A Novel and Practical Control Scheme for Inter-Clock At-Speed Testing

著者	Furukawa Hiroshi, Wen Xiaoqing, Wang Laung-Terng, Sheu Boryau, Jiang Zhigang, Wu Shianling
journal or publication title	2006 IEEE International Test Conference
year	2007-02-05
URL	http://hdl.handle.net/10228/00007610

doi: info:doi/10.1109/TEST.2006.297641

A Novel and Practical Control Scheme for Inter-Clock At-Speed Testing

Hiroshi Furukawa, Xiaoqing Wen
Dept. of Computer Science and Electronics, Kyushu Institute of Technology, Iizuka 820-8502, Japan

Laung-Terng Wang, Boryau Sheu, Zhigang Jiang, Shianling Wu
SynTest Technologies, Inc., 505 S. Pastoria Ave., Suite 101, Sunnyvale, CA 94086, USA

Abstract

The quality of at-speed testing is being severely challenged by the problem that an inter-clock logic block existing between two synchronous clocks is not efficiently tested or totally ignored due to complex test control. This paper addresses the problem with a novel inter-clock at-speed test control scheme, featuring a compact and robust on-chip inter-clock enable generator design. The new scheme can generate inter-clock at-speed test clocks from PLLs, and is feasible for both ATE-based scan testing and logic BIST. Successful applications to industrial circuits have proven its effectiveness in improving the quality of at-speed testing.

1. Introduction

As process features shrink into the deep submicron (DSM) range and circuit speeds accelerate into the GHz domain, timing-related physical defects (often modeled as transition or path delay faults) are rapidly becoming the dominant cause of failing integrated circuits [1]. Nowadays it is mandatory to conduct *at-speed testing* in order to screen out chips that cannot operate at rated frequencies [2].

Different from conventional slow-speed structural testing, at-speed testing presents many new challenges to the test community, in terms of *test data volume*, *test application cost*, *test quality*, and *test control*. These issues, if not properly addressed, will make it impossible to achieve effective and efficient at-speed testing, as described bellow:

Test Data Volume: This issue is mainly because two test vectors are needed to detect one delay fault in at-speed testing. In addition, more stringent constraints need to be followed in at-speed test vector generation. These factors make it difficult to detect many delay faults with a single pair of test vectors, resulting in larger test data volume. Recently, various test compaction and compression techniques have been proposed to address this issue [3-6].

Test Application Cost: This issue can be very significant in at-speed testing if not properly handled. The major reason is the need for using functional clock frequencies to capture test responses during at-speed testing. As more and more circuits, including even low-price consumer chips, are designed to run at high frequencies, the conventional approach of using high-speed automatic test equipment (ATE) for at-speed testing is not sustainable from the test cost point of view. A more practical test application approach is to use on-chip clock sources for at-speed testing, in one of two forms. In the *partial* form, high-speed test clocks, e.g. capture clocks in scan testing, are provided from modified on-chip phase-locked loops (PLLs), while low-speed test clocks, e.g. shift clocks in scan testing, are provided from an external ATE [7]. In the *complete* form, all test clocks are provided internally from an on-chip test controller, as in logic built-in self-test (BIST) [8].

Test Quality: This is a critical issue that needs special attention. The fact is that the quality of at-speed testing cannot be measured by merely relying on one simple parameter, usually fault coverage, as often used in conventional slow-speed testing [9]. This is mainly due to three reasons: (1) The transition delay fault model has weak correlation with distributed small-delay defects. As a result, test vectors generated by conventional transition delay test generation methods often fail to detect small-delay defects. (2) The path delay fault model can better reflect the effect of defective delay increase in a circuit. However, this fault model suffers from severe challenges in terms of path selection quality and test generation efficiency. (3) Accurate evaluation of a delay test set is difficult, especially in the presence of noise and process variation. Recently, improving the quality of at-speed testing has been the focus of many studies [10-14].

Test Control: Test control, whose purpose is to issue all necessary test signals, such as clocks and scan enables, is another critical issue in at-speed testing. At-speed test control significantly differs from conventional slow-speed test control for the following reasons: (1) At-speed test control requires more complex waveforms than slow-speed testing. This is because, in order to conduct at-speed testing, a value transition for activating a delay fault should be launched at the start-point of a path and its response should be captured at the end-point at the rated clock speed, which can be very fast. (2) Timing relations among inter-dependent test control signals should be strictly followed, especially for a multi-clock circuit. As a result, many previous at-speed test control schemes, though logically sound, are not feasible for physical implementation because of high complexity, use of many timing-critical signals, or costly circuit modification for handling clock skews [15, 16]. Although there are some easy-to-implement test control schemes, they can only provide test control for *intra-clock logic blocks*, while totally ignoring *inter-clock logic blocks* [17-19]. Obviously, this results in incomplete at-speed testing, which significantly lowers test quality.

Generally, the combinational portion of a clock domain in a multi-clock circuit consists of two types of logic blocks: An *intra-clock logic block* is surrounded by flip-flops (FFs) driven by the same clock, while an *inter-clock logic* block exists between FFs driven by two different but synchronous clocks. Growing circuit sizes, shrinking process features, and accelerating circuit speeds are all making inter-clock logic blocks more and more significant. Obviously, ignoring inter-clock logic blocks in at-speed testing presents a tremendous risk for chip quality, which is not acceptable. Therefore, there is a strong and urgent need to establish an easy-to-implement test control scheme for inter-clock at-speed testing. This is the focus of this paper.

Previous test control schemes for inter-clock at-speed testing are too complex or too difficult for physical implementation [15, 16], mainly because they mix inter-clock test control with intra-clock test control and use the launch-on-shift approach that requires timing-critical scan enable signals. This observation leads to the key idea of this paper: *separating inter-clock test control from intra-clock test control and using the launch-on-capture approach that does not require any timing-critical scan enable signal*.

Based on the above key idea, this paper proposes a novel and practical inter-clock at-speed test control scheme, with the following characteristics:

- **Novel Clock Enable Generator**: A compact and robust on-chip inter-clock enable generator design is proposed, which creates two clock enable signals from two free-running synchronous functional clocks. The clock enable signals, when gated with the synchronous functional clocks, accurately generate all capture clock pulses required for the at-speed testing of the inter-clock logic block existing between the two functional clocks.
- **Easy Physical Implementation**: Inter-clock at-speed testing is conducted with the launch-on-capture approach. As a result, a single and non-timing-critical scan enable signal is sufficient for the scan testing of the entire circuit, significantly simplifying physical implementation.
- **Low Application Cost**: At-speed capture clock pulses are generated by on-chip circuitry through clock-gating with functional clocks from on-chip PLLs. There is no need to provide any high-speed test clock from ATE.
- **Easy Integration**: The new inter-clock at-speed test control scheme can be easily integrated with any existing easy-to-implement intra-clock at-speed test control scheme [17-19] at the top-level to achieve complete at-speed testing. This greatly improves test quality.
- **High Flexibility**: The new inter-clock at-speed test control scheme can be readily applied in both ATE-based scan testing and logic BIST.

The new inter-clock at-speed test control scheme has been successfully applied to large industrial circuits. As proven in layout and on tester, this scheme is easy to implement and can improve the at-speed test quality of multi-clock circuits required by today's system-on-a-chip (SoC) designs.

