6 Conclusions

We have given necessary and sufficient conditions under which the effect of SSAFs can be characterized and sufficient conditions which ensure the testability of SSAFs in asynchronous control circuits with delay assumptions. The intuition given for these conditions provide a foundation on which synthesis for testability of such circuits can be based. We have also presented an automated testability checker, which analyzes the testability of a circuit implementation, checks all possible effects of SAFs, and determines which SAFs are not testable.

The conditions for testability can be incorporated into an automated system for the synthesis of testable "timed" asynchronous circuits. However, before this goal can be reached, automated synthesis of asynchronous control circuits under timing and gate delay assumptions needs to be addressed. In the mean time, the testability checker can serve as an analysis tool for choosing design alternatives to ensure a fully SSAF testable implementation.

Before testability of asynchronous circuits is perceived to be understood to the point where people do not use it as a reason against the use of asynchronous circuits, the fault model used must be expanded and the sufficient conditions given here must be tightened. One hope for asynchronous control circuits is that many faults in "timed" circuits will halt the circuit (without necessarily fixing the output), as they do in speed-independent circuits. Further research is needed to characterize these faults and develop an efficient algorithm which finds them in a circuit.

6 Conclusions

We have given necessary and sufficient conditions under which the effect of SSAFs can be characterized and sufficient conditions which ensure the testability of SSAFs in asynchronous control circuits with delay assumptions. The intuition given for these conditions provide a foundation on which synthesis for testability of such circuits can be based. We have also presented an automated testability checker, which analyzes the testability of a circuit implementation, checks all possible effects of SAFs, and determines which SAFs are not testable.

The conditions for testability can be incorporated into an automated system for the synthesis of testable "timed" asynchronous circuits. However, before this goal can be reached, automated synthesis of asynchronous control circuits under timing and gate delay assumptions needs to be addressed. In the mean time, the testability checker can serve as an analysis tool for choosing design alternatives to ensure a fully SSAF testable implementation.

Before testability of asynchronous circuits is perceived to be understood to the point where people do not use it as a reason against the use of asynchronous circuits, the fault model used must be expanded and the sufficient conditions given here must be tightened. One hope for asynchronous control circuits is that many faults in "timed" circuits will halt the circuit (without necessarily fixing the output), as they do in speed-independent circuits. Further research is needed to characterize these faults and develop an efficient algorithm which finds them in a circuit.

6 Conclusions

We have given necessary and sufficient conditions under which the effect of SSAFs can be characterized and sufficient conditions which ensure the testability of SSAFs in asynchronous control circuits with delay assumptions. The intuition given for these conditions provide a foundation on which synthesis for testability of such circuits can be based. We have also presented an automated testability checker, which analyzes the testability of a circuit implementation, checks all possible effects of SAFs, and determines which SAFs are not testable.

The conditions for testability can be incorporated into an automated system for the synthesis of testable "timed" asynchronous circuits. However, before this goal can be reached, automated synthesis of asynchronous control circuits under timing and gate delay assumptions needs to be addressed. In the mean time, the testability checker can serve as an analysis tool for choosing design alternatives to ensure a fully SSAF testable implementation.

Before testability of asynchronous circuits is perceived to be understood to the point where people do not use it as a reason against the use of asynchronous circuits, the fault model used must be expanded and the sufficient conditions given here must be tightened. One hope for asynchronous control circuits is that many faults in "timed" circuits will halt the circuit (without necessarily fixing the output), as they do in speed-independent circuits. Further research is needed to characterize these faults and develop an efficient algorithm which finds them in a circuit.

6 Conclusions

We have given necessary and sufficient conditions under which the effect of SSAFs can be characterized and sufficient conditions which ensure the testability of SSAFs in asynchronous control circuits with delay assumptions. The intuition given for these conditions provide a foundation on which synthesis for testability of such circuits can be based. We have also presented an automated testability checker, which analyzes the testability of a circuit implementation, checks all possible effects of SAFs, and determines which SAFs are not testable.

The conditions for testability can be incorporated into an automated system for the synthesis of testable "timed" asynchronous circuits. However, before this goal can be reached, automated synthesis of asynchronous control circuits under timing and gate delay assumptions needs to be addressed. In the mean time, the testability checker can serve as an analysis tool for choosing design alternatives to ensure a fully SSAF testable implementation.

Before testability of asynchronous circuits is perceived to be understood to the point where people do not use it as a reason against the use of asynchronous circuits, the fault model used must be expanded and the sufficient conditions given here must be tightened. One hope for asynchronous control circuits is that many faults in "timed" circuits will halt the circuit (without necessarily fixing the output), as they do in speed-independent circuits. Further research is needed to characterize these faults and develop an efficient algorithm which finds them in a circuit.

