Testing Delay Faults in Asynchronous Handshake Circuits

Feng Shi
Electrical Engineering Dept.
Yale University
New Haven, Connecticut
feng.shi@yale.edu

Yiorgos Makris
Electrical Engineering Dept.
Yale Univerisity
New Haven, Connecticut
yiorgos.makris@yale.edu

ABSTRACT

As a class of asynchronous circuits, handshake circuits are designed to tolerate variation of gate delays. However, certain timing constraints, such as the bundled data assumption, are exploited in the single-rail implementation of these circuits in order to simplify them. Therefore, any delay fault in the circuit may cause one of two problems, namely performance degradation or logic errors. To address the challenges incurred by the autonomous behavior of handshake circuits during at-speed test, we propose test methods for both types of delay faults based on a DFT strategy which greatly simplifies the complexity of test generation. The efficiency of the proposed methodology is demonstrated through experimental results on several handshake circuits.

Categories and Subject Descriptors

B.7.3 [**Integrated Circuits**]: Reliability and Testing

General Terms

Algorithms, Reliability, Verification

Keywords

Asynchronous Circuits, Handshake Circuits, Delay Faults, Test Generation

1. INTRODUCTION

The advantages of asynchronous circuits over their synchronous counterparts are demonstrated not only by academic research but also by commercial products, such as the newly-released asynchronous ARM996HS™processor [1]. Asynchronous circuits have the potential for higher performance, lower power consumption and design reusability. At the same time, they avoid a key emerging challenge of traditional synchronous design, namely high-frequency clock distribution. As chip complexity increases, clock skew effects are significantly amplified, making the problem intractable. Consequently, interest in asynchronous circuits has resurfaced and several commercial products have already been designed.

Despite the recent streak of progress, the development of CAD solutions for the design and test of asynchronous circuits is far from being sufficient. Among these issues, we focus on the problem of testing asynchronous circuits. Due to the autonomous behavior of asynchronous circuits, manufacturing defects may demonstrate themselves in a different way than in synchronous circuits and, thus, require a different test methodology. Considerable research [2, 4, 8, 10, 11, 13] has been done on test generation and design for testability (DFT) methods for stuck-at faults in various classes of asynchronous circuits. However, not much research [3, 5, 7, 12] has been conducted on testing delay faults.

Given the inherent robustness of many classes of asynchronous circuits to timing variations, a delay fault may only degrade their performance. However, most practical asynchronous circuits operate correctly by assuming that the implementation obeys certain timing constraints, hence they are no longer totally robust. Therefore, in these circuits, a delay fault may also cause the circuit to malfunction, i.e. generate logic errors. In addition, the autonomous behavior of asynchronous circuits implies that the existing test generation and test application methods for delay faults in synchronous circuits cannot be applied directly to asynchronous circuits. Instead, customized methods are necessary. In this paper, we study the two aforementioned types of delay faults in asynchronous circuits, we propose test methodologies for each of them, and we illustrate the effectiveness of our methods using one of the most established design styles of asynchronous circuits, namely handshake circuits.

The rest of this paper is organized as follows. First, in Section 2, we briefly introduce the design and implementation of handshake circuits, as well as the full-scan technique used for testing stuck-at faults in these circuits. Then, in Section 3, we classify the two types of delay faults in handshake circuits and outline the challenges in testing faults in each of the two classes. In Section 4, we propose test methods for both types of delay faults and in Section 5 we provide experimental results that demonstrate the efficiency of these methods.

2. HANDSHAKE CIRCUITS

As one of the most established design styles of asynchronous circuits, handshake circuits have a fully automated design flow from the high-level behavioral description to the physical layout, as well as test solutions for stuck-at faults.

