A Partial Scan Methodology for Testing Self-Timed Circuits

Ajay Khoche Erik Brunvand
Department of Computer Science
University of Utah
Salt Lake City, UT, 84112

Abstract

This paper presents a partial scan method for testing control sections of macromodule based self-timed circuits for stuck-at faults. In comparison with other proposed test methods for self-timed circuits, this technique offers better fault coverage than methods using self-checking techniques, and requires fewer storage elements to be made scannable than full scan approaches with similar fault coverage. A new method is proposed to test the sequential network in this partial scan environment. Experimental data is presented to show that high fault coverage is possible using this method with only a subset of storage elements being made scannable.

1 Introduction

Asynchronous and self-timed circuits have recently been receiving renewed interest by circuit designers as an alternative to globally synchronous system organization. As the size and speed of systems grow, so do the problems related to the global clock signal. Asynchronous and self-timed circuits that avoid timing problems by enforcing simple communication protocols between parts of the circuit can help avoid these problems. These types of systems can also allow simpler system composition, show increased robustness in the face of process and environmental variation, can exhibit much lower power consumption, and can even show increased performance when compared to globally synchronous systems in some cases.

Testing asynchronous circuits, however, is a relatively new area. Despite the growing number of recent efforts in the specification and design of asynchronous circuits, testing these circuits has not been explored to any great degree. Traditionally, testing asynchronous circuits has been considered a difficult problem, especially when compared to the synchronous circuits, where significant advances have been made. Unfortunately, methods used to test synchronous circuits are not directly applicable to asynchronous circuits. This is due, in large part, to the absence of the global clock signal in the asynchronous circuits. New methods are required to adapt the rich knowledge about testing synchronous circuits to test asynchronous circuits. This is precisely the subject of this work: to adapt scan path technology to a class of asynchronous circuits.

Asynchronous style control circuits can be classified broadly into two categories: centralized and distributed. In the centralized style the control is designed like conventional state machines, where a single state machine controls the sequencing in the circuits. These machines are typically designed with restrictions on input and outputs and need proper adjustment of delays to handle an asynchronous environment. Many approaches have been proposed to design control circuits in this style [9, 20, 30, 32].

In the distributed style of design the control unit consists of an interconnection of many smaller state machines (macromodules). These macromodules are typically designed to follow certain protocols at their interfaces that obey delay-insensitive or speed-independent properties to make their composition simpler [3, 18, 21, 28]. Self-timed macromodular control circuits have been used in a wide variety of academic research efforts [4, 19, 23], as well as in industrial research settings [2, 27], and it is this style of distributed self-timed control that we focus on in this work. Using these modules, distributed self-timed control can be built easily by connecting the modules directly into a control network. These modules also allow simple syntax-directed translation from language descriptions into control networks [3, 18, 21]. In particular, the set of macromodules used by Brunvand [3, 5] and Sutherland [28] are the modules used to build the circuits which are the target of this paper. The particular set of macromodules used is, however, not critical as the techniques we present could be applied to any similar set of control circuits.

In this paper a partial scan method is proposed to test the control portion of macromodule based self-timed circuits for stuck-at faults. This method provides better fault coverage than methods using the self-checking property [12] of self-timed circuits which assumes that the circuit halts in response to faults. It also requires fewer storage elements to be made scannable than full scan methods while offering acceptable fault coverage.

0-8186-7000-2/95 $04.00

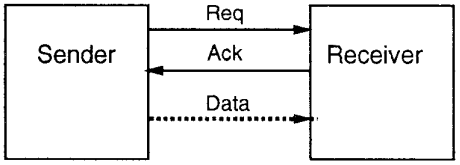

Figure 1: Bundled Data Path Connection

The paper is organized as follows. In the next section, self-timed circuits and the overall design methodology will be described briefly. Section 3 discusses the testability of macromodule based self-timed circuits. Section 4 reviews related work. Section 5 presents the proposed partial-scan method including the the overall architecture, modifications to the various modules, and the procedures used to test sequential logic consisting of XOR and C-elements. Section 6 presents experimental results obtained on four examples. Finally, Section 7 offers some conclusions.