SSAF Testability Checker Results					
Circuit	No. of Reachable States	No. of State Transitions Checked	No. of TSPs Checked	Fully SSAF Testable?	CPU-Time
Pipeline Handshake	19	28	0	Yes	0.0 sec
DeMux	121	296	350	Yes	0.2 sec
Mux	355	909	1431	Yes	0.7 sec
Memory Interface	125	421	12792	Yes	3.3 sec
"Timed" Pipeline Handshake	16	23	11	Yes	0.0 sec
"Timed" Memory Interface	115	206	361	Yes	0.9 sec

Table 1: Number of reachables states, state transitions, TSPs, and run-time statistics given by the automated testability checker.

circuit, shown in Figure 3, is a memory interface circuit with a user-choice of the read/write mode.

The fifth example is a "timed" version of the pipeline handshake circuit depicted Figure 2. It can be derived by replacing the NAND gate labeled A2 in Figure 2 with a direct connection to the C-element from $\overline{Ain}$. It operates correctly only if the AND gate is slower than the bottom C-element. Our testability checker has shown that with this delay assumption the circuit is fully testable.

The sixth example is a "timed" version of the memory interface circuit, as shown in Figure 4. By assuming that the environment is relatively slower than the circuit, one gate could be removed from the speed-independent version shown in Figure 3, reducing the area of the circuit while maintaining a hazard-free design and retaining 100% SSAF testability.

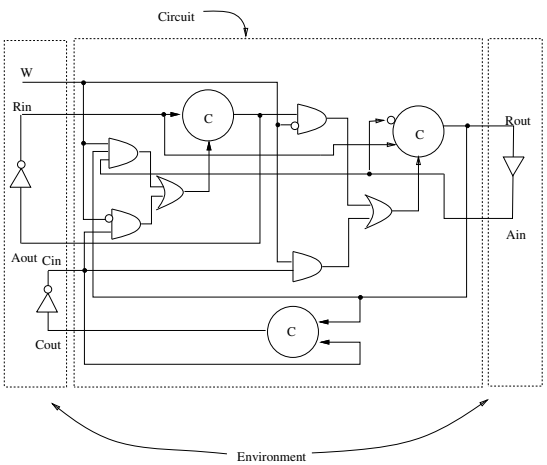

Figure 4: Fully SSAF testable "timed" version of the memory interface circuit derived from Figure 3.

6 Conclusions

We have given necessary and sufficient conditions under which the effect of SSAFs can be characterized and provided sufficient conditions that ensure the testability of SSAFs in "timed" asynchronous control circuits with delay assumptions. The intuition given for these conditions provide a foundation on which synthesis for testability of such circuits can be based. We have also presented an automated testability checker which analyzes the testability of a circuit implementation, checks all possible effects of SAFs, and determines which SAFs are not testable.

The conditions for testability can be incorporated into an automated system for the synthesis of testable "timed" asynchronous circuits. However, before this goal can be reached, automated synthesis of asynchronous control circuits under timing and gate delay assumptions needs to be addressed. In the mean time, the testability checker can serve as an analysis tool for choosing design alternatives to ensure a fully SSAF testable implementation.

Before testability of asynchronous circuits is perceived to be a well-understood subject so that people do not use it as a reason against the use of asynchronous circuits, the fault model used must be expanded and the sufficient conditions given here must be tightened. One advantage of using asynchronous control circuits is in fact that many internal faults in these circuits will halt the circuit. Further research is directed to characterizing these faults and developing algorithms that can locate them efficiently.

References

[1] Peter A. Beerel and Teresa H. Y. Meng, "Semi-Modularity and Self-Diagnostic Asynchronous Control Circuits", *Proceedings of Advanced Research in VLSI Conference,* 1990.

[2] Peter A. Beerel and Teresa H. Y. Meng, "Testability of Asynchronous Self-Timed Circuits with Delay Assumptions", to be submitted to the *IEEE Transactions on Computer Aided Design.*

[3] Tam-Anh Chu, *Synthesis of Self-Timed VLSI Circuits from Graph-theoretic Specifications*, PhD thesis, Massachusetts Institute of Technology, 1987.

[4] Srinivas Devadas, Hi-Keung Tony Ma, A. Richard Newton, and Alberto Sangiovanni-Vincentelli, "A Synthesis and Optimation Procedure for Fully and Easily Testable Sequential Machines", *IEEE Transactions on Computer-Aided Design*, October 1989.

[5] Michiel Kamps, "Testing of Delay-Insensitive Circuits: A Case Study", *Designing Delay-Insensitve Circuits*, Eindhoven University of Techology, September 1990.

