

Testing Micropipelined Asynchronous Circuits

Matthew L. King and Kewal K. Saluja
Department of Electrical and Computer Engineering
University of Wisconsin - Madison
{mking, saluja}@ece.wisc.edu

Abstract

Despite advances in the design of asynchronous circuits, little progress has been made in their testing or design for testability. This paper proposes a new strategy for testing micropipelines, by treating the asynchronous elements, such as the C-element, as atomic state elements for testing purposes. By treating asynchronous elements as finite state machines, tests for these can be generated that verify their correct operation and also detect nearly all testable faults in the circuit. Design for testability methods for a micropipeline are also presented that reduce the amount of hardware added to the design while increasing its overall testability compared to other micropipeline testing methods.

I. Introduction

As the limitations of conventional circuit designs grow ever more apparent, asynchronous circuit design techniques are likely to become more popular. By completely doing away with the clocks, asynchronous designs offer several advantages. The most obvious benefit of doing away with the clock(s) is removing power consuming and complicated clock trees. Asynchronous circuits also offer the promise of higher performance, by allowing computations to proceed at much higher speed than the worst case delay enforced by synchronous circuits. Unfortunately, designing and building asynchronous circuits presents numerous challenges, many of which have yet to be addressed.

Asynchronous circuit designs remain largely unused in high volume commercially produced circuit designs despite the existence of a prototype, Fleetzero, asynchronous chip produced by Sun Microsystems that outperforms its synchronous counterparts [1]. Reasons for the lack of adoption abound. Asynchronous circuits are more difficult to design than their clocked counterparts. Also, producing a product, with high enough yield to be profitable, can be difficult without adequate testing techniques; and such techniques for asynchronous circuits designs are currently lacking.

Any circuit design destined for fabrication must do more than simply function correctly. The circuit, once fabricated, must be tested for correct operation. Many circuit designs do not easily lend themselves to this type of testing. While many techniques for testing combinational and synchronous circuits have been developed, testing methods for asynchronous circuits are almost non-existent.

Even with the current lack of adoption, many still feel that asynchronous circuit designs will soon become commonplace. The clock tree in a high performance CPU currently consumes from 5-30% of the power dissipated by the chip [7][8]. As clock speeds continue to rise, the amount of power consumed by clocks will continue to increase, causing numerous problems for designers. In addition to offering the potential for lower power consumption and higher performance than synchronous circuits, asynchronous circuits are becoming desirable for use even in largely clocked designs. Trends pushing clock speeds ever higher have also created an increase in the number of clock domains in current designs, as designers try and push each component of the design to its maximum speed. Transferring data between clock domains presents a unique challenge however, especially at high clock speeds where clock skew between the different clock domains can impose strict restraints. An asynchronous circuit used as a buffer between clock domains would be under no such constraints, making it an increasingly attractive design alternative.

This paper addresses the test issues related to testing a micropipeline by testing all the components used to implement it. The next section of this paper will provide a more detailed description of the problem. Subsequent sections will examine a method for testing asynchronous circuit elements and improved methods for testing a micropipelined circuit.

II. Background

Until recently a major problem with asynchronous designs was finding a way to organize

a self-timed circuit that would provide for a simple conceptual framework upon which large circuits could be built. Such a framework is provided by Southerland in his paper outlining Micropipelines [2]. As shown in Figure 1, a micropipeline consists of a datapath and the corresponding pipeline latches controlled by a series of interlocking Muller C-elements.

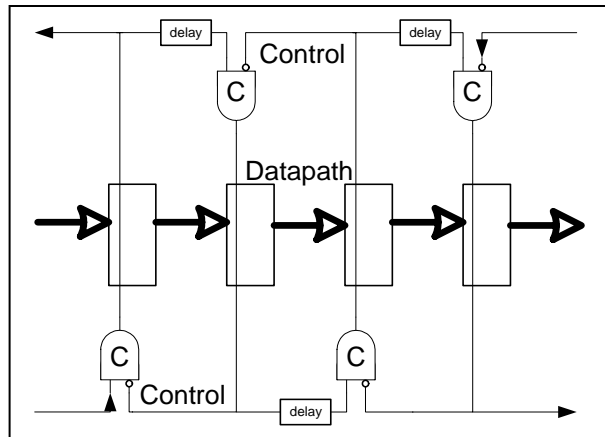


Figure 1: Architecture of a micropipeline

Though the micropipeline framework provides a very simple design concept, it did not provide the same simple conceptual framework for testing that it does for design. A consequence of the micropipeline design is that any stuck-at fault in the C-elements will halt the forward progress of the circuit [3][9]. This fact has led to several strategies for testing the datapath of a micropipelined circuit [3][4][9]. While these strategies might work for the original micropipeline presented in [2], they may not be as effective for testing micropipelines using more sophisticated feedback and control. Furthermore, this strategy will test only stuck-at faults; other faults, such as delay and transition faults, that may also affect correctness or performance will go undetected.

