View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by The University of Utah: J. Willard Marriott Digital Library

Testing Self-Timed Circuits using Partial Scan

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 both the control and data path parts of macromodule based self-timed circuits for stuck-at faults. Compared with other proposed test methods for testing control paths in self-timed circuits, this technique offers better fault coverage under a stuck-at input model than methods using self-checking properties, and requires fewer storage elements to be made scanable than full scan approaches with similar fault coverage. A new method is proposed to test the sequential network in the control path in this partial scan environment. The partial scan approach has also been applied to data paths, where structural analysis is used to select which latches should be made scannable to break cycles in the circuit. Experimental data is presented to show that high fault coverage is possible using this method with only a subset of storage elements in the control and data paths 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 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 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 inputs 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 [10, 22, 33, 35].

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 self-timed protocols at their interfaces that obey delay-insensitive or speed-independent properties to make their composition simpler [3, 20, 23, 30]. Self-timed macromodular circuits have been used in a wide variety of academic research efforts [4, 21, 25], as well as in industrial research settings [2, 29], 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, 20, 23]. In particular, the set of macromodules used by Brunvand [3, 5] and Sutherland [30] 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 both the control and data path portions of macromodule based self-timed circuits for stuck-at faults. This method

0-8186-7098-3/95 $04.00 © 1995 IEEE

provides better fault coverage in the control path than methods using the self-checking property of self-timed circuits which assumes that the circuit halts in response to faults [13]. It also requires fewer storage elements in the control path to be made scannable than full scan methods while offering acceptable fault coverage. The partial scan approach to testing the data path involves converting some of the latches in the data path into scannable latches such that cycles in the circuit can be broken.

The paper is organized as follows. In the next section, self-timed circuits will be described briefly along with some discussion on why they require different testing strategies than other circuits. Section 3 reviews related work. Section 4 presents the proposed partial-scan method for control and data paths of the circuits including the the overall architecture, modifications to the various modules, the procedures used to test sequential logic consisting of XOR and C-elements, and the modifications needed to test the data path. Section 5 presents experimental results obtained on four examples. Finally, Section 6 offers some conclusions.

2 Self-Timed Macromodule Circuits

Self-timed circuits are a subset of a broad class of asynchronous circuits which do not use a global clock for synchronization. Specifically, self-timed circuits are asynchronous circuits that generate completion signals to indicate that they are finished with their processing [28]. A signaling protocol used with the completion signal 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.

Although self-timed circuits can be designed to implement their communication protocols in a variety of ways, the circuits used in our library use two-phase transition signaling for control and a bundled protocol for data paths. Two-phase transition signaling [28] 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 [30]. This requires that the data bundle and the control wires be

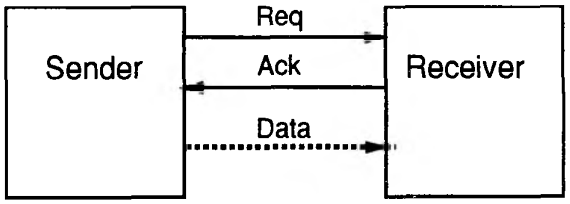

Figure 1: Two Modules Connected with a Bundled Data Path

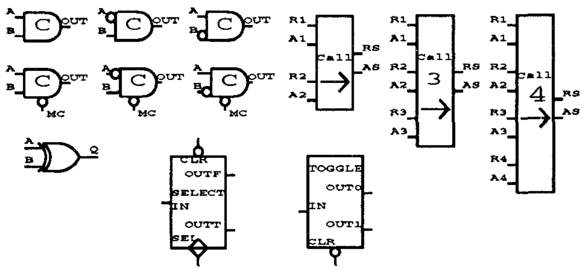

Figure 2: Control Modules for Self-Timed Designs

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. This condition is similar to, but weaker than, the equipotential constraint [28]. Two modules connected with a bundled data path are shown in Figure 1.

Our design method builds circuits using a variety of modules which communicate using two-phase transition signals. The modules used to build the control path, described in more detail elsewhere [5, 30], 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. The functionality of the main control modules is as follows:

XOR: An XOR behaves as an OR for transition signals. When a transition occurs on any of its inputs, the XOR generates a transition at its output.

C-Element: A C-element is used as an AND function for transitions. A transition occurs at the output only when there have been transitions at both of its inputs. Note that the C-element must start in a state where both inputs are at the same value to behave in this way.

A global clear signal to the control modules ensures this condition on system reset.

Select: A two-way transition Select module, in response to an input transition, causes a transition on one of two outputs depending on the value of its select (*SEL*) signal. The *SEL* signal should be valid before the input transition arrives and must remain valid until after an output transition is generated at one of the outputs. In other words, *SEL* is bundled with respect to the input transition.

