A Partial Scan Methodology for Testing Self-Timed Circuits

Ajay Khoche and Erik Brunvand

UUCS-95-001

Department of Computer Science
University of Utah
Salt Lake City, UT 84112 USA

January 31, 1995

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.

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 and asynchronous environment. Many approaches have been proposed to design control circuits in this style [13, 23, 34, 36].

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 [4, 21, 24, 31]. Self-timed macromodular control circuits have been used an a wide variety of academic research efforts [5, 22, 26], as well as in industrial research settings [2, 30], 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 [4, 21, 24]. In particular, the set of macromodules used by Brunvand [4, 6] and Sutherland [31] 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 [16] 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.

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 individual macromodules to asses the requirements for getting good fault coverage and to show the limitations of methods which use self-checking properties. Section 4 reviews related work. Section 5 presents the proposed partial-scan method including the the overall architecture, and 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 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 [29]. 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.

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 [29] 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.

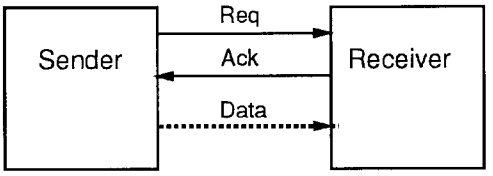

Figure 1: Bundled Data Path Connection

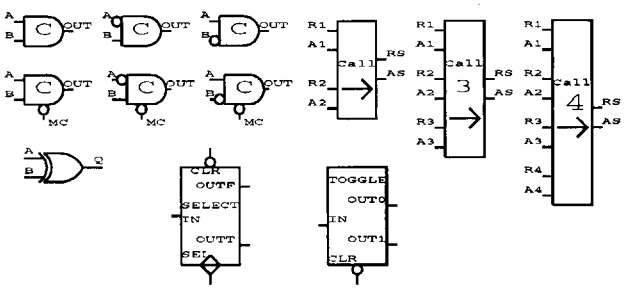

Figure 2: Control Modules for Self-Timed Designs

A bundled data path uses a single set of control wires to indicate the validity of a *bundle* of data wires [31]. 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. This condition is similar to, but weaker than, the equipotential constraint [29]. Two 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 [3, 6, 31], 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

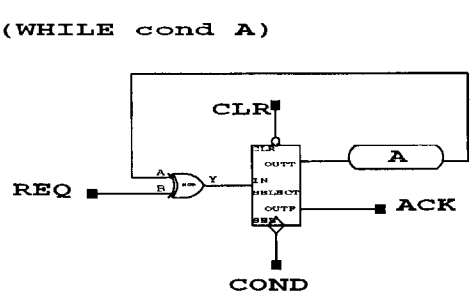

Figure 3: Translation of WHILE construct into Circuit

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 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 [4, 8]. 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 [4], to a simple wormhole router designed for a mesh-connected multiprocessor array [7]. This library has also been used to build large circuits by hand, including a self-timed microprocessor [5], 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:

- 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 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 synchronous circuit where 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.

3.1 Functional Testability of Individual Modules

In this section we discuss the testability of individual modules described in the previous section and

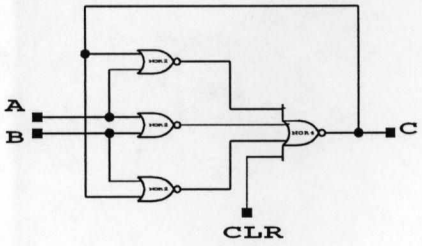

Figure 4: Gate Level Model of C element

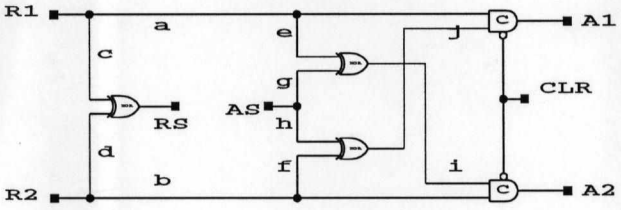

Figure 5: Basic Call Module

the effect of the environment in which these modules are used. This in turn will indicate the limitation of testing methods which use the functional behavior of circuits to generate tests [19].