The rest of the paper is organized as follows: Section 2 describes the background. Section 3 presents the new inter-clock at-speed test control scheme. This section also discusses the integration of inter-clock and intra-clock at-speed test control schemes at the top-level for complete at-speed testing. Section 4 shows application results on industrial circuits, and Section 5 concludes the paper.

2. Background

2.1 Intra-Clock Logic and Inter-Clock Logic

Modern SoC circuits commonly use multiple clocks, often provided internally from on-chip PPLs. This can be due to the need for interfacing with different external systems, such as PCI and USB. It may also be due to the chip size being too large for a single fast clock to be effectively distributed over the entire circuit. Such a circuit usually consists of multiple clock domains, as defined bellow [20]:

Definition 1: A *clock domain* is the part of a circuit driven by either a single clock or multiple clocks that have constant phase relationships.

For example, a clock and its inverted clock or its derived divide-by-two clocks are considered as one clock domain. On the other hand, domains that have clocks with variable phase and time relationships are considered as different clock domains.

The clock skew within a clock domain is strictly managed, typically through clock tree synthesis (CTS). As a result, FFs in the same clock domain receive clock edges almost at the same time. On the other hand, a clock domain often has multiple synchronous internal clocks. That is, different FFs in a clock domain may operate at different but synchronous frequencies. An example is shown in Fig. 1, where the clock domian contains two synchronous clocks: a fast clock *FCK* and a slow clock *SCK*, running at 660MHz and 330MHz, respecitvely.

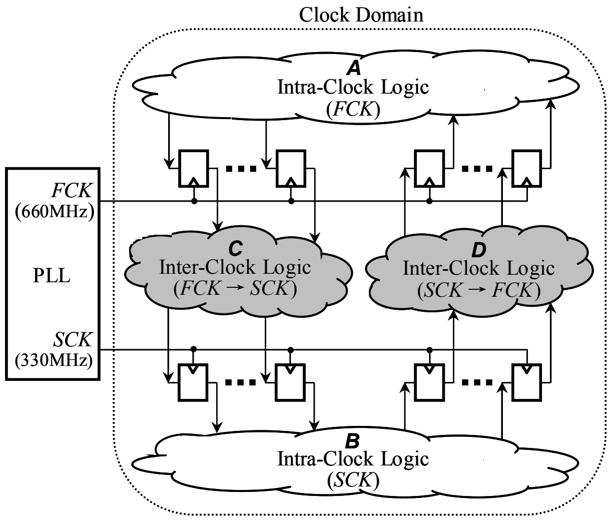


Fig. 1 Intra-Clock Logic and Inter-Clock Logic.

Therefore, a clock domain can be seen to consist of (1) FFs driven by a number of synchronous internal clocks and (2) combinational logic blocks surrounding those FFs. The combinational logic blocks can be further divided into two types as defined bellow:

Definition 2: *Intra-clock logic block* in a clock domain is the combinational logic portion existing between FFs driven by the same internal clock.

Definition 3: *Inter-clock logic block* in a clock domain is the combinational logic portion existing between FFs driven by two synchronous internal clocks.

For example, the clock domain shown in Fig. 1 consists of four combinational logic blocks: *A*, *B*, *C*, and *D*. Obviously, *A* and *B* are intra-clock logic blocks for internal clocks *FCK* and *SCK*, respectively. On the other hand, *C* and *D* are inter-clock logic blocks, with *C* being from *FCK* to *SCK* and *D* being from *SCK* to *FCK*.

Note that data transfer between two asynchronous clock domains needs to be made through synchronizers or dual-port random access memories (RAMs). Due to glitch concerns, functional logic is usually not placed between two clock domains with FF-based synchronizers.

From the discussions above, it is clear that the combinational portion in a circuit consists of intra-clock logic blocks and inter-clock logic blocks in all clock domains. Therefore, all intra-clock logic blocks as well as all inter-clock logic blocks need to be tested in order to achieve complete at-speed testing for the circuit.

2.2 Test Control for At-Speed Testing

2.2.1 General Concept

Logic testing, both slow-speed and at-speed, are usually based on full-scan design, in which all functional FFs in a circuit are replaced with scan FFs. A full-scan circuit operates in either *shift* or *capture* mode, selectable by *scan enable* (*SE*) signals. In shift mode (*SE* = 1), scan FFs form scan chains that operate as shift registers, through which a new test vector is shifted-in and a test response is shifted-out. In capture mode (*SE* = 0), scan FFs operate as functional FFs and catch the test response corresponding to the applied test vector. Such scan-based testing is the foundation of both ATE-based testing and logic BIST.

The basic concept of test control for scan-based testing is illustrated in Fig. 2. Basically, scan enable (*SE*) signals and clock (*CK*) pulses should be provided in proper order and with right timing relations in both shift and capture modes, so as to guarantee correct shift and capture operations.

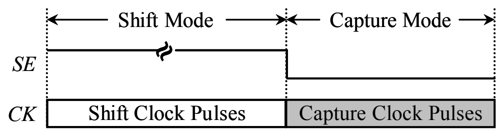


Fig. 2 Test Control for Scan-Based Testing.

Generally, test control for slow-speed testing targeted at structural defects can be easily conducted for both intra-clock and inter-clock logic blocks. However, test control for at-speed testing targeted at timing-related defects is significantly more difficult, especially for multi-clock SoC circuits. As a result, test control has become a major challenge in pursuing high-quality at-speed testing.

2.2.2 Control Tasks for At-Speed Testing

Test control for scan-based testing is realized by controlling scan enable and clock signals to conduct a series of control tasks, which are repeated for each test vector. For at-speed testing, there are three types of control tasks, *shift*, *launch*, and *capture*, as illustrated in Fig. 3, whose circuit is the full-scan version of the circuit shown in Fig. 1.

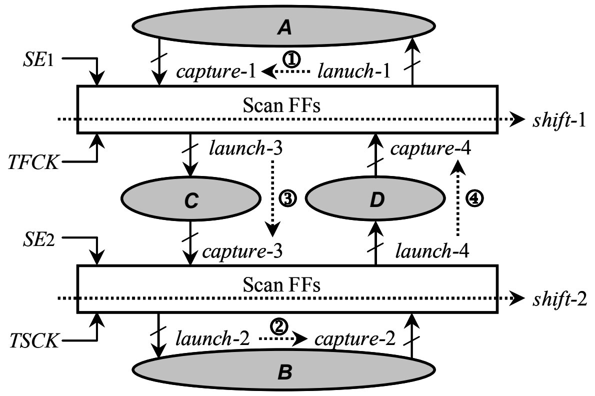


Fig. 3 Control Tasks for At-Speed Testing.

In Fig. 3, there are two shift tasks, *shift*-1 and *shift*-2, each for one scan chain. A shift task is to operate a scan chain as a shift register to load a test vector or unload a test response, by setting the corresponding *SE* signal to 1 and issuing the same number of shift clock pulses as the length of the corresponding scan chain. Since shift tasks are independent from each other, they can be conducted simultaneously without considering their inter-relations. In addition, a shift task can be conducted at any speed, usually a slow speed for test power reduction, even for at-speed testing. As a result, test control for shift tasks is easy, and will not be further considered in this paper.