Toggle: A Toggle module causes, in response to an input transition, an output transition alternately on its two outputs. After initialization, the first input transition causes a transition on *out0* and subsequent input transitions cause transitions on alternate outputs.

Call: A Call module acts as a hardware subroutine call allowing multiple requesters to access a shared resource. The Call module routes the *Req* signal from a client (for example, either *R1* or *R2* in a two-way Call) to the subroutine circuit, and after the subroutine acknowledges, routes the *Ack* back to the appropriate client. The requests must be mutually exclusive.

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. 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.

2.1 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:

- 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. In particular, races and hazards create two main types of problems in testing asynchronous circuits: the existence of a race or hazard can invalidate a test, and techniques for avoiding races and hazards in asynchronous circuits usually require the addition of redundant logic which may not be testable by any functional test.
- Self-Timed circuits operate in an autonomous way in the sense that once the control is passed from the environment to the circuit the progress is determined totally by the circuit. This is different from a synchronous circuit 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.
- In a synchronous system, it is possible to slow down the operation of the entire system simply by decreasing the clock speed to reason about noise related problems (which might interfere with testing). Self-timed circuits react to local handshake signals and there is no analogous technique for slowing system operation without modifying the circuit in other ways.
- 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 only observability mechanism is observing the primary output. A detailed discussion of these problems may be found in [17]. 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.

3 Related Work

Testing asynchronous circuits is a relatively new area. Very few attempts have been made to date. For test-

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

ing macromodule based self-timed circuits only two approaches have been reported in the literature that we know of. In [18] a functional approach is used for testing self-timed macromodule circuits using the self-checking property mentioned earlier. In this approach *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 in the control path. A method is also proposed to test the data path, where the data flow is controlled by influencing the control flow as described above. A modified D algorithm was proposed for test generation. This approach has the following drawbacks:

- 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 Tables 1 and 2.

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

- 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 [18]. State justification may be computationally complex in the type of circuits considered in this paper, which contain a lot of state distributed throughout the circuit.

- Loop structures in the control path can not be tested without adding extra observability to the circuit. Select modules are often used to build looping structures in the circuit. When faults keep the circuit in the configuration which executes the loop body, the control stays in that loop and there is no way out.

- Only pipeline style structures can be handled in the data path. The method is not clear about handling cyclic structures.

Other researchers [12, 14, 26] have also proposed methods based on the self-checking property for the control path of circuits similar to ours. However all these approaches will suffer the same disadvantages described in previous section. Hazewindus [12] 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 [15] 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 deal directly with self-timed macromodular circuits, however some of them are still relevant. In [34] a full scan approach was proposed for 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 [1]. 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. 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.

All these approaches with the exception of [12] do not consider data path testing. In [12] a method was proposed to test data path elements implemented in a specific way. This method is very tightly coupled to the way the data path elements are implemented is not applicable in general to the circuits considered in this paper.

Some other approaches have been used to test the data path of specific circuits, but these methods are not generally applicable. In [6] a scan based method was proposed to test asynchronous counter circuits in linear time. In [16, 24] methods have been proposed to test micropipeline structures. In a recent paper [27] a partial scan test is described for an asynchronous DCC error corrector circuit. However no general method was proposed to apply partial scan in the circuits. Also a stuck-at-output fault model [13] was assumed for the control path of the circuit which is less general than the stuck-at-input model [13] assumed in this paper.

4 Partial Scan Solution

As described in the previous section, if the gate to latch ratio in the control path of the circuits is low, the overhead of full scan will be high. Also due to the structure of the 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. The partial scan approach is also applied for testing the data path of the circuit by making a subset of the data path latches scanable. This provides a coherent approach for testing both control and data path circuits.

4.1 Partial Scan for the Control Path

4.1.1 Selection of Scan Latches

Current approaches in synchronous circuits for selection of scan latches are based upon testability analysis, test pattern generation, or structural analysis [7, 8, 11, 19]. In our method a combined approach involving testability analysis and structural analysis has been followed for scan latch selection. This process involves three stages:

1. Analysis of Select and Toggle elements revealed that faults inside the Select and Toggle modules are difficult to test using functional test methods [17], so all the Select and Toggle elements are added to the scan path. This partitions the remaining 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 4.3.
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 would otherwise be untestable using functional test on Call elements [17]. If this is not possible then extra transparent scannable latches are added to the circuit. These latches are added in such a way that delay incurred can be hidden. This will be explained in Section 4.1.2.
3. The last stage involves analysis of the circuit for loops not having any scannable latch in them. 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-elements should be made scannable such that many loops can be broken simultaneously.