2.1 Design and Implementation of HS Circuits

A handshake circuit is a network of handshake components, connected by point-to-point handshake channels, where all communication takes place via handshaking. It is the intermediate representation in the fully automatic compilation of a Haste program to a VLSI circuit. In the design flow of handshake circuits, first, a high-level design entry is written in a CSP-like programming language

ICCAD '06, November 5–9, 2006, San Jose, CA

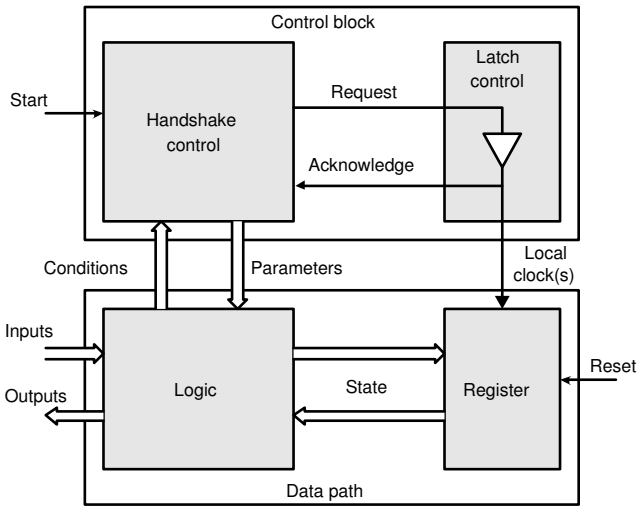

Figure 1: Gate-Level Implementation of a Handshake Circuit

called Haste. Then, it is compiled and translated in a transparent way into handshake circuits. After that, the gate-level implementation of the handshake circuit is generated by replacing all individual components with their gate-level implementations.

The gate-level single-rail implementation of a handshake circuit can be partitioned into a control block and a data path as illustrated in Figure 1. The control block operates on handshake signals, while the data path works on Boolean signals. The interface between the control block and the data path consist of the following three types of signals: conditions, parameters, and local clock signals. The control block uses the condition signals in combination with its internal state to determine the next action. Parameters are used by the control block to control the Boolean logic in the data path, for instance, by setting multiplexers in the correct state. The local clock signals are generated by the control block to enable a datapath register to capture a new data value.

2.2 Testing Stuck-At Faults Using Full-Scan

Besides the functional test method in [16] and the partial-scan method in [9, 10], a full-scan test method was proposed in [14, 15], exploiting commercial test tools for synchronous circuits to achieve high fault coverage while simplifying the test generation procedure for handshake circuits. Before scan insertion, combinational loops in the circuit are removed by inserting transparent scan flip-flops. Then, all state-holding cells are replaced by their scannable equivalents, which are connected together to form a scan chain. After that, the global control signals and logic are added into the circuit and connected to the scan elements, and the scan-testable netlist is generated. Meanwhile, the netlist is remodeled by replacing original scan cells with their remodeled equivalents to generate two separate netlists for the control block and the data path respectively, which are readable by commercial test tools for performing test generation. Then, a test-protocol expansion procedure translates the initial protocols generated by the ATPG tools into the top-level protocols.

The above full-scan method was refined in [13] by introducing multiplexer-based scannable C-elements, as illustrated in Figure 2. The proposed method significantly reduces the overhead in performance and area of the scan testable circuit. Moreover, the latch in Figure 2 is often shared with an existing scan latch/flip-flop in the datapath, which further reduces the test cost. However, any two

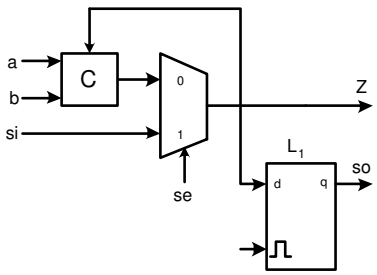

Figure 2: A Mux-based Scan C-Element

multiplexer-based scan elements connected in series cannot capture the circuit response at the same time, hence they cannot be enabled at the same time and are, thus, connected to different scan enable signals. Therefore, testing the control block is performed for several partitions respectively, and a test control block is embedded in the circuit to control the test procedure.