After shift tasks are finished, a pair of launch and capture tasks should be conducted in order to apply at-speed testing to each logic block. There are two types of launch-capture pairs as defined bellow:

Definition 4: An *intra-clock launch-capture pair* is used for the at-speed testing of an intra-clock logic block, while an *inter-clock launch-capture pair* is used for the at-speed testing of an inter-clock logic block.

In Fig. 3, there are two intra-clock launch-capture pairs, ① and ②, as well as two inter-clock launch-capture pairs, ③ and ④. For example, ① (<*launch*-1, *capture*-1>) is the intra-clock launch-capture pair for *A*, where *launch*-1 is to create value transitions at the inputs of *A* and *capture*-1 is to capture the test response from the outputs of *A* at the rated speed of the clock *FCK* in order to achieve intra-clock at-speed testing. On the other hand, ③ (<*launch*-3, *capture*-3>) is the inter-clock launch-capture pair for *C*, where *launch*-3 is to create value transitions at the inputs of *C* and *capture*-3 is to capture the test response from the outputs of *C* following the exact inter-clock relation from *FCK* to *SCK* in order to achieve inter-clock at-speed testing.

Note that the free-running functional clocks *FCK* and *SCK* shown in Fig. 1 are replaced with test clocks *TFCK* and *TSCK*, respectively, as shown in Fig. 3. *TFCK* and *TSCK*, as well as scan enable signals *SE*1 and *SE*2, should be properly provided from a test control scheme in order to accomplish all test control tasks. In this sense, *FCK* and *SCK* can be seen as inputs to a test control scheme, while *TFCK* and *TSCK* are outputs from the test control scheme.

2.3 Previous At-Speed Test Control Schemes

There are two approaches to test control for scan-based at-speed testing: *launch-on-shift* and *launch-on-capture* [21, 22]. The major difference is how a launch task is conducted, i.e. how value transitions are created at the inputs of a logic block for delay testing. Launch-on-shift creates value transitions by the difference between the next-to-last and the last shifted-in values in shift mode, while launch-on-capture creates value transitions by the difference between the last shifted-in values in shift mode and the first-captured values in capture mode.

In the following, we briefly describe three test control schemes, focusing on how launch-capture pairs are conducted. The *launch aligned skewed-load* scheme and the *capture aligned skewed-load* scheme are based on the launch-on-shift approach, while the *double-capture* scheme is based on the launch-on-capture approach.

Launch Aligned Skewed-Load Scheme: This scheme is illustrated in Fig. 4, corresponding to the circuit shown in Fig. 3. This scheme aligns the last shift clock pulse *S*1 of *TFCK* with the last shift clock pulse *S*2 of *TSCK* to create input value transitions simultaneously for both *TFCK* and *TSCK*, and then issues at-speed capture clock pulses *C*1 and *C*2 to capture corresponding test responses at the rated speeds of *FCK* and *SCK*, respectively [15].

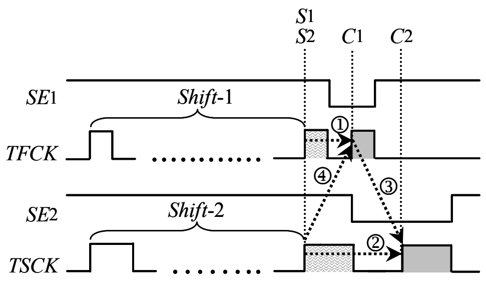


Fig. 4 Launch Aligned Skewed-Load Scheme.

Although this scheme can accomplish all launch-capture pairs ① ~ ④ to achieve complete at-speed testing, it requires two *SE* signals, *SE*1 and *SE*2, which may be timing-critical since both of them need to settle from 1 to 0 during the system clock periods of *FCK* and *SCK*, respectively. This makes already-hard physical implementation even more difficult. In addition, the complexity of this scheme increases rapidly with the number of clocks.

Capture Aligned Skewed-Load Scheme: This scheme is illustrated in Fig. 5, corresponding to the circuit shown in Fig. 3. This scheme aligns the at-speed capture clock pulses *C*1 and *C*2 following the last shift clock pulses *S*1 and *S*2 so as to capture corresponding test responses simultaneously for the clocks *TFCK* and *TSCK* , respectively [16].

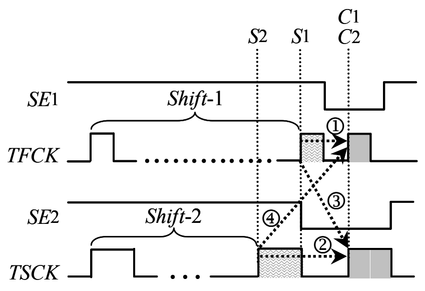


Fig. 5 Capture Aligned Skewed-Load Scheme.

This scheme can also accomplish all launch-capture pairs ① ~ ④ to achieve complete at-speed testing, but also suffers from increased difficulty in physical implementation as in the launch aligned skewed-load scheme. In addition, frequency pre-scaling may have to be conducted in some cases, making it hard to achieve real at-speed testing. Moreover, costly circuit modification may be needed in order to prevent clock skews from disturbing test responses.

Double-Capture Scheme: This scheme is illustrated by the example of Fig. 6, corresponding to the circuit shown in Fig. 3. This scheme uses two at-speed capture clock pulses for each clock: *C*1/*C*2 for *TFCK* and *C*3/*C*4 for *TSCK* [19]. The input value transitions for *TFCK* (*TSCK*) are launched by the difference between the values loaded by the last-shift pulse *S*1 (*S*2) and the values captured by the first-capture pulse *C*1 (*C*3). Note that capture operations are conducted for both clocks in the same capture window (*SE* = 0).

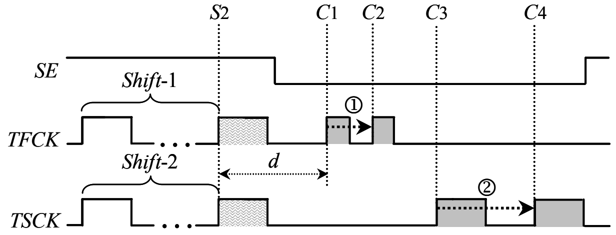


Fig. 6 Double-Capture Scheme.

The biggest advantage of this scheme is its easy physical implementation. This is because only one scan enable signal *SE* is used, which is not timing-critical if *d* is made long enough. Some other schemes can achieve similar effects [17, 18]. However, the biggest disadvantage of these schemes is that they can only achieve intra-clock at-speed testing. As shown in Fig. 6, this is because only intra-clock launch-capture pairs, ① and ②, are accomplished, while inter-clock launch-capture pairs, ③ and ④, are totally ignored. That is, these schemes cannot provide test control for inter-clock at-speed testing. As a result, complete at-speed testing cannot be achieved for a multi-clock circuit.

Therefore, there is a strong need for an easy-to-implement inter-clock at-speed test control scheme. Combining it with any easy-to-implement intra-clock at-speed test control scheme, such as the double-capture scheme [17-19], forms an integrated test control scheme to achieve complete at-speed testing for all logic blocks in a circuit. As a result, the at-speed test quality can be significantly improved.