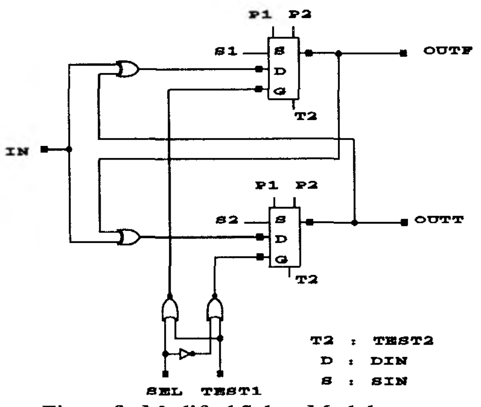

Figure 3: Modified Select Module

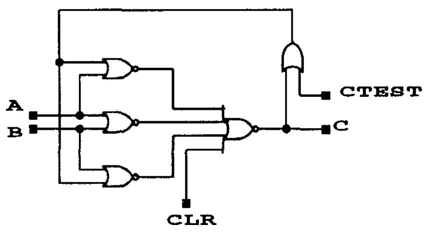

Figure 4: Modified C-element

4.1.2 Modifications to Circuit Elements

Some of the circuit modules in the library require some modification to fit into the partial scan environment.

Select Module The modified Select element is shown in Figure 3. 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 justified on the scan path. The *CLR* signal 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 output of the network under test to be captured into the the master latch.

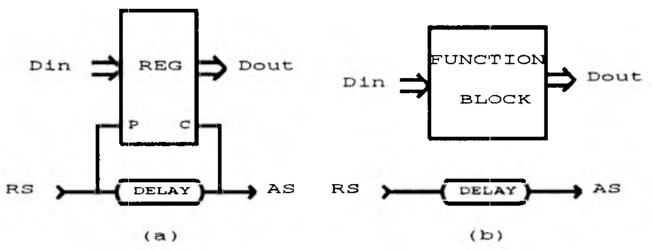

Figure 5: Register and Functional Block Modeling in the Circuits

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 its feedback wire, so if we can control the state of the feedback wire we can make it act like an AND gate or an OR gate. One way to do this could be 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 [32] 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 the modified C-element is shown in Figure 4. This gate level circuit is a model of the actual circuit which is described in terms of transistors elsewhere [31, 3]. The procedure to test C elements will be described in Section 4.3.

Modifications to Registers and Function Blocks In a Call element it is sometimes not possible to control the value of the *AS* line independent of values of *R1* and *R2*, which is required to test for certain faults. This typically happens when the Call module is used to share a register or a functional module. The way registers and functional modules are used is shown in Figure 5. In this case the *RS* line directly feeds the *AS* line through a delay element and then, due to reconvergent fanout certain faults are untestable [17]. In such a case the delay elements are converted into transparent latches during scan mode, so that the *AS* line is independently controllable. This allows us to hide the delay introduced by the extra storage elements.

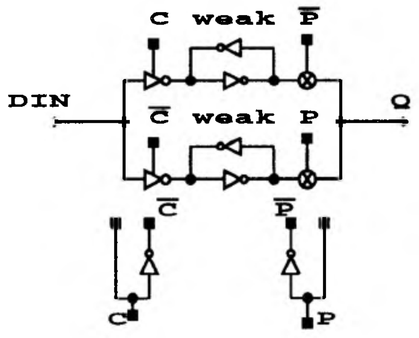

Figure 6: Basic TLNO

4.2 Partial Scan for the Data Path

4.2.1 Selection of Latches

A structural analysis method is followed to select the latches in the data path which are to be made scanable. It was shown in [7] that the two factors which affect the complexity of test generation most are cycles and sequential depth. Cycles are especially important in asynchronous circuits [1], where the control is autonomous, which results in uncontrollable number of frames in an iterative model for test generation. In our approach the latches are selected for scanning so that the cycles in the circuit can be broken. Currently the minimum number of latches that need to scanned to break all the cycles are selected manually, however techniques proposed in the literature for synchronous sequential circuits can adopted here because it is a structural property. The latches in the remaining circuits are modified as described in the next section to be transparent in test mode. This allows logic in the partitioned circuits to be tested in combinational way.