These problems with current micropipeline testing methods suggest that a new approach for testing micropipelines, and self-timed circuits in general, is needed. A major problem with testing asynchronous circuit designs is handling the many feedback paths that occur on both the local and global level. Combinational test generators cannot adequately handle any type of feedback path in the circuit under test. Test generators for sequential synchronous circuits are able to handle feedback paths, albeit in a limited fashion.

While test generation methods for synchronous circuits cannot be directly applied to asynchronous circuits, there are several similarities between the two types of design that make adaptation of synchronous test methods possible. Synchronous circuits rely heavily on latches and/or flip-flops to hold data despite the fact that these latches and flip-flops are asynchronous elements with local feedback paths themselves. One of the keys to testing a synchronous circuit is to ignore the internal structure of these latches and flip-flops, treating them instead as atomic storage (or state) elements. Thus the need to test the local feedback paths internal to the latches or flip-flops is avoided.

Synchronous test generators still must deal with global feedback paths. This is made possible because the convention in synchronous circuits is that only signals that are the output of a latch or flip-flop are used as feedback. This allows the use of methods like time frame expansion [12]. Because all local feedback paths have been removed from the circuit under test and all global feedback paths are the outputs of state elements, a synchronous circuit test generator can successfully produce tests for most clocked circuits.

A cursory glance at the diagram of a micropipeline in Figure 1 shows that it shares several key features with its synchronous counterparts. By replacing asynchronous elements with atomic state elements in the circuit under test, the same conditions that were present in synchronous circuits can be achieved; namely that all local feedback paths are removed and all global feedbacks will be the outputs of state elements. Thus, traditional methods of sequential circuit test generation in which flip-flops and latches are treated as atomic elements in the design can be adapted for asynchronous circuit testing.

Tests capable of finding faults in asynchronous circuit designs are necessary if those designs are to become reality. The key to testing asynchronous designs is the testing of the asynchronous state elements used in those designs. By testing the asynchronous elements used in the micropipeline design (such as the C-element) as black boxes, their correct operation can be verified and known methods can subsequently be applied to test the rest of the circuit.

The next section examines the methods used and the tests generated for two asynchronous elements which are commonly used, the Muller C-element and the Toggle element. The proposed approach is to develop checking sequences for these

elements and use design for testability (DFT) methods to modify the micropipeline such that the desired tests can be applied to all elements while the logic inserted by DFT avoids critical paths as much as possible.

III. Checking Asynchronous Elements

Asynchronous elements with feedback are not fully testable using ATPG methods designed for combinational circuits, as the states of such elements depend on a potentially unknown number of previous inputs. Instead of treating these elements as a series of interconnected gates with feedback, for testing purposes they should be treated as a single Finite State Machine (FSM). By treating the element as a FSM, its entire behavior can be captured in a form for which it is known how to generate tests.

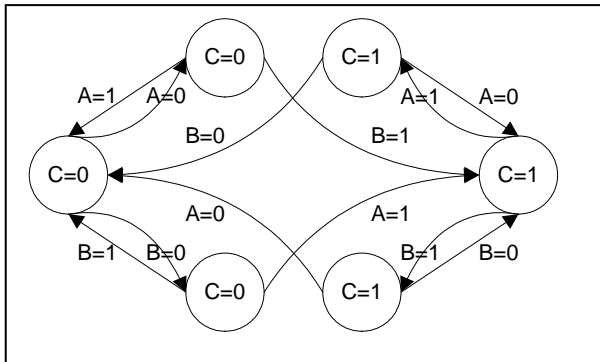


Figure 2: State diagram of a C-element

First, the C-element, the primary source of control in many self-timed circuits such as micropipelines, will be examined. The state diagram for the C-element is shown in Figure 2, and a gate level design is shown in Figure 3. From this diagram it is possible to construct a checking sequence for the FSM represented [5]. By applying this checking sequence to a C-element, the functionality of the C-element can be proven to be correct. By verifying that the C-element responds properly in each state, confidence of its correct functionality can be obtained regardless of its physical implementation.