[6] Alain J. Martin and Pieter J. Hazewindus, "Tesing Delay-Insensitive Circuits", *Proceedings of Advanced Research in VLSI Conference,* 1990.

[7] Edward J. McCluskey, *Logic Design Principles with Emphasis on Testable Semicustom Circuits*, Prentice-Hall, Englewood Cliffs, NJ, 1986.

[8] Teresa H. Y. Meng, Robert W. Brodersen, and David G. Messershmitt, "Automatic Synthesis of Asynchronous Circuits from High-Level Specifications", *IEEE Transactions on Computer-Aided Design*, November 1989.

A sink state of a SAF-TSP is a state in which either the circuit is expected to stop, or the SAF prevents the only unstable transition out of that state. Hence, either the sink state is a stable state in the SAF-free circuit, or it is an unstable state with a single unstable transition leading out of it which is prevented by the SAF under the conditions described earlier. It should be noted that the state in which the circuit may stabilize after initial fault propagation may not be unique.

4 Sufficient Conditions for SSAF Testability

Detecting a SAF in an asynchronous control circuit consisting of two stages: find an initial state in which the SAF will propagate to a stable state in which the outputs are different than expected; and determine if this state can be reached with our testing condition. The complication is that there exists a host of unstable states in which fault propagation for the same SAF may occur. If any of these unstable states are visited in the process of reaching the desired test state, we cannot be assured that the SAF will not cause fault propagation earlier than desired.

It is clear that if the expected sink state and the actual sink state have different outputs the SAF is detected. If the circuit ends up in a state in which the outputs are the same but the memory elements are different it is not clear that additional testing will detect the fault. This state may behave unexpectedly to scheduled subsequent input changes. While states such as these can be further simulated with additional input changes to see if the fault can subsequently be detected, the added complexity to the testing procedure would be high. Instead, if there exists a fault and the sink sate of its SAF-TSP has the correct output and incorrect memory, we state that the fault is undetectable.

We now formally state sufficient conditions for 100% SSAF testability of an asynchronous control circuit:

Theorem 4.1 *An asynchronous control circuit is 100% single-stuck-at-fault testable if the following two conditions are satisfied:*

1. *For all faults, there exists a reachable state in which the fault will propagate (by either preventing a transition or creating an unexpected transition) and leave the circuit in a state in which the outputs are different than expected.*
2. *For all faults, there does not exist a valid state (stable or unstable) in which fault propagation can occur and leave the circuit in a state in which the outputs are as expected but the memory elements outputs are different.*

This theorem in conjunction with the necessary and sufficient conditions which govern the effects of all SSAFs as described in Section 3 define sufficient conditions to analyze single-stuck-at-fault testability of asynchronous control circuits. The first condition indicates that for all faults there exists a reachable stable state from which the fault will be detected. The second condition ensures that no state which can be reached on the way to the test state will make the SAF undetectable. Hence, either we will reach the desired stable test state, or we will detect the fault earlier. In both cases the fault will be detected. A formal proof of this theorem can be found in [2]. As described earlier, we tighten these conditions by adding that all faults which fix an output of a circuit are always detectable.

We can expand the above results to circuits with input conditionals fairly easily. Input conditionals allow the circuit to be in one of many **modes,** which can be controllably entered by changing external conditional signals. Simply stated, faults now have multiple chances of being detected, one chance in each mode of operation.

5 Automated Testability Checker

An automated tool has been developed which uses the theory presented in the previous section to determine if a "timed" asynchronous control circuit is fully testable for SSAFs. The input to the program is a complete circuit implementation with minimum and maximum gate delays assigned to each internal gate. Reset states need to be specified for all possible combinations of input conditionals. The output is a declaration of the testability of every SSAF in the circuit and a list of states traversed for testing, providing the sequence of input test vectors required to test the circuit. In addition, the checker provides details of all SAF-TSPs, giving the transition path in each SAF-TSP as well as the total number of the SAF-TSPs found.

The testability analysis, as in the synchronous case [4], can be decomposed into two parts: 1) assume state justification is possible, i.e., either the desired test state is reached or the fault is detected early, and then show that the faults is detectable; 2) demonstrate state justification. Hence, when a SAF-TSP or an unstable state made stable satisfies the first condition of Theorem 4.1, the fault which caused it is detectable, assuming state justification is possible. Once all the SAF-TSPs and unstable states made stable are found, the second condition of Theorem 4.1 is tested which ensures state justification for all faults.