3. CHALLENGES OF DELAY TESTING

There are two types of delay faults in asynchronous circuits depending on the two different consequences of the fault. The first type of delay faults only slows down the performance of the circuit without causing any logic error. This is common when the component affected by the fault is *delay-insensitive*. If we only consider delay faults modeled at gate output which *increase* the delay, delay faults in the control part of a handshake circuit fall into this category. Despite not causing a logic error, we may still want to test for this type of delay faults, since the circuit should not be expected to be unreasonably slow. The second type of delay faults causes an asynchronous circuit to malfunction and produce logic errors, when the delay faults result in violation of inherent timing constraints that need to be satisfied for the circuit to operate correctly. Such timing constraints exist in almost all practical asynchronous circuits, since the class of delay-insensitive circuits built with basic gates is quite limited [6]. For instance, the data path in a handshake circuit usually needs to follow bundled data, setup, and hold time constraints. Obviously, the second type of delay faults is necessary to test for, and targeted delay faults in the data path of a handshake circuit fall into this category.

Similar to synchronous circuits, at-speed delay-test methods are necessary to capture the fault effect in asynchronous circuits. If the delay fault is in the control part of the handshake circuit and it only degrades the performance but does not cause any logic error, the fault effect can be captured by applying the test clock according to a pre-specified acceptable performance, just like in synchronous circuits. However, if the delay fault is in the data path and violates some timing constraint, such as the bundled data constraint, hence causes the circuit to fail during normal operation, the fault effect is more difficult to capture, since the speed of an asynchronous handshake circuit is inherent to the circuit and it is usually difficult to know the exact delay of a certain logic path. Hence, it is almost impossible to clock the circuit during test in such a way that the delay mismatch can be identified. A possible solution to the above problem is to switch the circuit from test mode to normal operation mode immediately after the test patterns are applied and the fault is activated, so that the fault effect may be captured in time.

However, there are several difficulties in performing at-speed test for asynchronous handshake circuits by switching between test mode and normal operation mode. First, the test patterns should be generated carefully to make sure that they will not cause any

hazards or races when the circuit is switched to normal operation mode. Second, the switching mechanism between test mode and operation mode should be carefully designed so that it neither impacts the timing of normal operation, nor causes any hazards or races. To achieve this, careful physical design may be necessary to make sure that test control signals follow certain timing constraints, and a robust switching method is desirable to simplify or even eliminate these timing constraints.

In addition, the test patterns for at-speed delay test need to make sure that the deterministic fault effect is captured by scan latches or flip-flops, or observed on the primary outputs when the circuit settles in a stable state. As we know, unlike a synchronous circuit, an asynchronous circuit has no clock signal to control the feedback paths, hence once it is switched to normal asynchronous operation mode, it may perform a sequence of operations and it is impossible, in general, to single-step the circuit. The test patterns must guarantee not only that the deterministic fault effect is captured correctly, but also that the fault effect is not overwritten and does not disappear during the subsequent operations. This complicates the fault simulation and automatic test pattern generation processes significantly. Moreover, the violation of a timing constraint may cause logic errors on multiple bits of the data path, hence the fault effect may not be unique and the final response of the faulty circuit is difficult to derive if the circuit progresses through a sequence of operations.

4. PROPOSED METHODS

Since there are two cases of delay faults in handshake circuits, as described in the previous section, we propose two methods to test for each of them, respectively. First we present the method to test for delay faults that only degrade the performance. Then, we focus on testing for delay faults that cause the circuit to malfunction. The proposed methods are built upon the multiplexer-based full scan test method in [13] with minimal additional hardware.

4.1 Test for Performance Degradation

As discussed in Section 3, the targeted delay faults in the control block of a handshake circuit do not cause any logic error; therefore, they are tested for degrading the performance of the circuit.