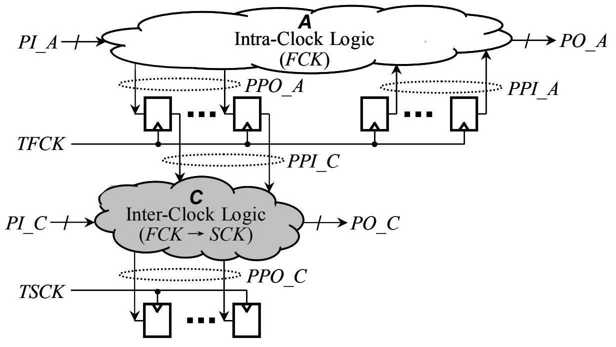
3. Inter-Clock At-Speed Test Control Scheme

3.1 Basic Idea

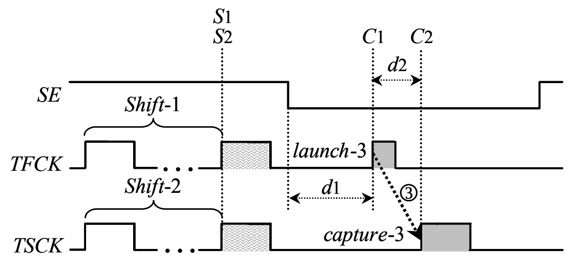
As described in 2.3, previous test control schemes for complete at-speed testing suffer from high complexity and difficult physical implementation. This is mainly because they mix inter-clock test control with intra-clock test control and use timing-critical scan enable signals due to the launch-on-shift approach. This observation leads to the basic idea of this paper as follows:

- Separate inter-clock test control from intra-clock test control in order to simplify test control operations.
- Use the launch-on-capture approach in order to avoid the use of any timing-critical scan enable signal.

The basic idea is illustrated in Fig. 7, using the example of at-speed testing for the inter-clock logic block *C* in Fig. 1.



(a) Circuit portions related to at-speed testing of *C*



(b) Proposed at-speed test control waveforms for *C*

Fig. 7 Basic Idea of Inter-Clock At-Speed Test Control.

The two shift tasks in Fig. 7 (b), *shift*-1 and *shift*-2, are conducted with the same shift clock in this example. However, it is allowed to use any shift control method with any number of shift clocks running at any frequencies. After shift tasks are finished, the launch-capture pair ③ (<*launch*-3, *capture*-3>) in Fig. 3 should be performed in order to conduct at-speed testing for the inter-clock logic block *C* in Fig. 7 (a). Based on the basic idea described above, this paper proposes a new inter-clock at-speed test control scheme that uses the test control waveforms of Fig. 7 (b) to accomplish the launch-capture pair ③ as follows:

Launch Control: *launch*-3 is conducted by the capture clock pulse *C*1 of *TFCK*, as shown in Fig. 7 (b). That is, value transitions at the inputs *PPI_C* are created by the difference between the values loaded by the last-shift pulse *S*1 of *TFCK* and the values captured by the capture pulse *C*1 of *TFCK*. Since the launch-on-capture approach is used, *d*1 can be made long enough to allow the use of a single and non-timing-critical scan enable signal *SE* for all scan chains. This significantly simplifies physical implementation.

Capture Control: *capture*-3 is conducted by the capture clock pulse *C*2 of *TSCK*, as shown in Fig. 7 (b). For at-speed testing of the inter-clock logic block *C*, it is required that *d*2 be set by following the exact clock-triggering relation from *FCK* to *SCK* for transferring data as in functional operations. As an example, Fig. 7 (b) shows the positive-to-positive single-cycle at-speed test requirement.

In the following, details on the new inter-clock at-speed test control scheme are described by using the example shown in Fig. 7. The focus is on how capture pulses *C*1 and *C*2 for test clocks *TFCK* and *TSCK* are generated from free-running functional clocks *FCK* and *SCK* (PLL clocks as shown in Fig. 1), respectively, by using new on-chip circuitry so that the requirements on *d*1 and *d*2 are satisfied.

3.2 General Architecture

Fig. 8 shows the general architecture of the new inter-clock at-speed test control scheme for the example of Fig. 7.

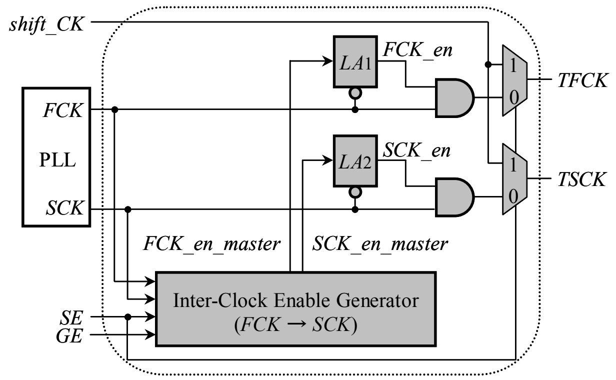


Fig. 8 Architecture of Inter-Clock Test Control Scheme.

In shift mode (*SE* = 1), *shift_CK*, either generated internally or provided externally, is used as the shift clock for both *TFCK* and *TSCK*. Note that *shift_CK* can be a slow clock for reducing power dissipation during test application.

In capture mode (*SE* = 0), the inter-clock enable generator creates *master* clock enable signals *FCK_en_master* and *SCK_en_master*, the two latches generate *final* clock enable signals *FCK_en* and *SCK_en*, and the two AND gates conduct clock-gating to generate test clocks *TFCK* and *TSCK* from functional PLL clocks *FCK* and *SCK*, respectively. *SE* is a normal scan enable signal as shown in Fig. 7 (b), while *GE* is a generator enable signal used to indicate if the inter-clock enable generator is active or not. *GE* = 1 means that *FCK_en_master* and *SCK_en_master* are active clock enable signals, while *GE* = 0 will inactivate them by holding them at 0.

Note that circuitry for switching between test and normal modes is not shown in Fig. 8 for clarity. As far as test concerned, only one multiplexor and one AND gate are added to a functional clock line, while the rest of the circuitry in the new inter-clock at-speed test control scheme, especially the inter-clock enable generator, does not cause any performance degradation. In addition, this new scheme can be used in both ATE-based testing and logic BIST.

Obviously, the key part of the new inter-clock at-speed test control scheme is the inter-clock enable generator for creating master clock enable signals. Detailed information on this part is described in the following section.

3.3 Inter-Clock Enable Generator Design

Fig. 9 shows the schematic of the proposed inter-clock enable generator design used in Fig. 8.