2 Self-Timed Macromodule Circuits

Self-timed circuits are a subset of a broad class of asynchronous circuits. These circuits generate completion signals to indicate that they are finished with their processing [26]. A signalling protocol used with the completion signalling allows self-timed systems to be composed of circuits which communicate using self-timed protocols. Self-timed protocols are often defined in terms of a pair of signals, one to request or initiate an action, and another to acknowledge that the requested action has been completed. One module, the sender, sends a request event (*Req*) to another module, the receiver. Once the receiver has completed the requested action, it sends an acknowledge event (*Ack*) back to the sender to complete the transaction.

The circuits used in our library use two-phase transition signaling for control and a bundled protocol for data paths. Two-phase transition signaling [26] uses transitions on signal wires to communicate the *Req* and *Ack* events described previously. Only the transitions are meaningful; a transition from low to high is the same as a transition from high to low and the particular state, high or low, of each wire is not important.

A bundled data path uses a single set of control wires to indicate the validity of a *bundle* of data wires [28]. This requires that the data bundle and the control wires be constructed such that the value on the data bundle is stable at the receiver before a signal appears on the control wire and remains valid until *Ack* is received. modules connected with a bundled data path are shown in Figure 1.

Our design method builds control circuits using a variety of modules which communicate using two-phase transition signals. These modules, described in more detail elsewhere [5, 28], are shown symbolically in Figure 2. Other modules in the library, such as transition-controlled latches, and completion-sensing adders, are used to build self-timed data paths.

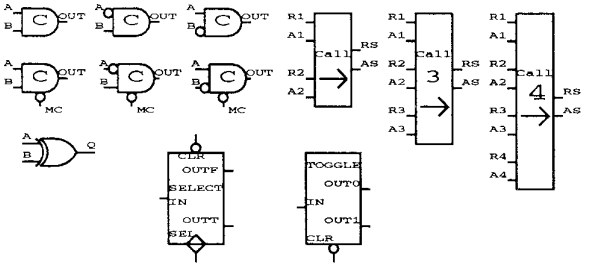

Figure 2: Control Modules for Self-Timed Designs

The control modules include circuits that act as OR gates for transitions (implemented using an XOR) and AND gates for transitions (implemented using a C-element). A transition-OR gate will produce a transition at its output whenever there is a transition at either input. An AND requires transitions at both inputs before producing a transition at the output. Also included are modules that steer transitions depending on a Boolean input signal (a Select module) or alternate transitions on the output for each transition on the input (a Toggle module). A Call module allows multiple sender circuits to have access to a common receiver circuit by implementing a hardware subroutine call. The Call module requires that the multiple requests be mutually exclusive so that it need not perform arbitration.

One way this module library is used is with an OCCAM based automatic circuit compilation system [3]. The software constructs of OCCAM have been implemented using these library components and allow programs written in OCCAM to be translated automatically into self-timed circuits. An example of translation of a WHILE construct is shown in Figure 3. The OCCAM compiler has been used to build a number of systems ranging from a memory controller for standard DRAMs used in a self-timed environment [3], to a simple wormhole router designed for a mesh-connected multiprocessor array [3]. This library has also been used to build large circuits by hand, including a self-timed microprocessor [4], using commercial schematic capture software from Viewlogic .

3 Testability of Macromodule based Self-Timed Circuits

Some of the problems associated with testing asynchronous circuits in general and self-timed circuits in particular include:

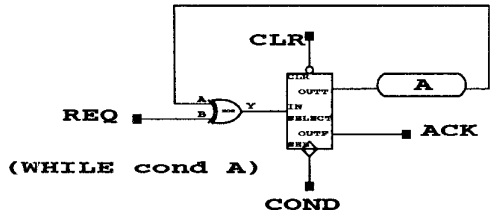

Figure 3: Translation of WHILE construct into Circuit

- Asynchronous and self-timed circuits are more sensitive to races and hazards than synchronous circuits. This puts an additional requirement on the test methodology that test application must also be hazard and race free otherwise the test may be invalidated.