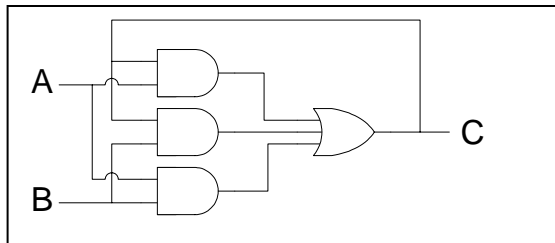


Figure 3: Gate level realization of a C-element

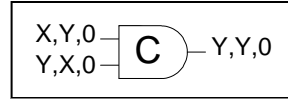


Figure 4: A test for a C-element

Constructing a checking sequence for a generic FSM is detailed in the literature [5][6]. Such methods often produce very long checking sequences however. By analyzing both the FSM and the gate level design of the C-element, it was possible to reduce the length of the checking sequence substantially. The reduced sequence that has been created consists of two parts. Sequence X = 01011010 and sequence Y = 00001111. To test a C-element, X is applied to one input while Y is applied to the other, and vice versa. Finally, 0 must be applied to both inputs. The result is a test pattern of A=X,Y,0 and B=Y,X,0 as shown in Figure 4.

Tests must also be able to detect the faults from an underlying fault model. To verify that the checking sequence developed does this, simulations were run using the Mentor Graphics toolset. The simulations show that the developed test sequence will successfully detect any single-stuck-at fault in the C-element, as well as test all of the robustly testable path delay faults and all of the testable transition faults.

Table 1: Fault Coverage of C and Toggle elements

	C-element	Toggle
Single Stuck-at faults	26	54
Tested	26	48
Potentially Tested	0	6
Transition Faults	26	48
Tested	12	24
Potentially Tested	0	7
Untestable	14	12
Robust Path Delay Faults	8	N/A
Tested	4	N/A
Untestable	4	N/A

The same method can be used to develop a test sequence for a toggle element. The finite state diagram for a toggle element is shown in Figure 5. With fewer states than the C-element, developing a checking sequence for the toggle element was comparatively easy. With seven gates and a gate level design similar to that of a D flip-flop it was not surprising that the checking sequence produced also resembled that of a D flip-flop. The resulting checking sequence for a toggle element was 001100110.

While the checking sequence for a Toggle element was less complex than for a C-element, the

larger number of gates and feedback paths in the design of a Toggle element makes it a much more difficult circuit in terms of fault coverage. Whereas the checking sequence developed from the FSM of a C-element was able to detect every testable fault of the fault models considered, such was not the case for the Toggle element. Still, very good coverage was obtained. The statistics on fault coverage for C-element and a Toggle element are given in Table 1.

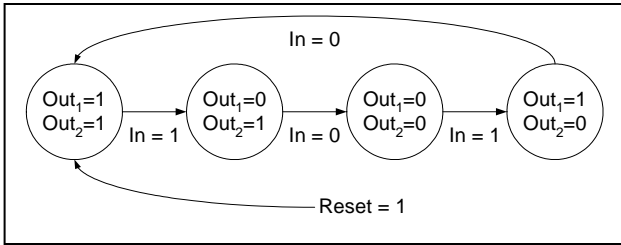


Figure 5: State diagram for a Toggle element

The method presented here has been proven to be capable of testing a wide array of asynchronous elements [14]. There are limits to its applicability though. Circuits, such as the arbiter given in [2], which are composed of multiple interconnected asynchronous elements will not be adequately tested by this method. The lack of effectiveness arises because such circuits can prevent their component elements from reaching all possible states; hence the component elements cannot be fully tested.

Testing a large asynchronous design requires that the above method be applied to more than a single asynchronous element. The interlocking series of C-elements in a micropipeline can be logically arranged in several ways. The next section examines two of those potential structures and how they can be tested.

IV. Checking Interconnected C-elements

Having developed a checking sequence for a single C-element, the next challenge is how to apply this sequence to connected C-elements. The basic C-element produces a transition on the output after transitions have occurred on both of its inputs. If more than two signals need to be synchronized in this fashion, multiple basic C-elements must be used.

Tree of C-Elements: By connecting C-elements in a tree structure, the number of signals whose transitions can be synchronized is greatly increased.

Expanding the sequence produced in section 3 to check a tree of C-elements requires the outputs produced by the application of the two input sequences be examined. As shown in Figure 4, an input of X,Y (applied to inputs A and B respectively) or Y,X will produce an output of Y. To test a tree of C-elements, X and Y sequences need to be generated on the output of any given C-element so that C-elements above it in the tree can be tested. The application of Y,Y produces an output of Y and inputs of X,X produce an output of X. With the ability to produce both X and Y sequences at the output of a C-element, C-elements above the first level of the tree can be tested via application of the sequences X and Y, just as the single C-element was tested. The critical observer will note that applying the same input sequence to two inputs results in multiple simultaneous input changes, which is not defined in the FSM for a C-element. A method for avoiding this problem is presented shortly.

From the results in section 3 and the input-output sequences given above it can be observed that if the sequence Y is applied to either input, the output will be Y. In order to propagate a sequence X to the output of a C-element, it must be applied at both of the inputs.