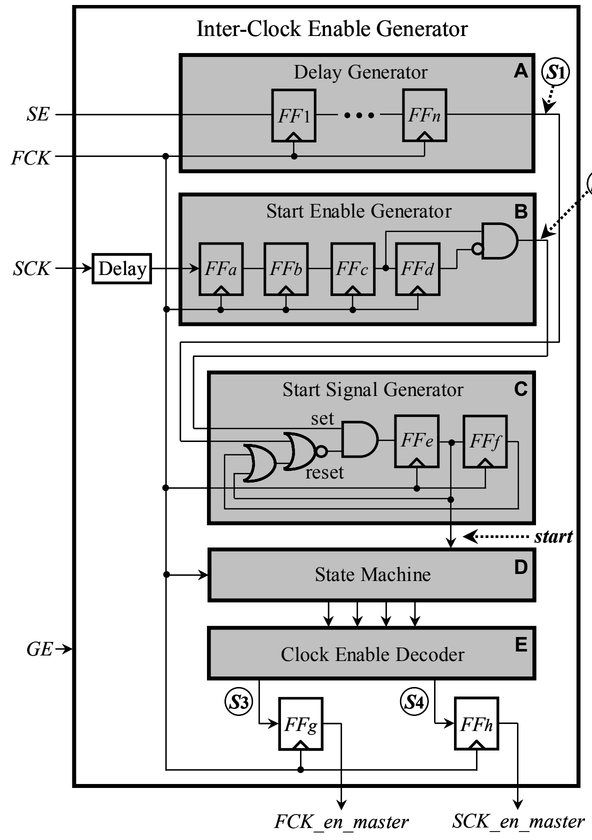


Fig. 9 Schematic of Inter-Clock Enable Generator.

The inter-clock enable generator consists of a delay generator (A), a start enable generator (B), a start signal generator (C), a state machine (D), and a clock enable decoder (E), in the operational order. Its inputs are two free-running synchronous clocks *FCK* (fast clock) and *SCK* (slow clock) as well as two enable signals *SE* and *GE*, while its outputs are two master clock enable signals *FCK_en_master* and *SCK_en_master*.

The inter-clock enable generator is a compact synchronous circuit driven by the fast clock *FCK*. Note that slow clock *SCK* is used as data input for the start enable generator. The operation of this circuit is shown in Fig. 10. As to be described, many design techniques are used to make this circuit functionally accurate and operational robust.

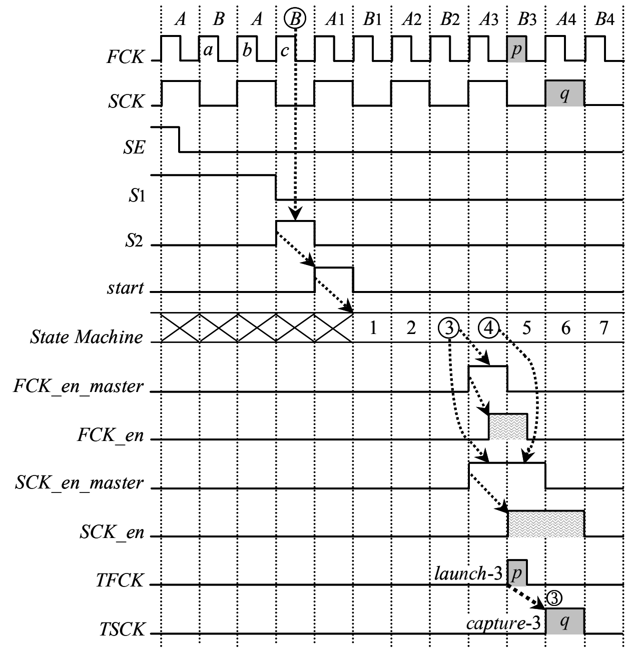


Fig. 10 Waveforms of Inter-Clock Enable Generator.

A. Delay Generator

The delay generator is basically an *n*-stage shift register driven by the fast clock *FCK*. Its purpose is to delay the effective arrival time of the falling edge of the scan enable signal *SE* at least by the time of (n-1) × T_{FCK}, where T_{FCK} is the period of *FCK*. This delay is used to make d_1 in Fig. 7 (b) long enough so that *SE* becomes non-timing-critical. This will significantly ease physical implementation.

For example, Fig. 10 shows the case where *n* is set to 3. That is, the *SE*→0 event takes a little more than 2 *FCK* clock periods to bring the internal signal S_1 to 0. It is this S_1→0 event that marks the actual start of a series of capture operations. This way, d_1 can be made as long as required.

Generally, d_1 needs to be made longer than the maximum delay of *SE* when *SE* is designed as a data signal, not as a clock signal requiring CTS. This maximum delay is then used to figure out the necessary value for *n*. For example, *n* was set to 8 for a 533MHz industrial circuit with 11.1M gates and 404.9K FFs. This created at least 13ns for d_1 in Fig. 7 (b), long enough to make *SE* non-timing-critical.

B. Start Enable Generator

The start enable generator consists of a 4-stage shift register, composed of $FF_a \sim FF_d$, as shown in Fig. 9. Its purposes are (1) to avoid *metastability* and (2) to provide a *re-timing* mechanism so that the state machine can be started properly.

Metastability may occur if the slow clock *SCK* is applied directly to the data input of an FF driven by the fast clock *FCK*. Two measures are taken against metastability: one is to slightly delay *SCK* before sending it to the shift register driven by *FCK*, and the other is to add a synchronizer composed of two FFs, FF_a and FF_b, on the input side of the shift register, as shown in Fig. 9.

The re-timing mechanism is realized by using a 2-bit shift register composed of FF_c and FF_d, as shown in Fig. 9. The outputs of the two FFs are decoded by the AND gate to generate the start enable signal S_2 so that it becomes 1 at the falling-edge of *SCK*, as shown in Fig. 10. This signal is used as a timing base to guarantee that the *start* signal for the state machine becomes 1 at the rising-edge of *SCK*.

C. Start Signal Generator

The start signal generator is designed as shown in Fig. 9. When S_1 is 1, *start* is 0 and the state machine does not operate. $S_1 \rightarrow 0$ indicates the start of capture operations by releasing the reset input of the AND gate. After S_1 has become 0, if the start enable signal S_2 becomes 1, *start* will become 1. The feedbacks from the two FFs make *start* stay at 1 only for one *TCK* period, as shown in Fig. 10. This *start* signal is used to start the state machine.

D. State Machine

The state machine is basically a 4-bit counter, whose operation is shown in Fig. 11. It resets to state 0, and starts to count up from state 1 after the *start* signal becomes 1. After all states are traversed, it stops at state 0. The purpose of the state machine is to provide a means to accurately identify the phase and time relations of the clock pulses between the two input functional clocks *FCK* and *SCK*.

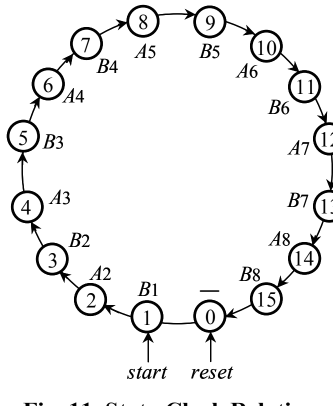


Fig. 11 State-Clock Relations.

In the example shown in Fig. 10, the state machine starts to count at the first rising-edge of *TCK* after *start* becomes 1. As a result, the dynamic relations between the fast clock *FCK* and the slow clock *SCK* can be fully represented by the current state of the state machine.

As shown in Fig. 11, states 1, 2, 3, 4, …, 15 correspond to the clock relations B_1, A_2, B_2, A_3, …, B_8, respectively. As shown in Fig. 10, *A* corresponds to the relation that *FCK* has a full-period and *SCK* has a half-period of 1, while *B* corresponds to the relation that *FCK* has a full-period and *SCK* has a half-period of 0. Obviously, by selectively decoding these states, master clock enable signals can be created at required timing and for required duration.