- Self-Timed circuits operate in an autonomous way, in the sense that once the control is passed from environment to the circuit, the operation is totally determined by the circuit. This is different than for synchronous circuits where an external clock dictates the operation of the circuit. This creates problems if one attempts to use an iterative array model for testing because the number of frames is not controllable.

- Using our library module approach, control is distributed throughout the system and is not centralized in a single controller with a convenient state register for the scan path. Each self-timed module is in fact a tiny state machine in itself. This increases the complexity of testing if functional testing of the entire circuit is desired.

- Functional testability of a self-timed module depends on the way it is used in a circuit. In other words, it depends on the environment which interacts with the module. Certain circuit configurations lead to only a subset of input patterns ever being applied statically to the module. For example, C-elements are often used as a rendezvous for two forked processes. In this case, the only static values seen at the inputs to the C-element will be 11 and 00. The 10 and 01 cases will be seen only during the time that one process has finished and the other is still executing. In addition, other faults may be masked by functional test methods where the observability mechanism is only observing the primary output. A detailed discussion of these problems may be found in [15]. The percentage of detectable faults for each self-timed control module using only functional test and the self-checking property is are shown in Table 1.

Module	Fault Coverage
C element	60.0%
Call2	60.2%
Select	71.5%
Toggle	90.0%

Table 1: Fault Coverage of Modules using Self-Checking Functional Test Only

4 Related Work

Testing asynchronous circuits is a relatively new area. Very few attempts have been made to date. For testing macromodule based self-timed circuits only two approaches have been reported in the literature that we know of. In [16] a functional approach is used for testing self-timed macromodule circuits using the self-checking property mentioned earlier. In this approach the *SEL* lines of the Select modules are made controllable thereby influencing the flow of control in the circuit. After selecting a particular path the input to that path is changed from low to high and then back. The observation mechanism consists of outputs also changing after waiting for sufficient amount of time. This approach targets faults only on module's input and output. this approach has the following drawbacks when the faults inside the modules are also considered.

- In their approach modules are considered atomic: faults inside the modules are not targeted. Since the faults on the input and output of the module form a small percentage of the total faults in the control circuits the fault coverage offered by this method is low when faults inside the modules are also considered, as shown in Table 1.

- The only observation mechanism is observing the output. This is not sufficient because a great deal of fault masking may occur inside the circuit as described in the previous section.

- Functional testing may result in high complexity when faults inside the modules are also considered. This is because certain faults inside the modules require state justification to activate them, which is not necessary for faults on the module input/output considered in [16]. State justification may be computationally complex in the type of circuits considered in this paper, which contain a lot of state distributed in the circuit.

- Loop structures can not be tested without adding extra observability to the circuit. Select modules are often used to build looping structures in the circuit as shown in Figure 3. When the loop body is executed in test mode the control stays in that loop and there is no way out.

Other researchers [11, 13, 24] have also proposed methods based on the self-checking property for circuits similar to ours. However all these approaches will suffer the same disadvantages described in previous section. Hazewindus [11], for example, proposed adding control/observation points for each untestable fault. Clearly this is impractical in this type of circuit as the number of such faults is large.

In [14] the authors propose a full scan approach where the scan path is instantiated on the req/ack lines of each of the control modules. Faults inside the modules were also considered. This method provides excellent fault coverage, but has high overhead. The overhead of our full scan approach is what initiated the partial scan work reported in this paper.

Other efforts reported in literature do not directly deal with self-timed macromodular circuits, however some of them are still relevant. In [31] a full scan approach was proposed for circuits generated using circuits generated from Signal Transition Graph (STG) descriptions. These circuits are essentially Huffman type asynchronous state machines. In their approach each storage element (C-element) is replaced by an SRL latch. This approach is not practical for macromodular circuits as the ratio of logic gates to latches is very low and thus it will result in high scan latch overheads. Even with a full scan path it may not be possible to test for all faults due to reconvergent fanout, which is very common in self-timed macromodular circuits. This will be explained in the next section. Full scan also implies longer scan chains, resulting in longer test application times, however this should be compared against the number of extra vectors in the partial scan where sequential testing may be required.