4.2.2 Modifications to Latches

The latches are modified either to be made scannable or to be transparent in test mode as described in the previous section. The latches used in the circuits considered in this paper are transition latches. Specifically the latches used in the example circuits are called Transition Latches Normally Opaque (TLNO). A schematic of a TLNO is shown in Figure 6. The state of control signals *C* and *P* decide when the state of the latch is transparent (when *C* and *P* are different) or opaque (when they are the same). These control signals can be connected as shown in Figure 5 to form a latch with a *Req/Ack* interface rather than a *C/P* interface. This circuit is modified with the addition of a single XOR gate as shown in Figure 7 such that when the test signal is asserted

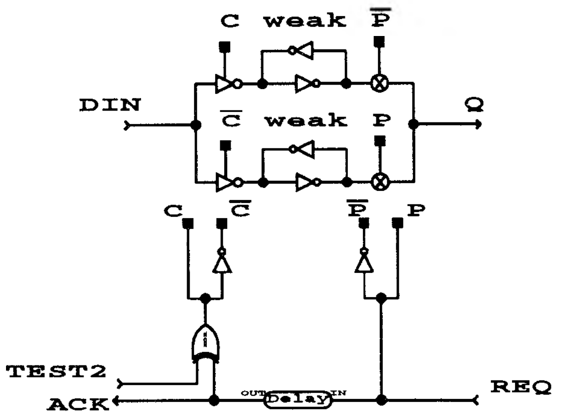

Figure 7: Transparent TLNO

the latch is transparent from input to output. This modification will not be required for a latch which is modified as described in the control path section, because the requirement for transparency can be satisfied by scanning an appropriate value in the scan path elements inserted between *C* and *P* signals.

The latches of the scan register are modified such that they act like a shift register during scanning and as a master slave register during test application. In normal mode the latches act like normal TLNOs. A schematic of a modified latch is shown in Figure 8. The latch is put in scan mode by asserting *Test1* and *Test2* control signals. Signals *P1* and *P2* provide nonoverlapping clocks to the scan path to provide a race free scan operation. Note that the clocks are used *only* during test mode. In normal mode the circuit returns to fully self-timed operation.

While in test mode, the *Test1* signal to the NOR gates disables the tristate buffers allowing the scan path to drive the latch. Similarly the *P* signal is controlled by *Test2* such that the upper pass gate is off, while the lower one is on during test mode. The input value on *DIN* is sampled during the test by deasserting the *Test1* signal which enables the input driver of upper latch (this is guaranteed by the OR gate at the bottom of the figure controlling the value of *C*). Thus the input value is captured in the master stage.

4.3 Test Procedure

A generic block diagram of the circuits is shown in Figure 9. The interaction between the control and data paths takes place in two forms: the select signals for the multiplexers in the data path are generated by the control path, and the data path generates the SEL signals for various Select elements in the control path. A single scan path is introduced across the control and data paths. The values of MUX control signals and the SEL signals can be set

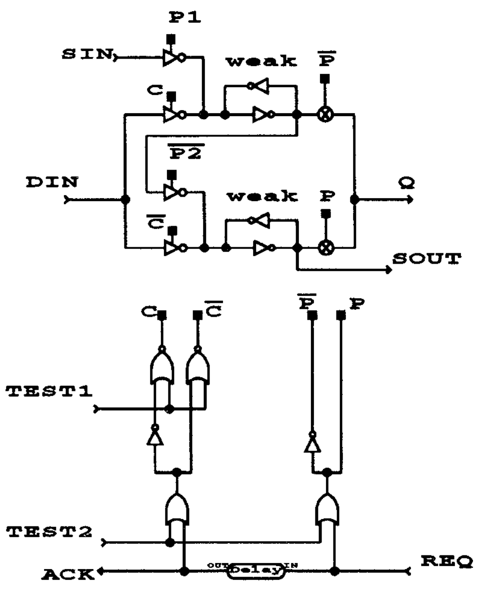

Figure 8: Scannable TLNO

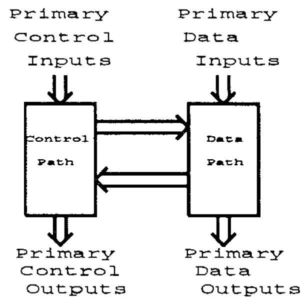

Figure 9: Block Diagram for Circuits

by scanning appropriate values into the scan path and by setting appropriate inputs. The control and data paths are tested in different phases. This section will first describe the scan path operation and then the procedure used for testing control and data paths will be described.

4.4 Scan Path Operation

The scan path and testing operations are controlled by a set of global signals consisting of *Test1*, *Test2*, *P1*, *P2*, and *Ctest*. The circuit is put into scan mode by asserting *Test1* and *Test2* control signals (shown in Figures 3 . The *P1* and *P2* signals provide two phase non-overlapping clocks to the scan register to provide a race free operation. Note that these clocks are used only to clock the scanning operation. The circuit is still self-timed in normal operation. The

Ctest signal controls the behavior of the C-elements in the circuit.