E. Clock Enable Decoder

The clock enable decoder creates two master clock enable signals, *FCK_en_master* and *SCK_en_master*, as shown in Fig. 9. The start time and duration of each master clock enable signal are determined by decoding the states of the state machine.

In the example shown in Fig. 10, the clock enable decoder reacts to the state 3 to create a pulse (not shown in Fig. 10) with the duration of one *TCK* period on its S_3 output. This pulse, delayed by FF_g for one *TCK* period, becomes the master clock enable signal *FCK_en_master*. Similarly, the clock enable decoder reacts to the state 3 and the state 4 to create a pulse (not shown in Fig. 10) with the duration of two *TCK* periods on its S_4 output. This pulse, delayed by FF_h for one *TCK* period, becomes the master clock enable signal *SCK_en_master*.

This way, two master clock enable signals, *FCK_en_master* and *SCK_en_master*, are accurately generated by the inter-clock enable generator of Fig. 9. Then, as shown in Fig. 8, the final clock enable signal *FCK_en* is obtained by delaying *FCK_en_master* by half period of *FCK* with the latch LA_1, while the final clock enable signal *SCK_en* is obtained by delaying *SCK_en_master* by half period of *SCK* with the latch LA_2. This is also shown in Fig. 10.

As shown in Fig. 10, the clock enable pulse on *FCK_en* can accurately pick out the *FCK* clock pulse *p* for the test clock *TFCK*, and the clock enable pulse on *SCK_en* can accurately pick out the *SCK* clock pulse *q* for the test clock *TSCK*. As a result, two capture pulses are generated for the test clocks *TFCK* and *TSCK* so that the inter-clock at-speed test requirements shown in Fig. 7 (b) are satisfied. Therefore, the at-speed testing of the inter-clock logic block *C* (*FCK*→*SCK*) can be successfully achieved.

The characteristics of the inter-clock enable generator are summarized as follows:

- **High Flexibility**: Although different inter-clock logic blocks with different inter-clock relations need to use different inter-clock enable generators, only the clock enable decoder part needs to be slightly changed. That is, it is only necessary to make a clock enable decoder to decode the necessary states so that a master clock enable signal becomes 1 at the time and for the duration that are required by a specific inter-clock relation. In addition, multi-cycle paths can also be easily handled by slightly changing the decoding logic to increase the duration of a master clock enable pulse.

- **Wide Application**: The inter-clock enable generator design can be used in both ATE-based testing and logic BIST. Since only on-chip PLLs are needed to generate fast test clocks, there is no need to use a high-speed tester for at-speed testing. This helps reduce test costs.
- **Robust Operation**: Subtle issues such as metastability and clock skews are all carefully considered and measures against them are properly taken in logic design. This guarantees the robust operation of the inter-clock enable generator, even in the case of large and complicated industrial circuits.
- **Easy Implementation**: Since subtle issues such as metastability and clock skews are all handled by logic design techniques, no extra burden is placed on layout design in order to make the inter-clock enable generator work properly. This simplifies physical implementation.
- **Low Overhead**: The inter-clock enable generator design is compact. It contains about 20 FFs and only a small number of logic gates. Its area overhead is roughly 124 equivalent 2-input NAND gates for a typical configuration with an 8-stage delay generator.

3.4 Use of Multiple Inter-Clock Enable Generators

A multi-clock circuit usually contains multiple inter-clock logic blocks that should be targeted in at-speed testing. Basically, the at-speed testing of one inter-clock logic block needs one inter-clock enable generator. An example is shown in Fig. 12, in which there are 3 inter-clock enable generators, corresponding to three synchronous clock pairs: *FCK*1→*SCK*1, *SCK*1→*FCK*1, and *FCK*2→*SCK*2.

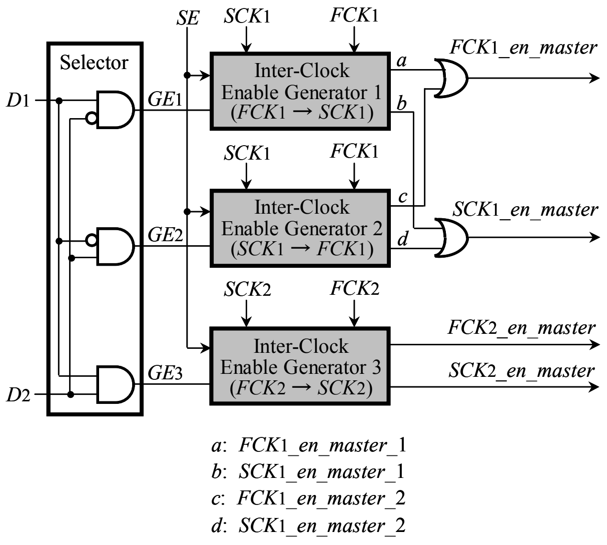


Fig. 12 Clock Enable Generator Selection.

Although it is possible to share some part of inter-clock enable generation circuitry among different inter-clock enable generators, this often increases control complexity and introduces more delay on functional clock lines. Given the fact that an inter-clock enable generator is small, it is advisable to use multiple inter-clock enable generators, one for each inter-clock logic block. In this case, a selection mechanism needs to be provided to activate the inter-clock enable generators one by one during at-speed testing.

In the example shown in Fig. 12, the selector is used for this purpose. *D*1 and *D*2 are decoded into 3 generator enable signals, *GE*1, *GE*2, and *GE*3, such that only one of them is 1 at any time. The activated inter-clock enable generator is used for test control, and the at-speed testing of the corresponding inter-clock logic block is achieved.

3.5 Top-Level Integration

The basic idea of this paper is to separate inter-clock test control from intra-clock test control in order to simplify test control operations. Therefore, in order to achieve complete at-speed testing, it is necessary to combine the new inter-clock at-speed test control scheme with an easy-to-implement intra-clock at-speed test control scheme, such as the double-capture scheme [17-19] as described in 2.3. The task of combining inter-clock and intra-clock test control schemes is conducted in top-level integration. An example is shown in Fig. 13, in which there are one inter-clock enable generator and one intra-clock enable generator, and the selection between them is made by the signal *S*.

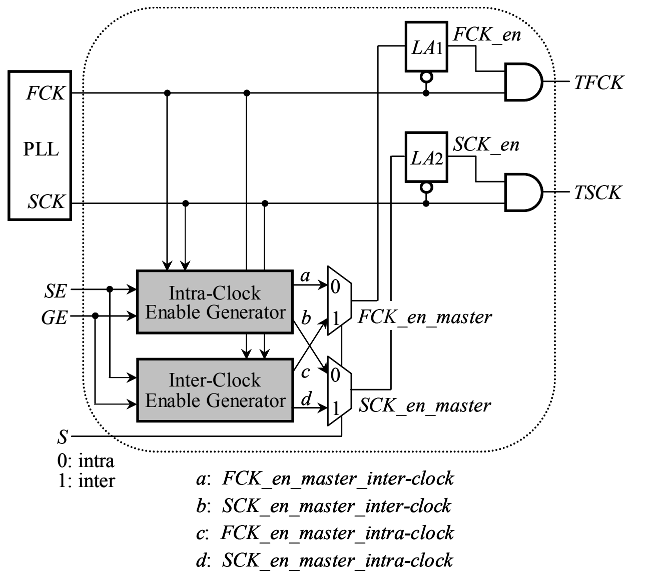

LA1
FCK_en
FCK
TFCK
PLL
LA2
SCK_en
SCK
TSCK
SE
GE
Intra-Clock Enable Generator
a
b
0
1
FCK_en_master
Inter-Clock Enable Generator
c
d
0
1
SCK_en_master
S
0: intra
1: inter
a: FCK_en_master_inter-clock
b: SCK_en_master_inter-clock
c: FCK_en_master_intra-clock
d: SCK_en_master_intra-clock

Fig. 13 Top-Level Integration.