5 Partial Scan Solution

As described in the previous section, if the gate to latch ratio of the circuits is low, the overhead of full scan will be high. Also due to the structure of circuits it may not be possible to test the circuits for 100% fault coverage even with full scan. A partial scan solution which requires less overhead but still offers acceptable fault coverage is described in this section.

5.1 Selection of Scan Latches

Present approaches for selection of scan latches are based upon testability analysis, test pattern generation, or structural analysis [6, 7, 10, 17]. In our method a combined approach involving testability analysis and structural analysis has been followed for scan latch selection. This process involves three stages:

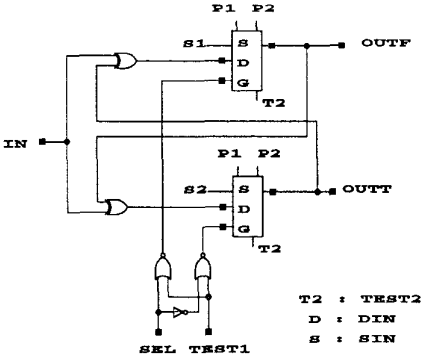

Figure 4: Modified Select Module

1. Analysis of Select and Toggle modules reveals that faults inside the Select and Toggle modules are difficult to test using functional test methods [15] so all the Select and Toggle elements are added to the scan path. This partitions the circuit into networks of XORs and C-elements (Call elements can be considered to be a network of XORs and C-elements). The C-elements are special sequential elements with only two states. These elements have been modified such that they can be tested in combinational way. This will be described in Section 5.3.

2. In the second stage, the Call elements are analyzed to see if it is possible to justify values of the *AS* line independent from the values of *R1* and *R2*. This is required to test the faults which are described in [15], that are untestable using a functional test on Call elements. If this is not possible, extra transparent scannable latches are added to the circuit. These latches are added in such a way that delay incurred can be hidden.

3. The last stage involves analysis of the circuit for loops that do not contain a scanable latch. In such cases a C-element in the loop is made scannable. We have developed software to detect these conditions and to suggest which C-element to be made scannable such that many loops can be broken simultaneously.

5.2 Modifications to Circuit Elements

Each of the circuit modules in the library requires some modification to fit into the partial scan environment. Examples of the Select and C-element module are given here. Other minor modifications to the basic self-timed control modules are described in [15].

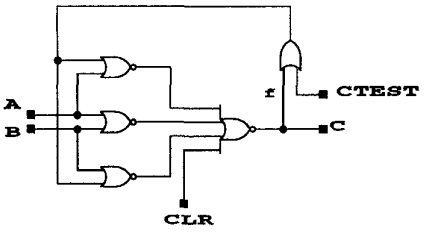

Figure 5: Modified C-element

5.2.1 Select Module

The modified Select element is shown in Figure 4. The first modification involves the latches inside the Select module. Originally these latches were single stage gated latches. These latches have been modified to become master-slave latches. This is done to reduce correlation between successive stages of the scan path which could restrict certain vectors from being scanned in. The *CLR* has also been removed from the latches as these latches can be reset using the scan path by putting the scan path in transparent mode from scan path input to output.

The second modification is made to the *SEL* line of the module. The *SEL* line is disabled during scan mode so that both latches receive input from the scan path rather than from their normal input. After the scanning is over and the circuit has stabilized the *SEL* line is enabled. This causes one of the latches to be enabled depending on value of *SEL* line which allows the ouput of the network under test to be captured into the the master latch.