Once the circuit has settled based on the scan path inputs, its output is captured by deasserting the *Test1* signal. In the control path deasserting *Test1* enables the *SEL* signal to the latches. Enabling the *SEL* causes one of the latches to be enabled depending on the value of the *SEL* line which is input to the control path. In the data path it causes the driver of the normal input to be enabled as described in the previous section. 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 get tested during the capture process by appropriately controlling its value through the data path.

4.5 Testing the Control Path

Testing control circuits in this partial scan environment involves testing the remaining network of XOR and C-elements. The procedure to test these network is described below.

1. In the first step the C-elements are put into OR-mode by asserting the *Ctest* 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. While the first testing step covers most of the faults in the network 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. 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 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 propagate 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 4. 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 a 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.

4.6 Testing the Data Path

Testing the data path involves controlling the *Test1* and *Test2* control signals. In addition to enabling the scan path, these signals put those latches which are not part of the scan path in the data section into transparent mode. The test vector for the data path includes the test for the circuit block to be tested and the values required to be scanned in the data path to set the proper values on MUX control signals. The observation mechanism is the same as described above.

5 Experimental Results

The method described in this paper was applied to four small example self-timed macromodular circuits. The circuits are described using schematic capture and test software from Viewlogic, or in VHDL descriptions. Test vectors were generated using Attest software. The results are listed in Table 2. The router circuit is a torus-connected wormhole routing chip for message routing in a multiprocessor [3, 9]. 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 for the control path obtained by the functional test method described in [18] which relies on the self-checking property that the circuit will halt in response to a class of faults. The coverage reported here assumes that some additional observability mechanisms have been used to detect faults inside the loop body as mentioned in Section 3, otherwise the coverage will be even lower. The Partial Scan column refers to the fault coverage for the control path 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 [34] were adopted for control path. The Partial Scan column gives the number of

Design	Control Path				Data Path	
	Fault Coverage		No. of Latches Scanned			
	Self-checking	Partial Scan	ALScan	Partial Scan	Total Latches	No. Scanned
Router	65.7%	98.2%	34	17	2(4/5)	1
IFstage	66.1%	97.4%	26	13	3(16)	1
GCD	74.6%	95.5%	11	7	3(8)	1
Division	67.5%	100.0%	7	6	5(8)	2

Table 2: Comparison of Fault Coverage and Number of Scannable Latches

latches that were made scannable in the control path using our method, which includes latches inside Select and Toggle modules and also any extra latches added to the circuits as mentioned in the previous sections.

The data path columns indicate the statistics for the data path. The Total Latches column indicates the number of latches in the data path and their widths (in parenthesis). The No. Scanned column indicates the number of latches required to break all the cycles in the circuit data path.

The table shows clearly that our method provides better fault coverage for the control path compared to the self-checking method of [18] for all the examples. It also shows that the number of latches that need to made scannable is much smaller than the full scan approach of [34]. 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. The datapath column shows that very few latches were required to be made scannable to break all the cycles and be able to test the data path logic in a combinational way.

6 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 relies 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 partial scan technique is also applicable to the data path where a fraction of latches can be made scannable to break all the cycles in the circuit's data path. The remaining latches are made transparent during test mode such that the logic between scannable latches can be tested in a combinational way.

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. volume 11, Summer 1994.

[3] Erik Brunvand. *Translating Concurrent Communicating Programs into Asynchronous Circuits*. PhD thesis, Carnegie Mellon University, 1991. Available as Technical Report CMU-CS-91-198.

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

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

[6] Gerald Carson and Gaetano Borriello. A Testable Asynchronous Counter. *IEEE Journal of Solid State Circuits*, 25(4), Aug 1990.

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

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

[9] William J. Dally and Charles L. Seitz. The Torus Routing Chip. *Distributed Computing*, 1, 1986.

[10] 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.

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

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

[13] 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.

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

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

[16] Ajay Khoche and Erik Brunvand. Testing micropipelines. In *Proceedings of Intl. Symposium on Advanced Research in Asynchronous Circuits and Systems*, 1994.

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

[18] 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.

[19] 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 *Intl. Test Symposium*, 1988.

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

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

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

[23] 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.

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

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

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

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

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

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

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

[31] Ivan E. Sutherland, Robert F. Sproull, and Ian Jones. Standard Asynchronous Modules. Technical Memo 4662, Sutherland, Sproull and Associates, 1986.

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

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

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

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