The proposed test method for delay faults in the control block makes use of the multiplexer-based full-scan method [13] for stuck-at faults in handshake circuits. After scan insertion, asynchronous handshake circuits can be tested similarly to synchronous circuits. Since scan cells connected in series cannot capture the circuit response at the same time, the control block is partitioned into several parts for test generation, and faults in different parts are tested accordingly. For each part, the input netlist for ATPG is a synchronous circuit generated through scan insertion, remodelling of asynchronous cells, and additional modifications. Hence, test patterns for not only stuck-at faults but also delay faults, such as transition faults or path delay faults, can be generated using commercial EDA tools. The proposed test method only supports the scan shifting mode, since the functional justification mode cannot be directly applied to the control block. In the scan shifting mode, the state transition from the first test vector to the second vector is initiated in the last scan load cycle, and the target fault is sensitized. Then, the circuit response is captured at a reference clock cycle time which is derived according to a pre-specified acceptable performance of the circuit, and then scanned out and compared to the correct response. This procedure is similar to that of delay test in synchronous circuits, except that a reference clock is used rather than an at-speed clock.

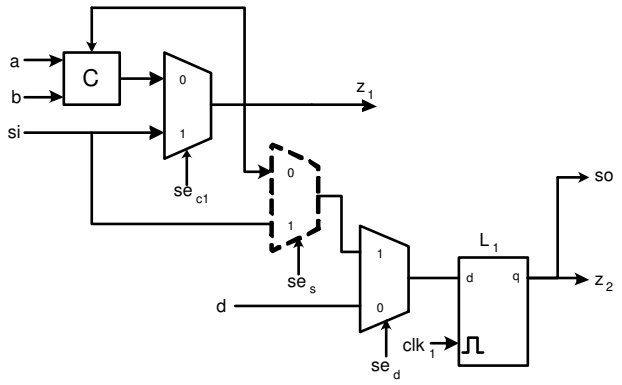

Figure 3: A Revised Mux Based Scan C-Element

4.2 Test for Timing Constraints

Delay faults in the data path of handshake circuits may increase the computation time, which not only may reduce the performance of the data path, but, more importantly, may also violate the bundled data constraint and cause the circuit to malfunction. Therefore, these delay faults are more critical to test for.

However, these delay faults cannot be handled in the same way as in synchronous circuits. First, test patterns cannot be applied by simply shifting through the scan chain, since the scan latch/flip-flop may be shared with the control block. A two-step test application procedure is usually necessary to set the state of the control block and then sensitize and detect the fault in the data path. Second, the circuit response is difficult to be captured by applying an at-speed clock. As a result, the proposed method captures the circuit response by switching the circuit to asynchronous operation mode rather than using the test clock.

Additional hardware support is necessary for the two-step test application procedure. For each multiplexer-based scan cell in the control block, a multiplexer (shown in dashed lines) is inserted, as illustrated in Figure 3, where the scan latch is shared by the control block and the data path block. As a result, there are two scan paths controlled by the additional control signal se_s. The original scan path is selected when se_s is set to low, while the multiplexer in the control block is bypassed when it is set to high. Although the inserted multiplexers introduce minor hardware overhead, they do not cause any performance overhead since they are only on the scan path.

The proposed timing of test application is the following. In the first step, when control signal se_s is set to 0 while all other *se* signals are set to 1, a test vector is shifted into the scan chain to set the state of the control block. Then, all *se* signals for all partitions of the control block are set to 0 to isolate the control block from the scan chain. After that, in the second step, se_s is set to 1, and the test vectors generated for the data path are shifted in through the alternative scan path. At the end of this step, the last scan shift operation initiates the launch phase and sensitizes the target delay fault. When the last shift operation is completed, the circuit is set to asynchronous operation mode during which the circuit response is captured. This is achieved when the latches or flip-flops behind the fault are enabled, or when the state of the control circuit changes according to the condition signals from the data path. After it stabilizes, the circuit is set back to test mode. If the fault effect is not already captured by the scan latches/flip-flops or propagated to the primary outputs (i.e. reflected in the state of the control block), an additional test clock is applied to capture the fault effect by the cor-