5.1 Circuit Examples

Table 1 presents performance figures and run-time efficiency of the SSAF testability checker obtained using a DECstation 3100. The first four examples in the table have been shown to be fully testable for SSAFs under speed-independence assumptions [1]. As expected, they are also found to be fully testable by our testability checker under liberal minimum and maximum rise and fall times delays on all gates.

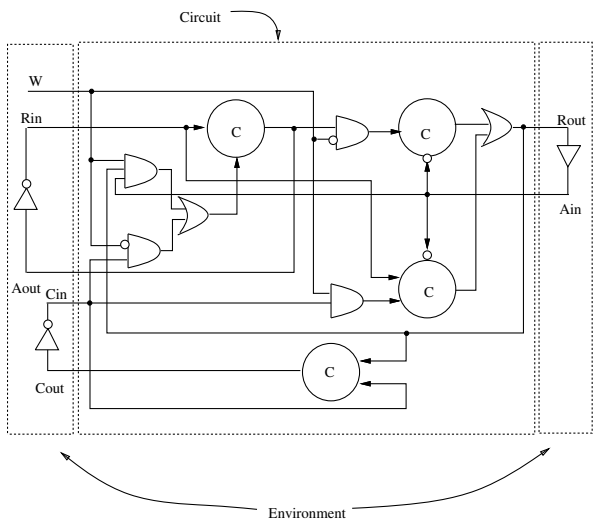

Figure 3: Fully SSAF testable implementation of a memory interface circuit.

The first example is the pipeline handshake circuit presented in Figure 2. The second and third circuits are implementations of a demultiplexer and multiplexer described in [8]. The fourth

- In state q, to falsely propagate a 1 to the output of the CLB, the output must be 0.

The first condition ensures that only one AND gate can be on; a SA-0 fault on this AND gate can achieve the desired fault propagation. The motivation behind the second condition is simple: the effect of a SA-1 fault on the output of any AND gate will be seen as long as the intended output of the CLB is 0. In both cases a "wrong" value is propagated to the output of the CLB.

These conditions are necessary but not sufficient; the conditions do not ensure that fault propagation will occur. In particular, the circuit may leave a state in which these conditions are satisfied before the fault propagation occurs, entering a state in which the conditions do not hold, in which fault propagation is prevented. This scenario is the main motivation for testing the circuit in fundamental mode. In fundamental mode, we are at least ensured that fault propagation will occur if these conditions are satisfied in stable states. Even under our testing conditions, we are still not assured that a faulty propagation will occur in an unstable state, although it may occur.

If and only if the circuit remains in state q long enough, will the SAF propagate to the output of the CLB. If the CLB block is part of an OLB block, then the propagation either will prevent a transition or cause an unexpected transition of the output signal. Hence the necessary and sufficient condition for all faults in an OLB to effect the state of the circuit is that the above conditions hold in at least one state q for each fault, and that the circuit remains in state q long enough for fault propagation to occur.

While propagation of faults in an OLB immediately effect the state of the circuit, fault propagation in a NSB must first go through a C-element. As in OLBs, fault propagation can occur by either preventing a transition or causing an unexpected transition of a state variable, in this case, the output of a C-element. For the fault to prevent a transition of a C-element, the circuit should be in a state where a C-element is excited. For a C-element with output C and inputs A and B, the conditions are: $C = 0, A = 1, B = 1$ or $C = 1, A = 0, B = 0$. For the fault to excite an unexpected transition, without loss of generality we present the condition for the fault to be in the CLB which leads to the A input of a C-element: $C = 0, A = 0, B = 1$ or $C = 1, A = 1, B = 0$. We conclude that the necessary and sufficient condition for all faults in a NSB to affect the state of the circuit is that the above conditions hold in at least one state q for each SAF, and that the circuit remains in state q long enough for the fault propagation to occur.

3.2 SAF in a C-element

Unlike in [5] we consider a C-element to be an atomic gate, as it is usually implemented by a stack of four transistors. Consequently we consider only SAFs at one of the two inputs and at the output of a C-element. The conditions on a C-element which allow fault propagation through its input A to its output are the same conditions which cause a fault on input A to be propagated to the output of a C-element, as described in the previous subsection. Hence, we are left with the trivial case of faults at the output of a C-element.

A fault on the output of a C-element is excited in states in which the expected output of the C-element is different than the value fixed by the SAF. It is not possible for a fault on a C-element output to cause an unexpected transition of state; it can only prevent a transition. However, a SAF may prevent the circuit from reaching its reset state during initialization, a more subtle practical issue that needs to be considered. Because the initialization circuitry is not usually involved in testability analysis, we do not want to derive testability conditions based on a particular reset state of the circuit. Hence, if the SAF fixes the output of a C-element at a different value than the reset state, we assume the circuit reaches its intended initialization state (which must be stable) but then due to the SAF is immediately driven to a different state. It is easy to adapt the testability conditions given in this paper to consider the particular reset state of a circuit.