5.2.2 C-element

The C-elements are in general not included in the scan path as described above. Instead they have been modified to be testable in a combinational way. This modification is based on the observation that a C-element acts as an AND gate if its internal state is 0 and as an OR gate if its internal state is 1. The state of a C-element is the state of feedback wire, so if we can control the state of feedback wire we can make it act like an AND gate or an OR gate. One way to do this is to add a MUX in the feedback wire controlled by a mode signal such that in test mode the value of feedback wire is determined by the other input of the MUX. This approach was proposed in [29] however this method leaves the normal feedback input to the MUX untested and also adds one extra control signal. In our method an OR gate is introduced in the feedback line so only a 1 value can be controlled on the feedback line. For controlling a 0 value we use the system clear signal, which is already present in the original C-element design. This allows a fault on the feedback line to also be tested and requires one fewer control signal. The model for a modified C-element is shown in Figure 5.

5.3 Test Procedure

Once the scan path is introduced, the test procedure uses this scan path to test the remaining parts of the control path which consists of XOR and C-elements. This section will first describe the scan path operation and then describe the procedures used to test networks of XOR and C elements.

5.4 Scan Path Operation

The circuit is put into scan mode by asserting *Test1* and *Test2* control signals (shown in Figure 4). *P1* and *P2* signals provide two phase non-overlapping clocks to the scan register to provide a race free operation.

Once the circuit has settled based on the scan path inputs, its output is captured by deasserting the *Test1* signal which enables the *SEL* signal to the latches. Enabling the *SEL* causes one of the latches to be enabled depending on the value of *SEL* line which is input to the control part. The circuit output is now captured through the normal input of the latch. The circuit is then returned to scan mode and the output is scanned out. The faults on the *SEL* line are tested during the capture process by appropriately controlling its value. It is important to note that although the test application is synchronous, the circuit operation remains asynchronous.

5.4.1 Testing XOR/C Networks

The network of XOR and C-elements is tested in three steps:

1. In the first step the C-elements are put into OR-mode by asserting the test signal to the C-elements. Thus in this step the network of XOR and C elements reduces to a network of XOR and OR gates. The tests for this network can be generated using any conventional test pattern generation software.

2. Even though the first testing step covers most of the faults in the interconnection of XOR and C-elements, about 40% of the faults inside the C-elements remain untested. These faults require the C-element to be put into AND-mode. The C-elements are put into AND mode by asserting the global clear signal, *CLR*. This signal is kept asserted during the scan-in operation (Note that the CLR signal does not go to the elements on the scan path). This is required because otherwise during the scan-in operation different inputs will appear at the input of the C-elements and the state of the C-elements at the end of the scan-in operation will be indeterminate. Once the scanning is over, *CLR* is deasserted and the signal values are allowed to flow

Design	Fault Coverage		No. of Latches Scanned	
	Self-checking	Partial Scan	ALScan	Partial Scan
Router	65.7%	98.2%	34	17
IFstage	66.1%	97.4%	26	13
GCD	74.6%	95.5%	11	7
Division	67.5%	100.0%	7	6

Table 2: Comparison of Fault Coverage and No. of scannable Latches

through the network of XOR and C-elements. This operation does, however, raise the issue of races and hazards since the signal changes at the input to the network propogate through the network in parallel. This puts additional requirements on the test generation for this test step that the tests should be hazard free. There are two alternatives. One is to generate the tests and then validate that the tests are hazard free. Second is to generate only hazard free tests, which will require a new test pattern generator. Presently the first approach is used to generate the tests.

3. After the first two test steps all the faults in the C-elements are tested except for the s-a-0 fault on the feedback line f to the OR gate in Figure 5. In order to test for these faults the C-elements are put into OR mode and a 01 or 10 input is justified at the inputs of the C-element which is under test. The test input feeding the OR gate is then deasserted. Now in a fault free case the output of the C-element will remain 1 while in faulty case the output will change to 0. This fault behavior can be propagated to the output of the network. The conditions for propagation are the same as in the first step where the C-elements are also put into OR mode.