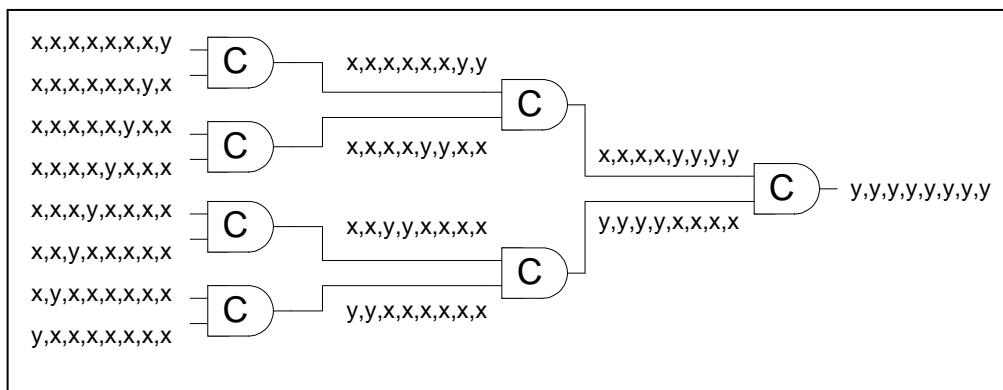


Figure 6: Simple test for a tree of C-elements

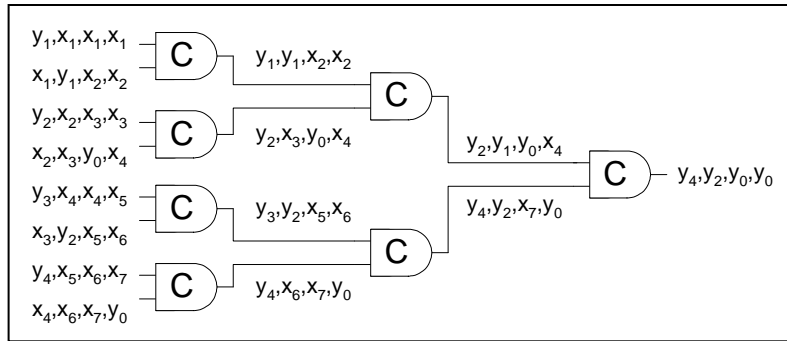


Figure 9: Improved test for a tree of C-elements

the checking of the root element, C_3 . Sequence X will then need to be applied to both inputs to propagate it to the output of C_2 . The same method gives the inputs for C_1 seen in Figure 8. Now that the root element in the tree, C_3 , has been completely tested, the last step is to finish testing the elements in the previous level of the tree. Only one more sequence needs to be applied to the inputs of the previous elements to complete their testing. In the case of Figure 8, the inputs to both C-elements C_1 and C_2 should be $A=Y$ and $B=X$. In this manner both halves of the checking sequence developed in section 3 will be applied to each element in the tree. Applying this method recursively allows the generation of a test sequence for trees of any depth. When expanded to 3 levels, the sequences shown in Figure 9 are produced.

This method is also subject to the same growth in the length of sequence X as the previous method; however, the number of applications of sequences X and Y is greatly reduced, becoming proportional to the depth of the tree.

Result 3: The length of the improved checking sequence to test a C-element tree with N C-elements is $O(N * \log N)$.[‡]

Cascade of C-Elements: C-elements can also be connected in a cascade manner to synchronize multiple signals as shown in Figure 10. Cascades of C-elements can be tested using the same basic ideas as for a tree of C-elements. By applying X and Y to the element closest the output and working back towards the inputs, a checking sequence for the entire cascade can be easily developed.

While there are substantial optimizations possible for testing a tree of C-elements, cascades of

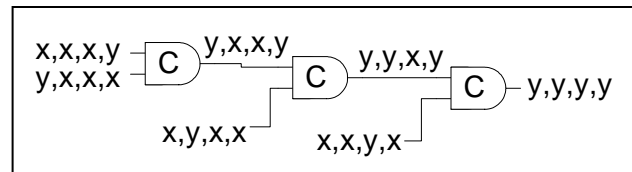


Figure 10: Test for a cascade of C-elements

C-elements prove more difficult. Because both modes of growth for the checking sequence increase proportional to the depth of the tree or cascade, the test for a cascade becomes much longer than for a tree with the same number of elements. Currently, there is no efficient way to reduce the length of the checking sequence for a cascade of C-elements.

Result 4: The length of the checking sequence to test a cascade of N C-elements is $O(N^2)$.