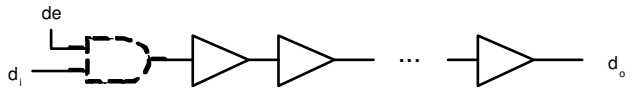

Figure 4: A Controlled Delay Chain

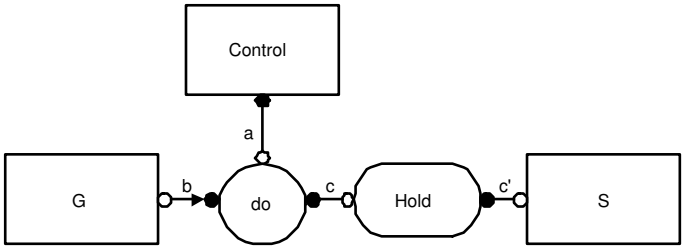

Figure 5: DFT of an Iteration

responding latches/flip-flops. Then, the circuit is set to scan shift mode and the circuit response is scanned out and compared to the correct response to check whether the fault exists.

When the state of the control block is set and isolated from the scan chain during the first step of the test application procedure, a mechanism is necessary to preserve this state from any autonomous changes, which will happen if the state is not stable. Since the timing constraints under test are, in fact, delay matchings between the data paths and the delay chains, the state that the control block is set to during the test is the one necessary right before the delay chain is exercised. Thus it can be held by simply making the delay chain controllable. As illustrated in Figure 4, an AND gate (shown in dashed lines) is inserted at the beginning of each delay chain, and signal *de* is used to control the delay chain. When *de* is low, the delay chain is disabled and no rising event can propagate through it, while when *de* is set to high, the delay chain works as normal. During the first step of test application, *de* is set to 0 to preserve the state of the control block. Then, when the circuit is switched back to normal asynchronous operation mode, it is set to 1 to enable the control block to operate normally. Since the inserted control gate can be incorporated into the delay chain, there is nearly no hardware and no performance overhead due to its introduction.

The generated test patterns for the above test method need to be able not only to sensitize the fault, but also to capture the deterministic fault effect. The part of the test pattern applied to the data path to exercise the fault can be generated using ATPG tools for synchronous circuits, since the proposed method exercises the fault in a way similar to testing a synchronous circuit. The part of the test pattern applied to the control block in the first step needs to make sure that fault effect is correctly captured when the circuit is switched back to normal operation mode. To reduce the test generation complication due to the autonomous behavior when the circuit is switched to asynchronous operation mode, which was discussed in the previous section, we propose a DFT strategy which constrains the autonomous behavior of the circuit during test. Hold elements [10, 15] are inserted between the handshake components when necessary, as illustrated in Figure 5 for example, to make sure that during test i) the latches/flip-flops following the fault are clocked only once to capture and maintain the immediate circuit response, or ii) the wrong state bits in the control circuit are preserved until the circuit stabilizes. Since each hold component is implemented as a single AND gate, the hardware overhead is negligible.

Based on the proposed DFT strategy, the test generation for violation of timing constraints is greatly simplified. The procedure of test generation takes places in two steps. First, for any target delay fault in the data path, any test generation tool for synchronous circuits may be used to generate the part of test patterns for the data path block based on the remodeled input netlist. The location where the fault effect propagates to is marked. Second, functional test patterns are found based on high-level simulation, in order to exercise the data path under test. If the fault effect location marked in the previous step is not a flip-flop/latch in the data path, which means that the fault is in the data path for guard evaluation, the test patterns are required to exercise the corresponding guarded statement. Otherwise, the test patterns are required to exercise the expression that corresponds to the data path under test, in which case the marked flip-flops/latches need to be clocked at least once. Only a test pattern that holds the same value on the parameter signals when exercising the data path under test, as in the previously generated other part of the pattern, is eligible to be selected. The snapshot of the transient state of the control block is kept when the data path is about to be exercised, and the signal value on the location of any scan element in the control block in the snapshot is collected to form the other part of the test pattern. Through this process, we obtain the two parts of the test patterns that are applied during each of the two steps of the test application procedure, respectively.