3.1.1 C-element

A gate level model of the 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 [32, 4]. As described in the previous section, a C-element acts as an AND gate for transitions. It is usually used in self-timed circuits to implement a join operation for two forking processes. In this context during the circuit operation both inputs receive the same signal value at any time in the steady state. The state where the the two inputs are different represents the time when one fork has completed and the other has not yet finished, and is therefore transient. The transient state can not be used to test the C elements reliably as the effect of a fault in this transient state is a correct transition to the final state, but one that happens prematurely. In order to test for faults activated in this way, inputs that are normally transient must be applied for a sufficient time to detect the faults. The fault coverage in the steady state when the two inputs are same will be the coverage obtained by tests 00 and 11. Functional tests of a C-element will not cover all possible faults inside the C-element. The fault coverages for these tests are shown in Table 1.

3.1.2 Call Element

As for the C-elements described above, the C-elements in the Call module shown in Figure 5 also have the same behavior of both the inputs being same in the steady state. Faults inside the C-elements described above will again not be detected by any functional test. In addition the faults on lines e and f in Figure 5 are also not detectable by functional tests as described below.

We will only describe faults on the wire labeled f in Figure 5 as the case of wire e is similar. Assume there is a stuck-at-1 (s-a-1) fault on line f. Also assume that the delay between an event on RS and the following AS event is d1, and that the delay from $R1$ to the corresponding C-element input is d2. In all practical circuits d1 > d2. Now when a rising event occurs on input $R1$, in the faulty case an event on $A1$ is produced immediately without waiting for AS to arrive. Note that this event on $A1$ is the correct event, however it appears earlier than desired. This happens because the s-a-1 fault on f prematurely puts a 1 at the corresponding input of the C-element. The following event on AS does not cause any event on the output of the C-element as a C-element requires changes on both inputs before it fires. In fact this later transition on the AS wire causes the firing of the next premature transition as the Call module is continued to be used.

Now assume that f is s-a-0. To activate this fault the value of input $R2$ has to be 1. Assume $R1$ starts at 0 and then rises to 1. This scenario implies that the value of AS is 1 before the transition on $R1$ arrives. The transition on the input of the C-element corresponding to this input transition has arrived before the corresponding transition on AS and hence the C-element will again fire prematurely. Again the value of $A1$ will be the desired one except that it will occur too early. The case of e is similar. Thus the faults on e and f result in premature transitions rather than incorrect transitions and are therefore not testable. The only way to see the effect of these faults is in the data path where data may be latched at the wrong time, but this requires very strict timing control on the value in the data path which is very difficult or may even be impossible depending on where the control element is in the circuit.

3.1.3 Select Element

A schematic of a Select module is shown in Figure 6. Faults on all nets except the feedback lines e, f and the latch gating signals are testable using the self-checking property. The behavior of a Select module under faults

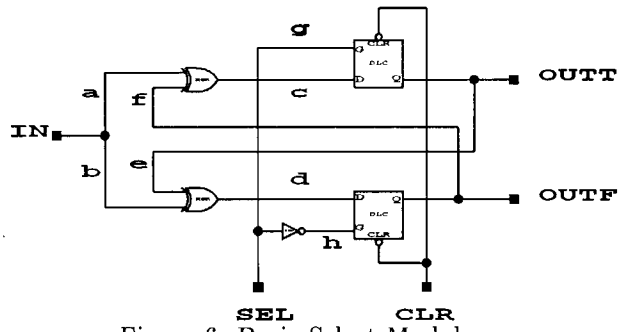

Figure 6: Basic Select Module

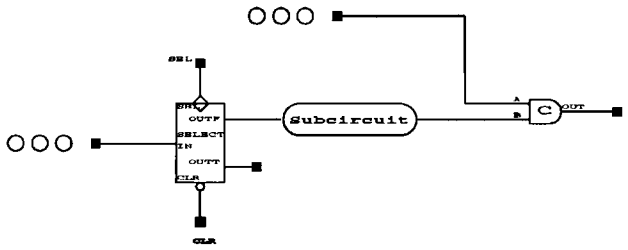

Figure 7: Premature Firing in Select Element

on feedback lines and latch gate signals is described below.

Faults on the feedback lines e and f result in correct, but premature transitions. A stuck-at-1 fault results in a premature transition in the initial state on $OUTT$ or $OUTF$ during functional test depending on the initial value of the SEL line. Both of these faults can be tested only if the SEL line is controllable. Stuck-at-0 faults on these lines can be activated only in state $OUTT,OUTF$ = 1,1. Thus testing of these faults will require a state justification process in the functional test. Assuming the justification is possible it may still be the case that these faults are not observable. Assume that the fault effect appears on line $OUTT$ and there is a path from this line to a C-element elsewhere in the circuit as shown in Figure 7. In this case the the fault effect will be masked if the other input of the C-element arrives before the premature transition is nullified. In that case a correct transition will be produced at the C-element output in spite of the fault.