V. Micropipelines

The structure for micropipelines outlined in [2] makes heavy use of C-elements to control the progress of the circuit. However the existing literature does not propose any way of testing these interlocking C-elements. It has been shown that any stuck-at fault on the input or output of a C-element will cause there to be no more than one transition on the output of that C-element [3]. Because of this, a micropipeline with any stuck-at faults in the C-elements will not make forward progress. Unfortunately, this is not an effective method for testing a micropipeline. Faults other than stuck-at faults which can effect both correct operation and performance will not be detected.

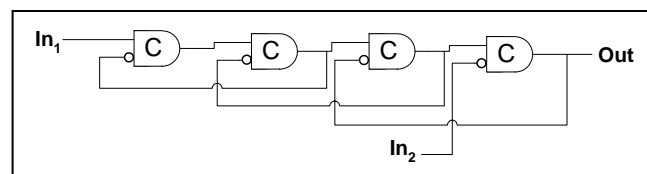


Figure 11: C-elements from a micropipeline

[‡] There is also growth in the length of sequence Y from applying it to multiple inputs simultaneously. This is small in comparison to the other modes of growth however. The actual length of the improved checking sequence is $O(N * \log N) + \log N$.

With the datapath components pulled out, the micropipeline control seen in Figure 1 reduces to an interlocked cascade of C-elements as shown in Figure 11. Since a method for testing a cascade of C-elements was demonstrated above, testing the control structure of the basic micropipeline in [2] is now simply an exercise in how to apply those test patterns to the interlocked cascade.

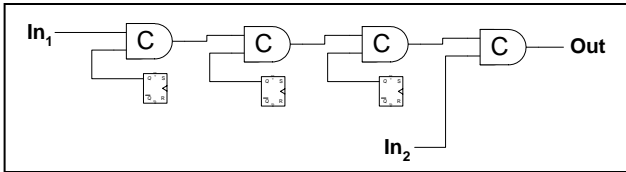


Figure 12: C-element cascade after scan latch insertion

By inserting scan latches into the feedback paths, it would be possible to obtain a straightforward cascade of C-elements, with the inputs fed by scan latches instead of feedback as shown in Figure 12. Note that in Figure 12 the inverters on the feedback input to each C-element have been absorbed into the scan latches for testing purposes without loss of generality. In addition, because of the layout of the micropipeline in [2], one of the inputs to each C-element is guaranteed to be off any critical timing paths (since the artificially inserted delays can be adjusted). This allows for the scan latches to be placed off the critical path, reducing any potential performance impact [13]. The downside to this arrangement of scan latches is that testing of cascades may prove to be prohibitive for larger micropipelines. Another difficulty of this structure is the number of scan chains required to effectively test it. As different input patterns are applied to every C-element when testing a cascade, there is no opportunity to combine any of the scan latches into a single scan chain. Because of this, other potential configurations for the scan latch placement are examined in this paper.

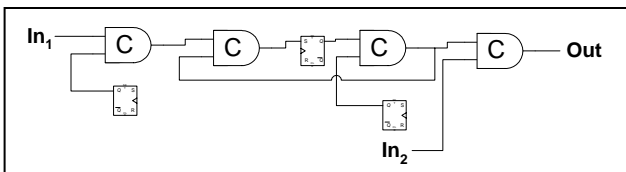


Figure 13: An alternate and efficient scan latch insertion method

A possible alternative is to alternate inserting scan latches into the feedback and feed forward paths as shown in Figure 13. This leads to every other C-element being directly controllable at both

inputs. Rearranging the elements in Figure 13 results in the structure shown in Figure 14.

By using the same strategy that was used to develop a test for trees of C-elements, a test for this structure can be constructed. The primary advantage to this choice for scan latch insertion is that it limits the depth of the tree structure. This greatly reduces the length of the test sequence that needs to be generated. Another advantage of this structure is that only 3 different input sequences are necessary to test the entirety of the structure, regardless of the length of the micropipeline. For example, the Y, X, X pattern appears on one input of every C-element in Figure 15. If the scan latches driving these inputs are connected into a single serial scan chain and input sequences X_1 is applied to the first scan latch after the scan chain is reset, sequences X_2, X_3 and X_4 will be generated on subsequent latches in the serial scan chain. The same can be done for the two other input patterns. This allows the number of scan chains to completely test this structure to be limited to three scan chains.