The test patterns generated through the above two steps are valid based on the following observations. First, although the test patterns determining the conditional signals are generated in the first step without considering the behavior of the control block, they will not cause undesired behavior of the control block. If the fault under test is in the data path for guard evaluation, its effect propagates through the control signals and leads the control block into an erroneous state, which is then captured to observe the fault. Otherwise, if the fault is in the data path for data computation, the control block must be set to the appropriate state before a computation expression in the corresponding Haste program to excite and capture the fault effect. Note that this state must be between guard evaluations, such as in the execution of the statement *S* in the *do* statement illustrated in Figure 5. At this time, the evaluation of any preceding guard has been completed, its result has been captured by the control block, and the control signals are not valid any more. Hence, any change on the control signal does not influence the behavior of the control block. Second, the test patterns that are applied to the control block to determine the values of the parameter signals will not conflict with the test patterns for the data path block, since only consistent pairs are selected during the test generation.

5. EXPERIMENTAL RESULTS

The proposed delay test methods are implemented based on the design tool set for handshake circuits from Handshake Solutions[1] [1]. First, the netlist generated after scan insertion by *htscan* and its synchronous model for ATPG tools are preprocessed such that they are consistent with the revised scan method. Additional test hardware, such as controlled delay chains and hold components are also inserted for testing for timing constraint violations. Then, the timing constraints in the circuit are identified and a list of them is generated using *htpost*. After that, the delay fault model (transition faults or path delay faults) is chosen, and the fault-list is generated. For each fault in this list, the type of fault (i.e. causing performance degradation or causing timing constraint violation) is determined

[1]The authors would like to thank Handshake Solutions for providing their design tool-set as part of a collaboration under the DARPA CLASS program.

Circuit Name	No. of Inputs	No. of Outputs	Area	Hardware Overhead	Faults in control block		Faults in datapath block		Total fault Coverage
					No. of faults (Untestable)	Faults Detected	No. of faults (Untestable)	Faults Detected	
conv1to8	8	11	408	24	524(0)	464	152(0)	136	88.76%
conv3to8	10	11	506	33	878(0)	782	194(0)	172	88.99%
fifo8	15	11	1119	66	1100(0)	909	1296(0)	1101	83.89%
gcd	23	11	390	25	286(0)	183	930(2)	859	85.83%
des-round	127	123	3271	24	258(0)	188	10472(401)	8631	85.38%
des	127	124	51456	252	4683(0)	4521	169552(7804)	139930	86.79%

Table 1: Results of ATPG for Transition Delay Faults

according to its location. Then, for each fault that degrades the performance, the test patterns are generated and applied through the proposed method in Section 4.1. For each timing constraint in the list, the functional test vectors that exercise the timing constraint under test are found through high-level simulation-based search, and the part of test patterns that is applied on the control block is obtained from the snapshot of the netlist simulation of the circuit. Then, for each fault that violates a listed timing constraint, the part of the test patterns that is applied on the data path is generated through ATPG tools. After that, the two parts of the test patterns are applied to the circuit during the two steps of the test application process, respectively, as discussed in Section 4.2. Lastly, for both types of faults, the generated test patterns are validated through simulation.

We experimented with the proposed methods on a set of example handshake circuits synthesized using the design tool flow from Handshake Solutions. Transition delay faults in both the control and data path blocks in each circuit are tested by the proposed method, which employs *TetraMax*® to perform part of the test generation task. The results are reported in Table 1. The name of each circuit is listed in the first column in Table 1, and the number of inputs, the number of outputs, and the circuit area (in 2-input NAND-gate equivalent) for each circuit are listed in the second, third, and fourth columns, respectively. Note that circuit *des* is a fully pipelined DES encoder/decoder, while *des-round* only performs one round of DES encoding/decoding operation. The area of DFT hardware overhead of the proposed method for each circuit is shown in the fifth column, indicating that the hardware overhead is negligible, especially for large circuits. The total number of transition faults and the number of detected faults in the control block of each circuit are listed in the sixth and seventh columns respectively, with the number in the parentheses indicating the number of untestable faults reported by TetraMax®. The same numbers are listed for the data path block of each circuit in the eighth and ninth columns. Finally, the total fault coverage for each circuit is given in the tenth column. For the circuits that we experimented with, a fault coverage of over 85% is achieved by the proposed methods.

