1, and DL2, the test in this phase covers paths through multiplexers M3 and M4 used to close the DLL loop (A-X, B-X, C-X, E-Y, G-Y, and H-Y).

Table II. Number of Delay Elements between Different Nodes in the Delay Lines

Nodes	# of delay elements
A and E	10
B and G	10
C and H	10
A and F	$10\frac{1}{2}$
B and E	$10\frac{1}{2}$

6.3.2 *Phase 2: Testing the Phase Splitter & Delay Circuit.* Referring to Table II, the delay between nodes A and F on one side and nodes B and E on the other are both equivalent to $10\frac{1}{2}$ delay elements. This is only true if the two outputs of the PSD circuit (A and B) are exactly half delay-element delay apart. This observation is used to test the PSD circuit for delay faults. As shown in Figure 6, the test is done first by closing the DLL loop using nodes A and F. When locked, DLL output V_n is recorded as V_{n_3}. Similarly, V_{n_4} is obtained by closing the DLL loop through nodes B and E. For fault-free PSD, the difference between V_{n_3} and V_{n_4} should be within acceptable limits (defined through characterization). The test in this phase covers the G-F-Y path as well.

6.3.3 *Phase 3: Testing Multiplexer M5.* This is done with the help of multiplexers M6 and M7. The main idea is to set the delay between CLK and DCLK (T_d) to one cycle of HFCLK (1 ns). The different paths in M5 are tested two at a time. As shown in Figure 6, to test the paths from CLK to CK_i and DCLK to CK_{i+1} (where i is any number between 0 and 4), M5 is set accordingly and M6 and M7 are used to close the DLL loop using nodes CK_i and CK_{i+1}. Under these conditions, if all signal paths are free of delay fault, when locked, the DLL output, V_n, should be equal to the values obtained in phase 1 of the test procedure (for fault free DL0, DL1, and DL2). In addition to M5, the test in this phase covers M6, M7, the 15-DCKL path in M2, the D-X path in M3, the I-Y path in M4, and the buffer used for CLK.

7. DELAY-FAULT SIMULATION RESULTS

Several simulations are carried out to test our design and to verify the possibility of testing the DUT and detecting small delay faults at low frequency. Furthermore, delay-fault simulations for the clock timing circuit are done in order to demonstrate our ability to test and characterize this circuit. In all our simulations, delay faults are inserted using buffers and delay elements. Using active elements to add extra delays ensures the existence of a predetermined delay fault without degrading the quality (rise and fall times) of the delayed signals.

7.1 Delay-Fault Simulation for the Test Vehicle

Delay faults are inserted in the test vehicle one at a time using buffers. Simulations are carried out on a number of paths in the DUT. For every

Table III. Test Vehicle Delay-Fault Simulation Results

Path #	Pipeline stage	Path delay (ps)	T_d (ps)	Vector type	Input A	Input B	Product (fault-free)	Product (faulty)
1	SN_L1	715	750	Init.	0002	FFFF	0001 FFFE	0001 FFFE
				Activ.	0000	FFFF	0000 0000	0000 0008
2	SN_L1	685	700	Init.	000C	FFFF	000B FFF4	000B FFF4
				Activ.	0008	FFFF	0007 FFF8	0007 FFF0
3	SN_L2	655	700	Init.	0001	FFFF	0000 FFFF	0000 FFFF
				Activ.	0000	FFFF	0000 0000	0000 0040
4	SN_L2	620	650	Init.	0070	FFFF	006F FF90	006F FF90
				Activ.	0040	FFFF	003F FFC0	003F FF80
5	CLA_L1	665	700	Init.	0000	FFFF	0000 0000	0000 0000
				Activ.	FFFF	FFFF	FFFE 0001	FFFD 0001
6	CLA_L1	425	450	Init.	FFFF	FFFF	FFFE 0001	FFFE 0001
				Activ.	0000	FFFF	0000 0000	0001 0000
7	CLA_L2	615	650	Init.	0000	FFFF	0000 0000	0000 0000
				Activ.	FFE0	FFF0	FFD0 0200	FFC0 0200
8	CLA_L2	415	450	Init.	FFE0	FFF0	FFD0 0200	FFD0 0200
				Activ.	0000	FFFF	0000 0000	0010 0000

target path, two vectors are used to test the circuit. The first vector initializes the DUT while the second vector activates the target path. The delay of a given path depends on the input vectors of the circuit. As a result, for every stage in the pipeline, the critical path and its delay might change depending on the applied vectors. For all paths tested in our simulations, activation vectors are chosen such that these paths are the critical paths in their pipeline stage for these vectors. As mentioned before, our design allows adjustment of T_d such that paths of different delays can be tested for small delay faults.