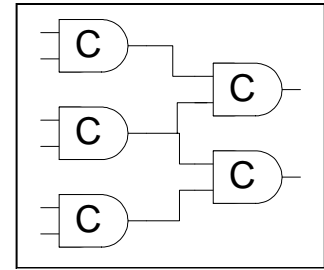


Figure 14: Re-arranged C-elements after efficient scan latch insertion

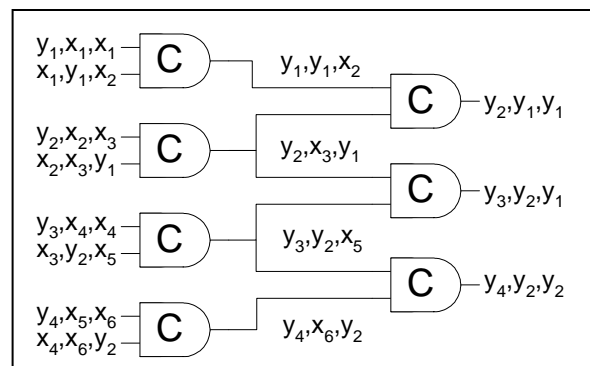


Figure 15: A test for a micropipeline's C-elements after efficient scan latch insertion

Adding scan latches to test the C-elements in the control structure of the micropipeline has other benefits as well. Looking at the micropipeline diagram with the scan latches inserted shown in Figure 16, it can be seen that every other C-element is directly controllable through the scan latches. Because the datapath latches are controlled by these C-elements, the latches of every other pipeline stage

can be directly controlled. And because the remaining C-elements are controlled by the outputs of the neighboring C-elements (both of which will be directly controllable) control over the remaining C-elements can be easily asserted (and subsequently the pipeline latches they control). This allows control over the latching of every pipeline latch in the datapath. Using this ability would make it possible to forgo inserting scan latches into the datapath itself; instead, the event-triggered latches already there could be used to perform tests on the datapath by controlling their latching through the already present control mechanism.

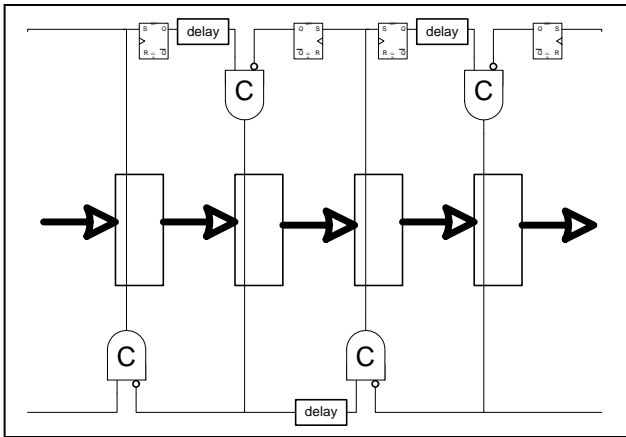


Figure 16: Micropipeline with scan latches inserted

VI. Generalized Micropipeline Testing

The micropipeline proposed in [2] used only C-elements to control the progress of the circuit. Other arrangements for micropipelines have been developed that insert additional elements into the control path. Figure 17 shows the generic micropipeline architecture. The latch control circuits developed in [10] and [11] use multiple asynchronous elements in their design to control the handshaking and latching signals.

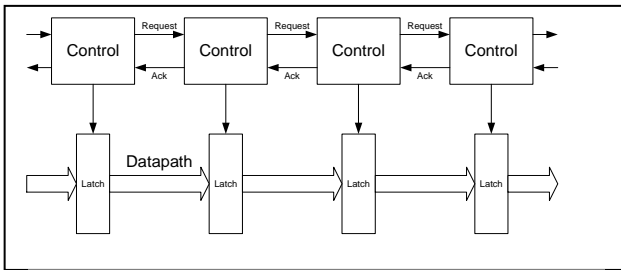


Figure 17: A generic micropipeline architecture

The control circuits of the generic micropipeline can be tested in the same manner as the previous example. By inserting scan latches as shown in Figure 18, direct control over the inputs to every other pipeline stage (labeled 'M' in Figure 18) is obtained. This allows direct application of the appropriate test sequence for the control circuit to those pipeline stages. The remaining pipeline stages can be tested by generating the appropriate patterns on the outputs of the stages labeled 'M₁' and 'M₂' by providing the necessary inputs through the scan latches. The process is explained in greater detail in [14]. As in the previous section, this scan latch insertion requires only one latch per pipeline stage. It also limits the number of scan chains necessary to apply the tests.

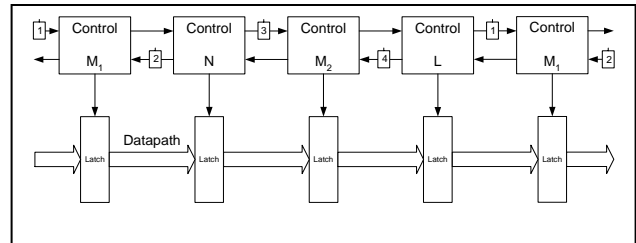


Figure 18: Scan latch insertion for a generic micropipeline

Also, this testing strategy allows for the datapath of the micropipelined circuit to be tested without adding any more hardware. The datapath of a micropipeline with the control circuits removed, shown in Figure 19, is not substantially different from the datapath of a synchronous pipelined circuit. Because of this, the same testing methods can be applied directly to the micropipelined datapath. Since the scan latches inserted into the micropipeline to allow testing of the control circuits for each stage also allow for the latching signals to be controlled, the latches in the micropipelined datapath can be activated as necessary. Applying the BALLAST method [16] to this structure is straightforward, the test pattern is applied to the primary inputs, and each latch in the pipeline is activated one time