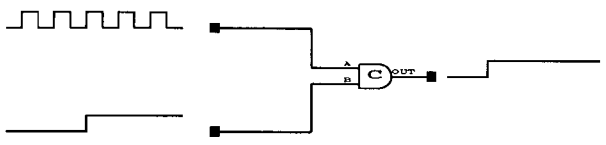

Figure 8: Filtering of Oscillations Through C Element

A fault on the SEL line results in one of the latches being permanently off and other permanently on. Thus one of the branches of the Select is always taken, and the other never is. This effect can be observed on the output if and only if the branch is individually observable. The reason for this lies in the way a Select is usually used to perform conditional execution of different circuits based on the Boolean SEL signal. The two branches of the Select are typically combined through a Merge (XOR) element to produce a single acknowledgement of the conditional block. It is not possible to distinguish which branch was taken if only the output of the block is observed. The only effects that might be seen are in the data path, which will be difficult to control, especially in asynchronous circuits where the operation of circuits is autonomous.

A fault on the signals controlling the latch gates which results in latch being permanently closed is testable as it results in the circuit halting. Faults which result in the latch being being permanently open however, cause oscillations in certain states where both the latches get enabled simulatneously in faulty case. These oscillations can be detected only if there is a path from any of these Select outputs to the circuit output without encountering any C-elements. A C-element acts as perfect filter for such oscillations as shown Figure 8.

3.1.4 Toggle Element

All the faults except those on latch gating lines are testable using functional tests. Faults on gating signals which result in one of the latches being permanently off are also testable. Faults which cause a latch to be permanently enabled cause oscillations in certain states as in the Select elements described above. In these states both of the latches are enabled thus forming a loop with odd number of inversions.

4 Related Work

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

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

For testing macromodule based self-timed circuits only two approaches have been reported in the literature that we know of. In [19] a functional approach is used for testing self-timed macromodule circuits using 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. 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 [19]. 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 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 faults keep the circuit in the configuration which executes the loop body, the control stays in that loop and there is no way out.

Other researchers [15, 17, 28] 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 [15], 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 [18] 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 [35] 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 [10, 11, 14, 20]. In

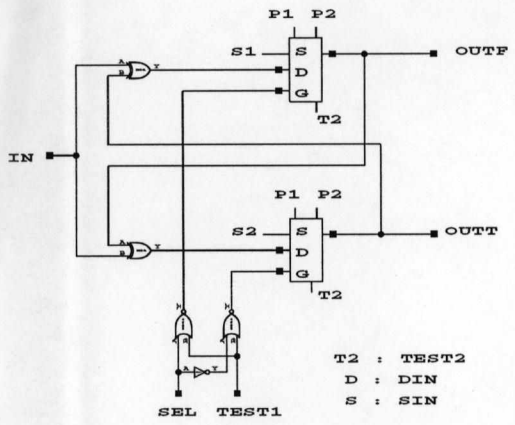

Figure 9: Modified Select Module

our method a combined approach involving testability analysis and structural analysis has been followed for scan latch selection. This process involves three stages:

1. As described in Section 3.1, faults inside the Select and Toggle modules are difficult to test using functional test methods, 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 were described in Section 3.1 to be untestable using functional test on Call elements. 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 5.2.

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. A software has been developed to detect these conditions and to suggest which C-element to be made scannable such that many loops can be broken simultaneously.

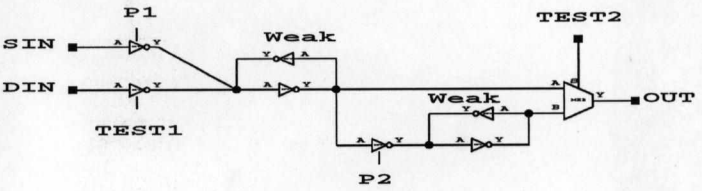

Figure 10: Modified Latch

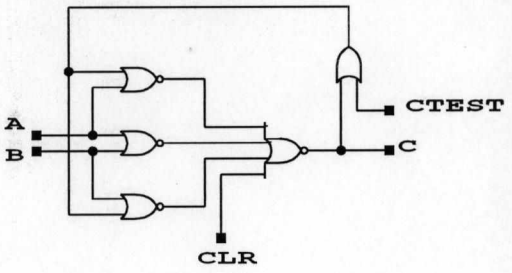

Figure 11: Modified C-element

5.2 Modifications to Circuit Elements

Each of the circuit modules in the library requires some modification to fit into the partial scan environment.