3.3 SAF Effect-Stabilization

After a SAF causes the initial deviation from the circuit's expected behavior as specified by the total state graph, the circuit may not immediately stabilize. Circuit stabilization is a stage of fault propagation referred to as **SAF effect-stabilization**. In this subsection we analyze this stage by characterizing the path of states that the circuit traverses and the state in which the circuit stabilizes after initial fault excitation occurs, i.e., we characterize the unstable states and the sink state of all SAF-TSPs.

There are two cases to consider: when a SAF propagation prevents a transition and when a SAF propagation causes an unexpected transition.

Consider first the case where a SAF propagation prevents a transition in an unstable state u. If there is only one possible transition out of u, then the fault forces the circuit to stabilize in state u without going through any other transient states. On the other hand, if there are multiple enabled transitions in state u, then any one of these other transitions can move the circuit into a new state. Hence, the circuit stabilizes when a fault prevents the only enabled unstable transition in a state.

Next, consider the case where a SAF propagation causes an unexpected transition. The SAF forces the circuit into a new state, which may be stable or unstable. If the state is unstable in the faulty circuit then the circuit moves through unstable states until it stabilizes. In most cases the unstable states that the faulty circuit traverses through represent part of a TSP in the fault-free circuit. This is because a SAF cannot cause two unexpected transitions without the circuit first stablizing. To examine this claim consider the possible locations and types of SAFs. Any stuck-at-fault on a C-element input or output either prevents a C-element from either firing high or firing low, hence the C-element can only fire once in a SAF-TSP. Any stuck-at-1 fault in a CLB fixes the output of the CLB at 1, and also prevents an output signal from either firing high or firing low. For a stuck-at-0 fault in a CLB to cause two unexpected transitions, it can easily be shown that a TSP must include both the rising and falling transitions of a primary I/O or memory element in the SAF-free circuit. However, as a result of the hazard-free assumption of the circuit, we are assured that no primary I/O or memory element will have this behavior in the SAF-free circuit and hence no stuck-at-0 fault will cause two unexpected transitions without the circuit first stabilizing.

On the other hand, the unstable states in a SAF-TSP may not always represent part of a TSP. A SAF can cause an unexpected signal transition in an unstable state u, advancing the circuit into state n, and after entering state n, the subsequent transitions can be different than as expected by the TSG. These unexpected subsequent transitions are transitions which are enabled in u, but are not enabled in n. Even though the enabling conditions of the these transitions no longer hold, due to relative delays in the circuit, they may still fire.

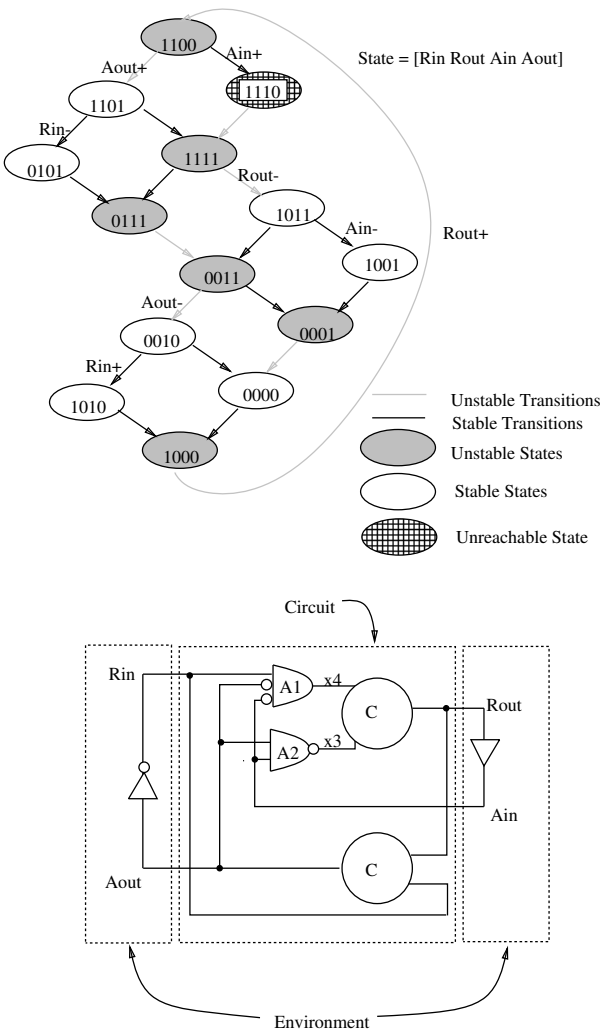

Figure 2: The reduced state graph and a complete circuit implementation of the pipeline handshake circuit shown in Figure 1.