6 Experimental Results

The method described in this paper was applied to four example self-timed macromodular control circuits. The results are listed in Table 2. The router circuit is a torus-connected wormhole routing chip for message routing in a multiprocessor [3, 8]. The circuit called IFstage is the instruction fetch unit of the NSR, a self-timed pipelined RISC processor [4]. GCD is an implementation of Euclid's algorithm and Division is a serial divider circuit. The self-checking column in the Fault Coverage section refers the fault coverage obtained by a functional test method such as that described in [16]. This relys on the self-checking property that the circuit will halt in response to a class of faults. This coverage assumes that some additional observability mechanisms have been used to detect faults inside the loop body as mentioned in Section 4, otherwise the coverage may be even lower. The Partial Scan column refers to the fault coverage obtained by the method described in this paper. In the No. of Latches section, the ALScan column reports the number of latches that would have been made scannable if the full scan method proposed in [31] were adopted. The Partial Scan column gives the number of latches that were made scannable using our method, which includes latches inside select and toggles and also any extra latches added to the circuits as mentioned in the previous sections.

The table clearly shows that our method provides better fault coverage compared to the self-checking method of [16] for all the examples. It also shows that number of latches that need to made scannable is much smaller than the full scan approach of [31]. To make fair comparison with ALScan one should also consider that overhead due to the changes made to the C-elements in our circuits. However the circuit overhead added to the C-element in our method is also much smaller than that needed to make it fully scannable. The scannable C-element design reported for ALScan uses about 55 transistors whereas we use 20 in our design. Fewer scan latches also implies a smaller test application time.

7 Conclusions

We have described a partial scan methodology to test self-timed macromodule-based circuits. The proposed method provides good fault coverage using a stuck-at model while requiring that only a subset of the latches in the circuit be made scannable. This is a substantial improvement over full scan techniques. The fault coverage is also much better than function-based testing that relys on the self-checking property of self-timed circuits.

The technique requires minor modifications to those control modules that contain internal state such as Select and Toggle modules, and provides for a novel method for testing the resulting XOR and C-element networks that make up the bulk of the remaining control circuitry. Although the scanning of tests is done using a test clock, the asynchronous nature of the circuit is unchanged in normal operation.

The current technique is targeted specifically at the control portion of self-timed systems. We are currently working on extending the techniques to test the data path, and the timing-sensitive bundling delays that are included in the data path.

References

[1] Miron Abramovici, Melvin A. Breuer, and Arthur D. Friedman. *Digital Systems Testing and Testable Design*. Computer Science Press, 1990.

[2] K. V. Berkel, R. Burgess, J. Kessels, M. Roncken, F. Schalij, and A. Peeters. Asynchronous Circuits for Low Power: A DCC Error Corrector. *IEEE Design and Test of Computers*, summer 1994.

[3] Erik Brunvand. *Translating Concurrent Communicating Programs into Asynchronous Circuits*. PhD thesis, Carnegie Mellon University, 1991.

[4] Erik Brunvand. The NSR Processor. In *Proceedings of the 26th International Conference on System Sciences*, Maui, Hawaii, January 1993.

[5] E. Brunvand and R. Sproull. Translating Concurrent Programs into Delay-insensitive Circuits. In *Proceedings of International Conference on Computer Aided Design (ICCAD)* 1989.

[6] K. T. Cheng and V. D. Agrawal. A Partial Scan Method for Sequential Circuits with Feedback. *IEEE Transactions on Computers*,pages 544-548, 1992.

[7] V. Chickermane and J. H. Patel. A Fault Oriented Partial Scan Design Approach. In *International Test Symposium*, 1991.

[8] William J. Dally and Charles L. Seitz. The Torus Routing Chip. *Distributed Computing*, 1:187–196, 1986.

[9] A. L. Davis B. Coates and K. Steven. Automatic Synthesis of Fast Compact Self-timed Control Circuits. In *Proc. of the IFIP Working Conference on Asynchronous Design Methodologies*, 1993.

[10] R. Gupta, R. Gupta and M. A. Breuer. BALLAST:A Methodology for Partial Scan Design. In *Fault Tolerant Computing Symposium*, 1989.

[11] Pieter Hazewindus. *Testing Delay-Insensitive Circuits*. PhD thesis, California Institute of Technology, 1991.