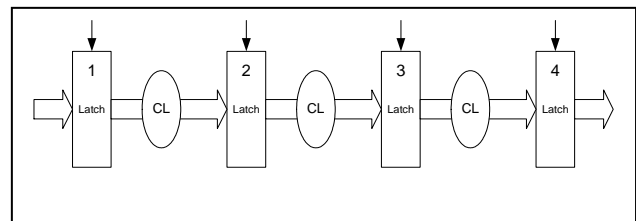


Figure 19: The datapath of a micropipeline

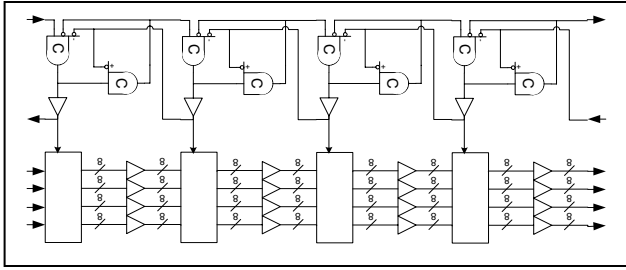


Figure 20: A micropipelined queue

starting with the latch closest to the input (i.e.: latch 1 is activated, then latch 2, then 3 and finally latch 4). The result is the same as if all the latches were controlled by a single signal and the test was applied in four steps [14].

In order to get a better understanding of the cost and effectiveness of the proposed method, a sample circuit will be analyzed. Figure 20 shows a micropipelined queue, using one of the four-phase control circuits presented in [10]. Figure 21 shows the same circuit with scan latches inserted.

The first step in testing this circuit (see Figure 21) is to apply the test sequence from [14] directly to the stages labeled 'M' by using the inserted scan latches. To test the stage labeled 'N,' the Rout output of the stage M_1 and the Ain output of stage M_2 must provide the test sequence to stage N. By applying a value of 0 on the scan latch labeled '1' and a value of 1 on the scan latch labeled '2' the Rout output of stage M_1 will be forced to 0. Similarly, applying 1,0 to scan latches 1 and 2 respectively will force the Rout output of stage M_1 to 1. Controlling the Ain output of stage M_2 is done in an analogous manner, allowing for the full test sequence to be applied to stage N. Testing for stage L is accomplished in the same way.

This allows for the control circuit of each stage to be tested. For this circuit, 100% fault coverage was obtained. Though the fault coverage obtained will vary depending on the control circuit used, good fault coverage was obtained for a variety of circuits examined in [14].

The datapath of Figure 21 can be tested by adapting the BALLAST method as discussed previously. In this case, the fault coverage was 100%, and the BALLAST method is known to achieve high fault coverage on a variety of pipelined circuits [16].

Compared to other methods for testing micropipelined circuits, the proposed method offers high fault coverage with reduced area overhead. Full scan methods such as the one proposed in [15] report overheads of 25-50%, without fully testing the asynchronous elements of the circuit. The overhead of the method proposed here is just one scan latch per pipeline stage, or 4% for the circuit shown in Figure 21. In general, the overhead of this testing method will vary between 2-5% for a 32 bit queue, depending on the type of control circuit and pipeline latches being used.

VII. Conclusion

As clock speeds increase and power requirements become ever more stringent, new methods will need to be utilized to continue the increase in performance that has been experienced by integrated circuits over the past several decades. Asynchronous circuit design techniques can potentially meet both of those constraints. Before asynchronous circuits become viable in commercial products however, testing methods must be developed.