circuits by definition need to satisfy a set of gate delay assumptions. This operating condition is commonly known as running the circuit in *fundamental mode* [7]. The circuits discussed here are not normally restricted to running in fundamental mode; we advocate only that they are tested in fundamental mode. Since all internal nodes have stabilized in stable states, stable states need only be visited once to determine what sequence of states can be reached from them, simplying the testing task. If this had not been the case, all stable states will have to be repeatedly visited under different gate delay assumptions, making the testability problem virtually intractable. Furthermore, the slow testing conditions reduces the number of valid states that are reachable, which limits the possible states which the circuit can be tested in and hence speeds up the testing time.

Since the circuit is hazard-free, we are ensured that no sequence of unstable states in the valid state graph can enable both the rising and falling transitions of a C-element, as this would represent a voltage spike at the output of the C-element. For simplicity, we will assume the same is true in the invalid state graph as well. (The adjustment of the theory to remove this assumption is minor and this assumption is not made in the automated program presented.) With this assumption it can be easily shown that if CLBs are either POSs or SOPs, no SSAF will cause internal state oscillations (C-element outputs repeatedly rising and falling with no change of inputs) [2]. Hence any SSAF circuit is ensured to stabilize after some finite amount of time after a primary input has changed.

The sufficient conditions for testability that we give do not rely on the testing equipment's ability to detect the state of a circuit when the circuit is unstable; to detect the effect of the fault we wait until after the effect of the fault stabilized. The justification for this methodology is based on two problems which result when this methodology is not used. First, for some cases, only under particular testing conditions is the effect of a fault observable at the outputs when the circuit is in unstable states (due to relative delays in the gates). Secondly, even if it can be determined that a fault will cause a voltage glitch at the outputs, the duration and size of the glitch may be small, making it difficult to detect. Hence, this methodology eases the job of the testing equipment.

3 SAF Excitation-And-Propagation

There are two ways a SAF can effect the behavior of a circuit. The first is to prevent a transition in a TSP, forcing the circuit to stabilize in what was initially considered an unstable state of the TSG. The second is for the SAF to cause an unexpected transition in the TSG. In this section we will discuss the necessary and sufficient conditions which govern both these SAF excitation effects.

In order to more easily discuss the affect of SAFs in a circuit we define two types of sequences of unstable states:

Definition 3.1 *(Transient state path): A* **transient state path** *(TSP) is a path of subsequent unstable transitions starting at a stable state, referred to as a* **source** *state, and ending in a stable state, referred to as a* **sink** *state.*

For example, in Figure 2 we see that the path of states $\{[0000] \longrightarrow [1000] \longrightarrow [1100] \longrightarrow [1101]\}$ constitutes a TSP where state $[0000]$ is the source state and state $[1101]$ is the sink state. We define a **SAF-TSP** as a TSP induced by a SAF. Hence, in this section we will characterize the source state of a SAF-TSP.

We decompose the effects of a SAF by considering the possible locations of the SAF: On a primary input, primary output, within a CLB, and at an input or the output of a C-element. As described earlier, due to the STG liveness property, faults on the primary inputs and outputs are easy to detect and hence will not be discussed further in this paper.

3.1 SAF in a CLB

In this subsection we consider SAFs within a CLB, as part of either the NSB or the OLB. Without loss of generality we assume the two-level logic is in the SOP form. After the testing analysis is complete, however, it is well known that **algebraic factorization** can be used to expand CLBs to multi-level implementations without changing the CLBs testability or modifying the test set.

For a fault within a CLB to propagate to the output of the CLB, the CLB and the current state of the circuit must satisfy one of the two following necessary conditions.

- In state q, to falsely propagate a 0 to the output of the CLB, the state must be an essential state in the cover of the CLB, where the associated 1-set in the cover corresponds to the AND gate whose output has a stuck-at-0 fault.

the circuit into an invalid state, when analyzing the testability of a circuit, it is sometimes necessary to refer to the state graph which depicts the behavior of the circuit in the invalid states, referred to as the **invalid state graph.**

We use the standard **sum of products (SOP)** or **products of sum (POS)** form to implement a Boolean function. A **literal** is a variable (signal) or its negation (v or $\overline{v}$). One possible combination of values of the literals is referred to as a **vertex.** A **1-set** in a Karnaugh map is a set of vertices such that a) whenever all literals are equal to 1, the output is equal to one, and b) if any literal is removed, condition (a) no longer holds [7]. A **cover** of a function is a collection of 1-sets which contain all vertices for which the output is 1 and contain no vertices for which the output is 0. In the context of this paper, a vertex is interpreted by a state. An **essential state** is a state whose corresponding vertex in the cover of the output function exists in only one 1-set. In an SOP implementation of a Boolean function, each AND gate corresponds to a 1-set in the cover of the function, and an OR gate implements the sum of 1-sets. It should be noted that an AND gate may have inverted inputs. For testability purposes we consider the combination of an AND gate and input inverters as one gate.