Delay-fault simulation results for the test vehicle are shown in Table III. The test is done at an input clock frequency of 100 MHz. Delay faults of 50 ps are inserted in each path one at a time. T_d from the clock timing circuit is set to the next higher value compared to the delay of the target path. Under these conditions, the multiplier gives incorrect output for all paths tested in our simulations, as shown in Table III. The left half of the table gives the delays of the different paths and the value of T_d used to detect a 50 ps delay fault in each path. The right half of the table gives the test vectors used for each path as well as the fault-free and faulty products of the multiplier. The extent of delay fault that goes undetected is a function of the slack between the delay of the path and the value of T_d used to test it. A path with larger slack will have a larger undetectable delay fault. For example, in the case of path # 2, we expect that delay faults as small as 15 ps should be detectable using the same value of T_d. On the other hand, for path # 3, it will take a delay fault of at least 45 ps to cause the timing failure. The situation is worse if the target path is not the critical path for the applied vectors. In general, delay fault detection is dependent on the target path delay. Most of the delay fault testing techniques have similar limitations. Balancing path delays is the most commonly used method to alleviate this problem.

Table IV. Delay-Fault Simulation Results for the Clock Timing Circuit

Fault #	Fault location	Faulty Path	Delay fault (ps)	V_{n_0} mV	V_{n_1} mV	V_{n_2} mV	$\triangle V_n$ mV
F1	DL0	C-H	60	**633**	611.5	611.5	21.5
F2	DL1	B-G	100	611.5	**649**	611.5	37.5
F3	DL2	A-E	200	611.5	611.5	**691.7**	80.2
F4	M3	A-X	60	611.5	611.5	**594.2**	−17.3
F5	M3	B-X	100	611.5	**583.3**	611.5	−28.2
F6	M3	C-X	200	**554**	611.5	611.5	−57.5
F7	M4	E-Y	60	611.5	611.5	**634**	22.5
F8	M4	G-Y	100	611.5	**648.7**	611.5	37.2
F9	M4	H-Y	200	**697**	611.5	611.5	85.5

(a) Faults detected in phase 1 (fault-free V_n is 611.5 mV)

Fault #	Fault location	Faulty Path	Delay fault (ps)	V_{n_3} mV	V_{n_4} mV	$\triangle V_n$ mV
F10	PSD	J-A	60	609.8	651.5	−41.7
F11	PSD	J-B	100	669	597.9	89.1
F12	M4	F-Y	200	720	628.6	91.4

(b) Faults detected in phase 2 (fault-free V_n is 628.6 mV)

Fault #	Fault location	Faulty Path	Delay fault (ps)	V_n mV	$\triangle V_n$ mV
F13	M2	15-DCLK	60	633.2	21.7
F14	M5	DCLK-CK4	100	647	35.5
F15	Buffer	U-CLK	200	554.8	−56.7

(c) Faults detected in phase 3 (fault-free V_n is 611.5 mV)

7.2 Delay-Fault Simulation for the Clock Timing Circuit

Delay-fault simulations for the clock timing circuit are carried out in order to verify our ability to characterize the performance of the circuit. Fifteen delay faults are inserted at different locations of the circuit one at a time. These fault are generated using the same delay element used in the design. The delay of the delay element is at its minimum value when $V_p = V_{ss}$ and $V_n = V_{dd}$. This value is 60 ps, and hence the smallest delay fault used in the simulations is 60 ps. Larger delays are achieved using larger values of V_p and smaller values of V_n. The test is done according to the procedure in Figure 6. The results are shown in Tables IV(a), (b), and (c), showing faults detected in each phase of the clock timing circuit test procedure.