6. CONCLUSION

Two types of delay faults that either degrade the circuit performance or cause logic errors are distinguished in asynchronous handshake circuits. In order to address the delay test challenges arising from the autonomous behavior of asynchronous circuits, test methods and DFT techniques that simplify the complexity of test generation have been developed for both of these types of delay faults. The efficiency of the proposed test methods has been demonstrated through experimental results on several handshake circuit examples.

7. REFERENCES

[1] Handshake solutions. http://www.handshakesolutions.com.

[2] S. Banerjee, S. T. Chakradhar, and R. K. Roy. Synchronous test generation model for asynchronous circuits. In *Proc. of the 9th International Conference on VLSI Design*, pages 178–85, 1996.

[3] G. Gill, A. Agiwal, M. Singh, F. Shi, and Y. Makris. Low overhead testing of delay faults in high-speed asynchronous pipelines. In *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pages 46–56, 2006.

[4] P. J. Hazewindus. Testing delay insensitive circuits. *Ph.D. Thesis, Department of Computer Science, California Institute of Technology*, 1992.

[5] M. Kishinevsky, A. Kondraytev, L. Lavagno, A. Saldanha, and A. Taubin. Partial-scan delay fault testing of asynchronous circuits. *IEEE Transactions on Computers*, 17:1184–1198, 1998.

[6] A. J. Martin. The limitations to delay-insensitivity in asynchronous circuits. In W. J. Dally, editor, *Advanced Research in VLSI*, pages 263–278. MIT Press, 1990.

[7] S. Nowick, N. Jha, and F.-C. Cheng. Synthesis of asynchronous circuits for stuck-at and robust path delay fault testability. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 16(12):1514–1521, December 1997.

[8] O. Roig, J. Cortadella, M. A. Peiia, and E. Pastor. Automatic generation of synchronous test patterns for asynchronous circuits. In *Proc. of the 34th Design Automation Conference*, pages 620–625, 1997.

[9] M. Roncken, E. Aarts, and W. Verhaegh. Optimal scan for pipelined testing: An asynchronous foundation. In *Proc. International Test Conference*, pages 215–224, 1996.

[10] M. Roncken and E. Bruls. Test quality of asynchronous circuits: A defect-oriented evaluation. In *Proc. International Test Conference*, pages 205–214, 1996.

[11] F. Shi and Y. Makris. SPIN-SIM: Logic and fault simulation for speed-independent circuits. In *Proc. of International Test Conference*, pages 597–606, 2004.

[12] F. Shi, Y. Makris, S. Nowick, and M. Singh. Test generation for ultra-high-speed asynchronous pipelines. In *Proc. of International Test Conference*, pages 39.1–39.10, Nov 2005.

[13] F. te Beest and A. Peeters. A multiplexer based test method for self-timed circuits. In *Proc. of IEEE International Symposium on Asynchronous Circuits and Systems*, pages 166–175, 2005.

[14] F. te Beest, A. Peeters, M. Verra, K. van Berkel, and H. Kerkhoff. Automatic scan insertion and test generation for asynchronous circuits. In *Proc. of International Test Conference*, pages 804–813, 2002.

[15] F. J. te Beest. *Full Scan Testing Of Handshake Circuits*. PhD thesis, Twente University, 2003.

[16] R. van de Wiel. High-level test evaluation of asynchronous circuits. In *Asynchronous Design Methodologies*, pages 63–71, 1995.