When a node is fixed at a logical low or logical high, we say that the node is **stuck-at-1 (SA-1)** or **stuck-at-0 (SA-0).** If the fault occurs in isolation the fault is called a single-stuck-at-fault (SSAF). That is, we assume that if a wire is stuck at some value, then all terminals of that wire are also stuck at the same value. In a combinational network an input vector is said to **excite** fault f if the output of the combinational network is different with and without the fault when the input is applied to the network. When this happens, the effect of the fault is said to **propagate** to the output of the network.

We define an asynchronous control circuit to be fully SSAF testable if there exists some sequence of inputs which will excite the fault and cause its outputs to observably behave in a manner other than expected.

2.1 Asynchronous Control Circuits

Asynchronous control circuits manage communication of data between concurrent communicating processes. In this paper, we use the notation of signal transitions for specifying circuit behavior, which is described by a set of partial orderings of signal transitions in a signal transition graph (STG) [3]. As an example we examine a simple handshake circuit which handles the communication between two pipeline stages described in [8]. The block diagram and its STG specification of the circuit is shown in Figure 1.

From the STG a reduced state graph can be determined [3]. The reduced state graph includes only bitvectors of input and output signals and the state transitions allowed to occur. For a circuit to be live, every cycle in the reduced state graph must contain both the rising and falling transitions of all input and output signals [3]. Hence, if a SAF fixes an input or output node, the fault is detectable. As described in [1] such a fault will cause the circuit to halt under normal operating conditions.

From the reduced state graph Boolean equations are formed and a circuit implementation is derived. The reduced state graph and a hazard-free implementation of the pipeline handshake is shown in Figure 2. Notice that the diagram includes both the circuit and the model of the environment in what is jointly called a **complete circuit.** The total state graph (TSG) is an expanded version of

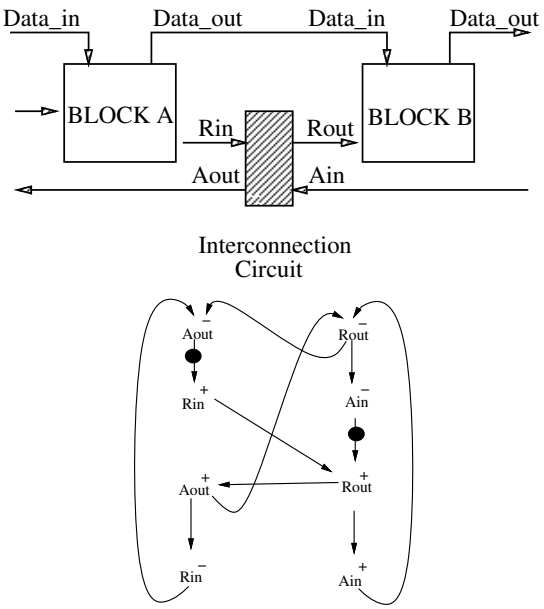

Figure 1: A pipeline handshaking circuit and its corresponding STG specification.

the reduced state graph where the state bitvector is appended with internal memory nodes not included in the reduced state graph (RSG). In the example above, the TSG and the RSG are identical because all memory elements' outputs are external output signals.

We define a signal transition to be **enabled** if the signal will change value given the set of inputs to the gate of which it is the output. When the signal does change value the transition is said to have **fired.** We define a **hazard-free** circuit as one whose inputs and outputs behavior satisfies the specification of its corresponding STG, and where no memory element can be enabled and disabled without the changing of one or more inputs. Hence, the outputs of the memory elements in a hazard-free circuit can not exhibit a voltage spike, sometimes called a **glitch.**

Asynchronous control circuits use C-elements as memory elements. The Boolean function of a two-input C-element can be described as $C' = AB + AC + BC$, where A and B are the two inputs to the C-element, C is the current state of the C-element and C' is the next state of the C-element. In this paper, we restrict each input to a C-element to be controlled by a combinational logic block (CLB) decoupled from other inputs to C-elements. Each primary output signal is controlled by a CLB decoupled from other outputs as well. The CLBs which feed the C-elements are collectively referred to as the **next state block (NSB).** The CLBs which feed the output signals are collectively referred to as the **output logic block (OLB)**. Feedback signals can be outputs of C-elements or primary output signals. Furthermore, we assume that there be no direct feedback from the output of a C-element to the CLBs which feed into it.