A fault detectable in phase 1 or phase 3 of the test procedure causes only one value of V_n (V_{n_0}, V_{n_1}, or V_{n_2} for phase 1 or $V_{n_{5+i}}$ for phase 3) to deviate from the the fault-free value. That is why for faults detected in phase 1 and phase 3, the last column in each table gives the difference in V_n compared to the fault-free value. This is different from faults detectable in phase 2, where the fault causes both V_{n_3} and V_{n_4} to deviate in opposite directions from the fault-free value. For faults detected in phase 2, $\triangle V_n$ in Table IV(b) is computed as the difference between V_{n_3} and V_{n_4}.

A positive $\triangle V_n$ indicates that the delay seen by the DLL is larger than it should be. As a result, the DLL causes the voltage to increase in to order to compensate for the extra delay. The opposite is true for negative $\triangle V_n$. For a certain delay fault, the larger the value of $\triangle V_n$, the easier it is to observe the error due to the fault. As shown in Table IV, the smallest value of $\triangle V_n$ is 17.3 mV, which can be easily measured off-chip. These results demonstrate our ability to test the clock timing circuit for delay faults. This is important to ensure its ability to give a true image about the operation and performance of the DUT.

8. CONCLUSIONS

In this article, we presented a methodology for testing high-performance pipelined circuits with slow-speed testers. In this technique, each pipeline stage is clocked using a separate clock generated from an on-chip clock timing circuit in test mode. The technique adds no extra hardware in the data path of the pipeline and therefore has no performance penalty.

A design for the clock timing circuit capable of achieving a timing resolution of 50 ps in 0.18 μm CMOS technology was presented. The design provides the ability to test the clock timing circuit itself.

The effectiveness of the technique was demonstrated using a 16-bit pipelined multiplier as a test vehicle. Simulations show that we are able to detect delay faults as small as 50 ps at an input clock frequency of 100 MHz. Simulations also prove our ability to test the clock timing circuit itself for delay faults.

REFERENCES

AGRAWAL, V. D. AND CHAKRABORTY, T. J. 1995. High-performance circuit testing with slow-speed testers. In *Proceedings of the International Test Conference*, 302–310.

ITRS 2001. International technology roadmap for semiconductor, 2001 edition.

KERKHOFF, H. ET AL. 2001. Design for delay testability in high-speed digital ics. *J. Elect. Test.: Theory Appl. 17*, 3–4 (June–August), 225–231.

MEHTA, M., PARMAR, V., AND E. SWARTZLANDER, J. 1991. High-speed multiplier design using multi-input counter and compressor circuits. In *Proceedings of the 10th IEEE Symposium on Computer Arithmetic*. IEEE Computer Society Press, Los Alamitos, Calif., 43–50.

MOYER, G. C. 1996. The Vernier Technique for Precise Delay Generation and Other Applications. Ph.D. thesis, The Department of Computer Engineering, North Carolina State University.

NEEDHAM, W., PRUNTY, C., AND YEOH, E. H. 1998. High volume microprocessor test escapes, an analysis of defects our tests are missing. In *Proceedings of International Test Conference*, 25–34.

NIGH, P. ET AL. 1997. So what is an optimal test mix? a discussion of the sematech methods experiment. In *Proceedings of International Test Conference*, 1037–1038.

NUMMER, M. AND SACHDEV, M. 2001. A methodology for testing high-performance circuits at arbitrarily low test frequency. In *Proceedings of IEEE VLSI Test Symposium*. IEEE Computer Society Press, Los Alamitos, Calif., 68–74.

NUMMER, M. AND SACHDEV, M. 2003. A DFT technique for testing high-speed circuits with arbitrarily slow testers. *J. Elect. Test.: Theory Appl. 19*, 3 (June), 299–314.

OHKUBO, N. ET AL. 1995. A 4.4 ns cmos 54 $\times$ 54-b multiplier using pass-transistor multiplexer. *IEEE J. Solid-State Circ. 30*, 3 (Mar.), 251–257.

SHASHANI, M. AND SACHDEV, M. 1999. A DFT technique for high-performance circuit testing. In *Proceedings of International Test Conference*, 267–285.

Received March 2003; revised July 2003; accepted July 2003