Note that, different from inter-clock at-speed test control, it is possible to use only one intra-clock enable generator for at-speed testing of all intra-clock logic blocks, based on the multi-capture technique [19] that generates at-speed capture pulses for all intra-clock logic blocks in a serial manner in the same capture window (*SE* = 1). Some other methods can achieve similar effects [17, 18].

3.6 Design Flow

The design flow for complete at-speed testing is shown in Fig. 14, which consists of three sub-flows: a netlist flow, an ATPG flow for ATE-based scan testing, and a fault simulation flow for logic BIST. The netlist flow is the basic flow that needs to be conducted for both ATE-based scan testing and logic BIST.

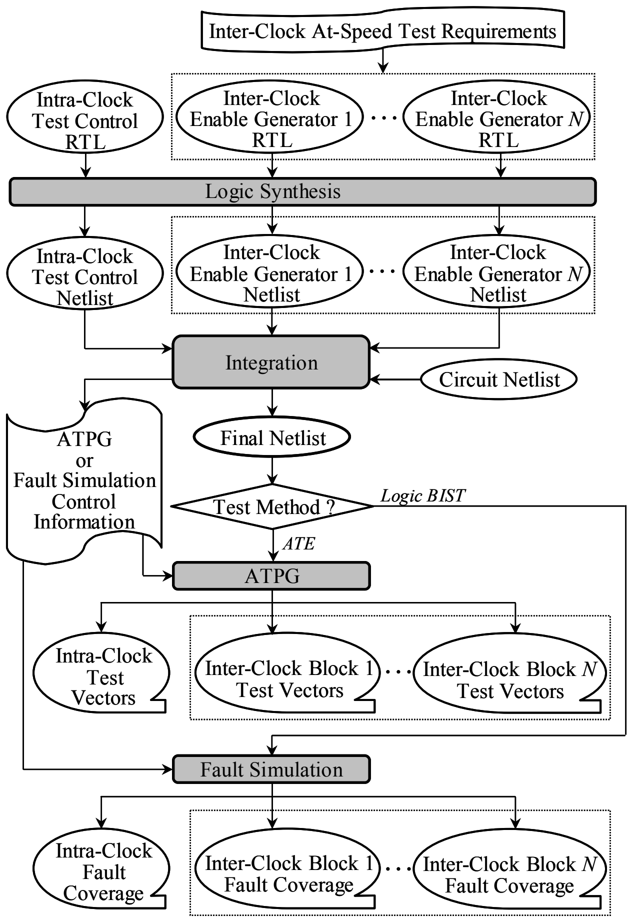

Inter-Clock At-Speed Test Requirements
Intra-Clock Test Control RTL
Inter-Clock Enable Generator 1 RTL
Inter-Clock Enable Generator N RTL
Logic Synthesis
Intra-Clock Test Control Netlist
Inter-Clock Enable Generator 1 Netlist
Inter-Clock Enable Generator N Netlist
Integration
Circuit Netlist
ATPG or Fault Simulation Control Information
Final Netlist
Test Method ?
Logic BIST
ATE
ATPG
Intra-Clock Test Vectors
Inter-Clock Block 1 Test Vectors
Inter-Clock Block N Test Vectors
Fault Simulation
Intra-Clock Fault Coverage
Inter-Clock Block 1 Fault Coverage
Inter-Clock Block N Fault Coverage

Fig. 14 Design Flow.

In the netlist flow, in addition to the intra-clock test control RTL design, RTL designs for all necessary inter-clock enable generators need to be prepared according to inter-clock at-speed test requirements. These RTL designs are synthesized into separate netlists. Then, at the integration stage, these test-oriented netlists are combined with the circuit netlist into the final netlist. Additional circuitry needed in inter-clock and intra-clock at-speed test control schemes are also added at the integration stage. In addition, control information necessary for the ATPG flow or the fault simulation flow is also generated at this stage.

In the ATPG flow, test generation is conducted for ATE-based scan testing. For all intra-clock logic blocks, it is possible to use only one ATPG run. However, in some cases, PLL-based multi-run ATPG is conducted, with one ATPG run for all intra-clock logic blocks related to one PLL. On the other hand, each inter-clock logic block requires its own ATPG run, and test vector sets for different inter-clock logic blocks are generated separately.

The fault simulation flow is similar to the ATPG flow in that each inter-clock logic block needs its own fault simulation run. The fault simulation flow is used for logic BIST, where test vectors are generated by a pseudo-random pattern generator (PRPG), and it is only necessary to fault-grade the test vectors.

3.7 Test Generation and Fault Simulation

As described above, in addition to the new inter-clock at-speed test control scheme, deterministic test vectors need to be generated for ATE-based scan testing, and random test vectors need to be fault-graded for logic BIST, in order to conduct inter-clock at-speed testing.

Fig. 15 shows the circuit model for test generation and fault simulation, corresponding to the at-speed testing of the inter-clock logic block *C* as shown in Fig. 7 (a). Obviously, only combinational logic blocks *A* and *C* need to be included for test generation and fault simulation of *C*.

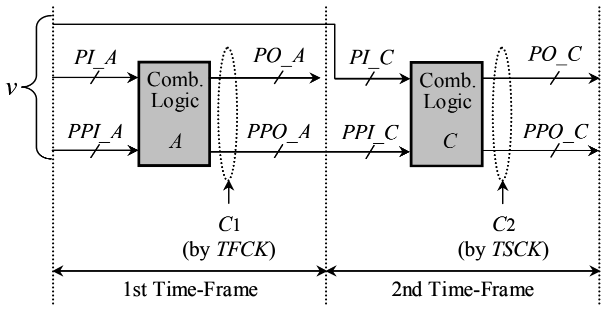


Fig. 15 Circuit Model for Test Generation & Fault Simulation.

Note that the circuit model shown in Fig. 15 is independent of the inter-clock at-speed test control scheme used. That is, the results of test generation or fault simulation are valid as long as at-speed capture pulses *C*1 and *C*2 are properly provided as shown in Fig. 7 (b). This separation of test generation and fault simulation from the underlying inter-clock at-speed test control scheme results in more flexibility in a test design flow.

4. Application Results

The new inter-clock at-speed test control scheme has been applied to a number of large-scale industrial designs, all with successful tape-outs. The inter-clock enable generator design and other circuitry necessary for the proposed test control scheme have been verified in logic design and layout implementation. Test vectors generated for inter-clock at-speed testing have also been verified on testers.

Due to page limitation, we only report the application results on one of the large industrial designs in the following. This design is in production now, and its circuit statistics are shown in Table 1.