2.2 Testing Environment

In this paper we assume that testing is conducted slow enough so that the circuit does not leave a stable state until all other nodes in the circuit have stabilized, i.e., an input is not changed until the effect of the previous input has stabilized. This condition on circuit timing can be easily calculated, and thus justified, as "timed"

Appeared in 28th ACM/IEEE Design Automation Conference Proceedings 1991

Testability of Asynchronous Timed Control Circuits with Delay Assumptions[1]

Peter A. Beerel and Teresa H.-Y. Meng
Computer Systems Laboratory, Stanford University

Abstract

This paper addresses the testability of "timed" asynchronous control circuits built of standard logic cells, in which each gate's rise and fall time is associated with a pair of minimum and maximum values. The circuit that we analyze is assumed to be hazard-free, given that the gate delay assumptions are met in its implementation. We first give sufficient and necessary conditions under which the affects of single stuck-at-faults (SAFs) can be characterized. Then we provide sufficient conditions which ensure that these faults can be detected without access to the memory elements. Finally we present an automated tool which analyzes the testability of "timed" asynchronous circuits based on the derived conditions.

1 Introduction

Asynchronous circuit design has recently seen an uprising of interest in the literature. As VLSI circuits become smaller and smaller, problems with synchronous designs have made people reevaluate the advantages of a clock-free design. Asynchronous circuits avoid problems that synchronous designs have such as clock distribution and skew, and enjoy the advantages of modularity and automatic power-down for low-power considerations. With the outcoming of automated design techniques, asynchronous designs have the promise of being easier to design with respectable speeds [8], but testing such designs has been considered a very difficult task.

In related papers it has been shown that when a hazard-free asynchronous circuit is speed-independent, that is when all delay bounds are arbitrary but wire delay is negligible the circuit must be self-diagnostic for most SAFs (single and multiple) [1, 6]. In addition, a hierarchical methodology to generate test vectors to test for SAFs in speed-independent circuits is presented in [5]. Because of performance considerations, the designer may not want to implement a speed-independent circuit but prefer the use of estimated minimum and maximum gate delays to optimize the design of asynchronous circuits. That is, rather than requiring that the circuit be hazard-free given arbitrary gate delays, we can design the circuit such that if some practical gate delay assumptions are met, the circuit is guaranteed to work correctly (the hazards remaining in the design can never cause a critical race). This compromise to a speed-dependent design, referred to "timed" circuit design, can reduce circuit complexity and increase circuit speed. Most asynchronous circuits used in standard interfaces are in fact "timed" rather than speed-independent. Hence the design for testability with relaxed hazard-free conditions is essential for designing asynchronous circuits of higher speeds. For these "timed" circuits, it is not assured that the circuit will be self-diagnostic and the methodology presented in [5] for the generation of test vectors and the determination of testability cannot be used, although the actual testing procedure in [5] is similar to that advocated in this paper.

In section 2 we discuss the asynchronous control circuit structure and present our delay assumptions. In section 3 we characterize all possible effects of SAFs on "timed" asynchronous circuits. In section 4 we give sufficient conditions for SSAF testable asynchronous control circuits under given delay assumptions. Then, in section 5, we describe a CAD tool which determines SSAF testability using the conditions derived in previous sections and present some example circuits. Finally, in section 6 we give our conclusions.

2 Hazard-Free Asynchronous Circuits

We describe the state of an asynchronous circuit as a bit vector containing one bit for each signal in the circuit that is of interest. The **total state** of a circuit contains one bit for each input, output and the memory element of the circuit. The total states can be differentiated in 3 ways. **Invalid states** are those states which the circuit, under normal operation, cannot enter; **valid states** are those which, under normal operation, the circuit can enter. **Stable states** are those in which the circuit will remain stable when the input does not change; **unstable states**, also referred to as **transient states**, are those states which the circuit will not remain in when the primary input does not change. **Reachable states** are those states which can be controllably entered; **unreachable states** are those in which the circuit cannot be controllably driven to or though. We use **total state graph (TSG)** to graphically depict all the possible state transitions of the circuit. In most cases only the valid states are of interest and hence the state graph shown includes only the valid states. This state graph is referred to as the **valid state graph**. However, because a fault may drive

[1]This research was supported by an ONR Grant no. N00014-89-J-3036 and a research grant from AT&T.