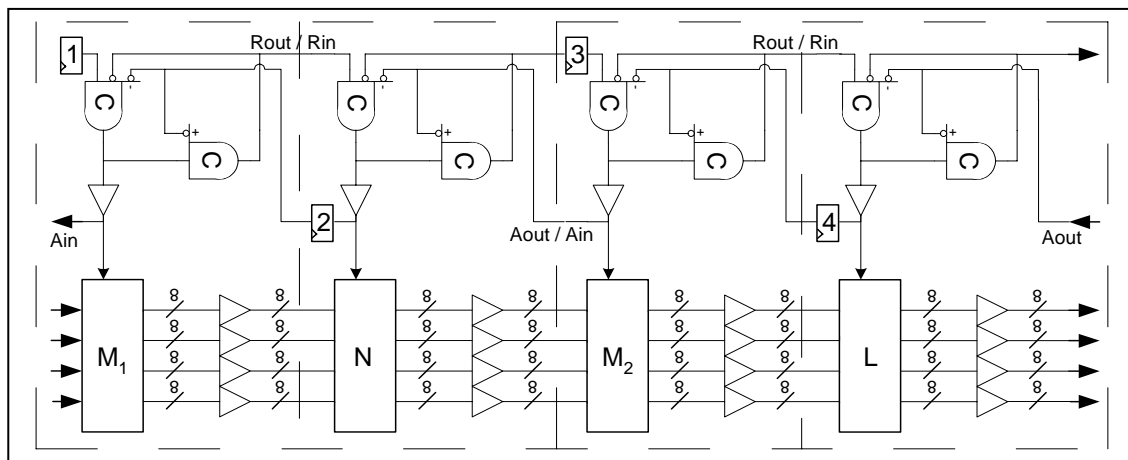


Figure 21: A micropipelined queue with scan latches inserted

This paper proposed a new method for testing micropipelined circuits by adapting to asynchronous circuits the same strategies that allow for the testing of clocked circuits. Treating locally asynchronous elements in the design as atomic finite state machines allows test to be generated that detect most of the faults in that element. Restraining global feedback to be the output of these state elements creates a system organized in a similar fashion to synchronous circuits and allows tests to be generated.

The checking sequences developed were shown to be very good at detecting faults in the asynchronous elements studied. Furthermore, these test sequences were effectively adapted to test a micropipeline. Examining the structure of the micropipeline led to several different DFT approaches that permit more thorough testing of a micropipeline than previous methods.

References

- [1] W. S. Coates, J. K. Lexau, I. W. Jones, S. M. Fairbanks and I. E. Sutherland. FLEETzero: an Asynchronous Switching Experiment. *Seventh International Symposium on Asynchronous Circuits and Systems*, pages 173–182, March 2001.
- [2] I. E. Sutherland. Micropipelines. *Communications of the ACM*, pages 720-738, June 1989.
- [3] A. Khoche and E. Brunvand. Testing Micropipelines. *Proceedings of the International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pages 239–246, Nov. 1994.
- [4] O. A. Petlin and S. B. Furber. Scan Testing of Micropipelines. *Proceedings of the 13th IEEE VLSI Test Symposium*, pages 296–301, May 1995.
- [5] F. C. Hennie. Fault Detecting Experiments for Sequential Circuits. *Proceedings of the Fifth Annual Symposium on Switching Theory and Logical Design*, pages 95-110, Nov. 1964.
- [6] C. R. Kime. An Organization for Checking Experiments on Sequential Circuits. *IEEE Transactions on Electronic Computers*, pages 113–115, 1966.
- [7] M. K. Gowan, L. L. Biro and D. B. Jackson. Power Considerations in the Design of the Alpha 21264 Microprocessor. *Proceedings of the 35th Annual Conference on Design Automation*, pages 726-731, 1998.
- [8] S. Manne, D. Grunwald and A. Klauser. Pipeline gating: Speculation Control for Low Power. *International Symposium on Computer Architecture*, pages 132-141, 1998.
- [9] S. Pagey, S. D. Sherlekar and G. Venkatesh. Issues in Fault Modeling and Testing of Micropipelines. *Proceedings of the First Asian Test Symposium*, pages 107–111, Nov. 1992.
- [10] S. B. Furber and P. Day. Four-Phase Micropipeline Latch Control Circuits. *IEEE Transactions on VLSI Systems*, pages 247-253, June 1996.
- [11] G. S. Taylor and G. M. Blair. Reduced Complexity Two-Phase Micropipeline Latch Controller. *IEEE Journal of Solid-State Circuits*, pages 1590-1593, Oct. 1998.
- [12] Y. Kim and K. K. Saluja. Sequential Test Generation: Past, Present and Future. *Integration - The VLSI Journal*, pages 41-54, Dec. 1998.
- [13] M. L. Bushnell and V. D. Agrawal. *Essentials of Electronic Testing*. Kluwer Academic Publishers, 2000.
- [14] M. L. King. *Testing and Design for Testability of Asynchronous Circuits*. MS Thesis, Department of Elect. and Comp. Engr., University of Wisconsin - Madison, May 2004.
- [15] F. Beest et al. Automatic Scan Insertion and Test Generation for Asynchronous Circuits. *Proceedings of the International Test Conference*, pages 804-813, 2002.
- [16] R. Gupta, R. Gupta and M. A. Breuer. The BALLAST Methodology for Structured Partial Scan Design. *IEEE Transactions on Computers*, pages 538-544, April 1990.