Table 1 Circuit Statistics

# of Gates	11.1M
# of FFs	404.9K
# of Scan Chains	32
Max. Chain Length	13598
No. of Clock Domains	11
Min. Frequency	66MHz
Max. Frequency	533MHz

Intra-clock logic blocks were tested by logic BIST with top-up ATPG vectors. The transition delay fault coverage for all intra-clock logic blocks was 96.9% with 64K random vectors in logic BIST and 8183 vectors in top-up ATPG.

Inter-clock logic blocks were tested by the new inter-clock at-speed test control scheme with ATPG vectors, and the results are shown in Table 2. In this case, 6 inter-clock logic blocks, $A \sim F$, were targeted. For example, A is a logic block from a 100MHz clock to a 300MHz clock, and contains 36858 transition delay faults. Table 2 also shows the information on the number of ATPG vectors, fault coverage, and CPU time. Furthermore, the area overhead in this application was mainly due to 6 inter-clock enable generators, each containing 20 FFs and consuming an area of roughly 124 equivalent 2-input NAND gates.

Table 2 Application Results

Inter-Clock Logic Blocks	From (MHz)	To (MHz)	# of Faults	ATPG		
				# of Vec.	Fault Cov.	CPU (h:m)
A	100	300	36858	232	86.4	4:30
B	133	533	8350	32	100	0:15
C	133	266	4942	36	100	0:14
D	533	133	1940	9	100	0:10
E	266	533	732	9	100	0:10
F	266	133	64	3	100	0:09

Table 2 shows that some inter-clock logic blocks, such as A, was quite large, and ignoring them in at-speed testing would have been a tremendous risk to chip quality. This was why inter-clock at-speed testing was made mandatory by the user company of this design. The new inter-clock at-speed test control scheme successfully provided a novel and practical way to improve the overall chip quality by making inter-clock at-speed testing easy and efficient.

5. Conclusions

The paper addressed the test control problem in at-speed testing with a novel inter-clock at-speed test control scheme, featuring a compact, robust, and easy-to-implement inter-clock enable generator design. By combining this scheme with any existing intra-clock at-speed test control scheme, complete at-speed testing of the entire circuit can be efficiently achieved, which significantly improves the quality of at-speed testing. The effectiveness of the new inter-clock at-speed test control scheme has been proven by successful applications to large industrial circuits.

References

[1] T. M. Mak, A. Krstic, K. Cheng, and L. Wang, "New Challenges in Delay Testing of Nanometer, Multigigahertz Designs," *IEEE Design and Test of Computers*, Vol. 21, No. 3, pp. 241-247, 2004.

[2] G. Aldrich and B. Cory, "Improving Test Quality and Reducing Escapes," *Proc. Fabless Forum*, pp**.** 34-35, 2003.

[3] L.-T. Wang, X. Wen, H. Furukawa, F. Hsu, S. Lin, S. Tsai, K. S. Abdel-Hafez, and S. Wu, "VirtualScan: A New Compressed Scan Technology for Test Cost Reduction," *Proc. Int'l Test Conf.*, pp. 916-925, 2004.

[4] C. Barnhart, V. Brunkhorst, F. Distler, O. Farnsworth, B. L. Keller, B. Könemann, and A. Ferko, "The OPMISR: Foundation for Compressed ATPG Vectors," *Proc. Int'l Test Conf.*, pp. 748-757, 2001.

[5] J. Rajski, M. Kassab, N. Mukherjee, N. Tamarapalli, J. Tyszer, and J. Qian, "Embedded Deterministic Test for Low-Cost Manufacturing," *IEEE Design & Test of Computers*, Vol. 20, No. 5, pp. 58-66, 2003.

[6] P. Wohl, J. Waicukauski, S. Patel, and M. Amin, "Efficient Compression and Application of Deterministic Patterns in a Logic BIST Architecture," *Proc. Design Automation Conf.*, pp. 934-939, 2003.

[7] M. Beck, O. Barondeau, M. Kaibel, F. Poehl, X. Lin, R. Press, "Logic Design for On-Chip Test Clock Generation - Implementation Details and Impact on Delay Test Quality," *Proc. Proc. Design, Automation and Test in Europe*, pp. 56-61, 2005.

[8] J. Savir and P. Bardell, "Built-in Self-Test: Milestones and Challenges," *Gordon and Breach Science Publishers, VLSI Design*, Vol. 1, No. 1, pp. 23-44, 1993.

[9] J. Rearick, "Too Much Delay Fault Coverage Is a Bad Thing," *Proc. Int'l Test Conf.*, pp. 624-633, 2001.

[10] S. Mitra, E. Volkerink, E. McCluskey, and S. Eichenberger, "Delay Defect Screening Using Process Monitor Structures," *Proc. VLSI Test Symp.*, pp. 43-52, 2004.

[11] Y. Sato, S. Hamada, T. Maeda, A. Takatori, Y. Nozuyama, and S. Kajihara "Invisiable Delay Quality – SDQM Model Lights Up What Could Not Be Seen," *Proc. Int'l Test Conf.*, Paper 47.1, 2005.

[12] P. Gupta and M. S. Hsiao, "High Quality ATPG for Delay Defects," *Proc. Int'l Test Conf.*, pp. 584-591, 2003.

[13] L. Lee, L. Wang, P. Parvathala, and T. M. Mak, "On Silicon-Based Speed Path Identification," *Proc. VLSI Test Symp.*, pp. 35-41, 2005.

[14] S. Kajihara, M. Fukunaga, X. Wen, T. Maeda, S. Hamada, and Y. Sato, "Path Delay Test Compaction with Process Variation Tolerance," *Prof. Design Automation Conf.*, pp. 845-850, 2005.

[15] G. Hetherington, T. Fryars, N. Tamarapalli, M. Kassab, A. Hassan, and J. Rajski, "Logic BIST for Large Industrial Designs: Real Issues and Case Studies", *Proc. Intl. Test Conf.*, pp. 358-367, 1999.

[16] B. Nadeau-Dostie, D. Burek, and A. Hassan, "ScanBist: A Multifrequency Scan-Based BIST Method," *IEEE Design & Test of Computers*, Vol. 11, No. 1, pp. 7-17, 1994.

[17] X. Lin, R. Press, J. Rajski, P. Reuter, T. Rinderknecht, B. Swanson, and N. Tamarapalli, "High-Frequency, At-Speed Scan Testing," *IEEE Design & Test of Computers*, pp. 17-25, Vol. 20, No. 5, 2003.

[18] M. Beck, O. Barondeau, F. Poehl, and X. Lin, "Measures to Improve Delay Fault Testing on Low-Cost Testers – A Case Study," *Proc. VLSI Test Symp.*, pp. 223-228, 2005.

[19] L.-T. Wang, X. Wen, B.-C. Hsu, S. Wu, and J. Guo, "A Flexible Logic BIST Scheme for Multiple-Clock Circuits," *Proc. Intl. Conf. on Computer Design,* pp. 475-478*,* 2005.

[20] "Clock Domain Crossing," *Cadence Design Systems, Inc*.

[21] S. Savir and S. Patil, "On Broad-Side Delay Test," *Proc. VLSI Test Symp.*, pp. 284-290, 1994.

[22] L.-T. Wang, C.-W. Wu, and X. Wen, (Editors), *VLSI Test Principles and Architectures: Design for Testability*, Elsevier Science, July 2006.