5.2.1 Select Module

The modified Select element is shown in Figure 9. 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 modified latch is shown in Figure 10.

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.

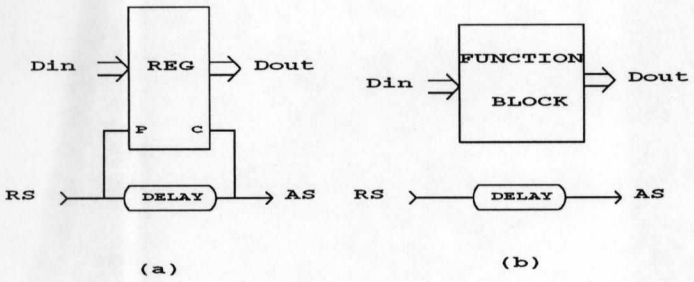

Figure 12: Register and Functional Block Modelling in the Circuits

5.2.2 C-element

The C-elements are in general not included in the scan path as describe 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-elements is the state of feedback wire, so if we could control the state of 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 [33] 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 modified C-element is shown in Figure 11. The procedure to test C elements will be described in Section 5.3.

5.2.3 Modifications to Registers and Functional 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 as described in Section 3.1. 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 12. In this case the *RS* line directly feeds the *AS* line through a delay element and then, due to reconvergent fanout, faults mentioned in Section 3.1 are not testable. 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.

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 to 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 9 and Figure 10). *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 get tested during the capture process by appropriately controlling its value.

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. While 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. This is required because otherwise during the scan-in operation different inputs will appear at the input

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

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 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 11. 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 [7, 12]. The circuit called IFstage is the instruction fetch unit of the NSR, a self-timed pipelined RISC processor [5]. 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 the functional test method described in [19] which 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. 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 [35] 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 [19] 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 [35]. 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. volume 11, pages 22–32, Summer 1994.

[3] Erik Brunvand. Using FPGAs to Implement Self-Timed Systems. *Journal of VLSI Signal Processing,* 6:173–190, 1993.

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

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

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

[7] Erik Brunvand, Nick Michell, and Kent Smith. A Comparison of Self-timed Design Using FPGA, CMOS, and GaAs technologies. In *International Conference on Computer Design (ICCD),* Cambridge, Mass., October 1992.

[8] Erik Brunvand and Mike Starkey. An Integrated Environment for the Design and Simulation of Self-timed systems. In *VLSI-91.* IFIP, August 1991.

[9] Gerald Carson and Gaetano Borriello. A Testable Asynchronous Counter. *IEEE Journal of Solid State Circuits,* 25(4):952–960, Aug 1990.

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

[11] V. Chickermane and J. H. Patel. A Fault Oriented Partial Scan Design Approach. In *International Test Symposium,* pages 400–403, 1991.

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

[13] 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.

[14] R. Gupta, R. Gupta and M. A. Breuer. BALLAST:A Methodology for Partial Scan Design. In *Fault Tolerant Computing Symposium,* pages 118–125, 1989.

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

[16] 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.

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

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

[19] 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.

[20] 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,* pages 730–734, 1988.

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

[22] A. J. Martin, S. M. Burns, T. K. Lee, D. Borkovic, and P. J. Hazewindus. The Design of an Asynchronous Microprocessor. In Carlo H. Séquin, editor, *Advanced Research in VLSI: Proceedings of Decennial Caltech Conference on VLSI*. MIT Press, 1990.

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

[24] 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.

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

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

[27] Gianfranco R. Putzolu and J. Paul Roth. A Heuristic Algorithm for Testing of Asynchronous Circuits. *IEEE Transactions on Computers*, c-20(6):639–647, June 1971.

[28] Marly Roncken and Ronald Saeijs. Linear Test Times for Delay-insensitive Circuits: A Compilation Strategy. In S. Furber and M. Edwards, editors, *Proceedings of IFIP Working Conference on Asynchronous Design Methodologies*, pages 13–27, Manchester, UK, 31 March – 2 April 1993, 1993. Elsevier Science Publishers.

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

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

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

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

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

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

[35] 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.

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