[12] Henrik Hulgaard, Steven M. Burns, and Gaetano Borriello. Testing Asynchronous Circuits: A Survey. Technical Report UW-CSE-94-03-06, Department of Computer Science and Engineering, Univ. of Washington, Seattle, 1994.

[13] Michiel Kamp. Testing Delay-insensitive Circuits: A Case Study. Technical report, Eindhoven Institute of Technology, September 1990.

[14] Ajay Khoche and Erik Brunvand. Testing Self-Timed Circuits Using Scan Paths. *5th NASA Symposium on VLSI Design*, Nov 1993.

[15] Ajay Khoche and Erik Brunvand. A partial scan methodolgy for testing self-timed circuits. Technical Report UUCS-95-001, Department of Computer Science, Univ. of Utah, Salt Lake City, 1995.

[16] P. Kudva and V. Akella. Testing Two Phase Transition Based Self-timed Circuits in a High Level Synthesis Environment. *High Level Synthesis workshop* May 1994.

[17] H. K. T. Ma, S. Devadas, A. R. Newton, and A Sangiovanni Vincentelli. An Incomplete Scan Design Approach to Test Generation for Sequential Machines. In *International Test Symposium*, 1988.

[18] Alain J. Martin. Compiling Communicating Processes into Delay-insensitive Circuits. *Distributed Computing*, 1(3), 1986.

[19] A. J. Martin, S. M. Burns, T. K. Lee, D. Borkovic, and P. J. Hazewindus. The Design of an Asynchronous Microprocessor. *Proc. of Advanced Research in VLSI*, MIT Press, 1990.

[20] S. M. Nowick and D. Dill. Automatic Synthesis of Locally-clocked Asynchronous Machines. In *Proc. International Conf. Computer-Aided Design (ICCAD)*, 1991.

[21] Cees Niessen, C.H. (Kees) van Berkel, Martin Rem, and Ronald W.J.J. Saeijs. VLSI Programming and Silicon Compilation; A Novel Approach from Philips Research. In *ICCD*, Rye Brook, NY, October 1988.

[22] S. Pagey, G. Venkatesh, and S. Sherlekar. Issues in Fault Modeling and Testing of Micropipelines. *First Asian Test Symposium*, Hiroshima, Japan, Nov 1992.

[23] N. Paver. *The Design and Implementation of an Asynchronous Microproceesor*. PhD Thesis, University of Manchaster, UK , 1994.

[24] Marly Roncken and Ronald Saeijs. Linear Test Times for Delay-insensitive Circuits: A Compilation Strategy. *Proceedings of IFIP Working Conference on Asynchronous Design Methodologies*, Manchester, UK, 1993. Elsevier Science Publishers.

[25] Marly Roncken. Partial scan test for asynchronous circuits illustrated on a dcc error corrector. In *Proceedings of International Symposium on Advanced Research in Asynchronous Circuits and Systems*, 1994.

[26] C. L. Seitz. System Timing. In *Mead and Conway, Introduction to VLSI Systems*, chapter 7. Addison-Wesley, 1980.

[27] R. Sproull, I. E. Sutherland, and C. E. Molnar. Counterflow Pipeline Processor Architecture. *IEEE Design and Test of Computers*, Fall 1994.

[28] Ivan Sutherland. Micropipelines. *CACM*, 32(6), 1989.

[29] C.A. Traver. A Testable Model for Stoppable Clock Asics. In *Proceedings of IEEE International ASIC Conf.,1991.*

[30] S. H. Unger. *Asynchronous Sequential Switching Circuits*. John Wiley & Sons Inc., 1969.

[31] Chin-Long Wey, Ming-Der Shieh, and P. David Fisher. ASLCScan: A Scan Design Technique for Asynchronous Sequential Logic Circuits. In *International Conference on Computer Design(ICCD)*, Cambridge, Mass., October 1993.

[32] K. Y. Yun D. Dill and S. M. Nowick. Synthesis of 3d asynchronous state machines. In *Proc. International Conf. Computer-Aided Design (ICCAD)*, 1992.