in Figure 2 are detected during test and scan mode.

7: Testing delay faults

There is another class of faults in micropipelines which can be detected using the proposed scan test technique. These are delay faults in combinational circuits between the stage registers of the micropipeline. The output data of each combinational logic block is latched after a certain delay when the data has arrived at its inputs. A delay fault in this combinational block will extend path delays. In the presence of such a fault the bundled data interface of the correspondent micropipeline stage will be violated, i.e. the outputs of the combinational logic will be latched before the output signals in the bundle are stable.

The algorithm used to detect delay faults in the processing logic of the micropipeline is similar to that exploited in delay testing of synchronous circuits which has been adapted by Khoche and Brunvand [12]. Basically, the pair of test patterns (p_1 and p_2) must be applied to the inputs of the combinational circuit to detect its path delay faults. According to this test approach three stage registers (R_{i-1}, R_i and R_{i+1}) are used to detect delay faults in the combinational logic F_i. The tests p_3 and p_1 are stored in the registers R_{i-1} and R_i respectively. The results of the test are saved in the register R_{i+1}. When the test patterns are loaded into the stage registers the combinational circuit is settled (test p_1). The delay fault is tested by applying a request signal to the input *Rin* of the micropipeline set in normal operation mode. This causes the application of the test p_2 to the inputs of the logic F_i ($p_2 = F_{i-1}(p_3)$). A data path of the circuit under test is activated. If there is a delay fault in this path it will cause a delayed response by the combinational circuit whereas the responses are latched after a fixed time determined by the corresponding delay.

8: Conclusions

The scan test technique presented in this paper supports testing for stuck-at and delay faults in micropipelines. The internal inputs and outputs of the processing logic blocks are fully controllable and observable through the scan path. The test patterns are scanned into the registers and the test results are shifted out from the register latches, united into one shift register. The scan path of the testable micropipeline is controlled by the STCL block. Two implementations of the STCL blocks which follow two different communication protocols have been presented. The universal structures of the STCL blocks allow them to be adapted for arranging either a global asynchronous shifting of the test data between different parts of the chip or a local scan path within a particular block.

The proposed testable micropipeline structure greatly simplifies the testing of micropipelines reducing the test complexity to that of the processing logic. The overall overhead can be estimated only for a particular case since it depends on the complexity of the processing logic.

Acknowledgements

The authors would like to express their gratitude to the members of the AMULET research group for creating the stimulating environment for this work.

References

[1] C. L. Seitz, “System timing,” Chapter 7, in *Introduction to VLSI Systems*, C. Mead and L. Conway, Addison-Wesley, 1980.

[2] G. Gopalakrishnan, P. Jain, “Some recent asynchronous system design methodologies,” *Technical Report UU-CS-TR-90-016*, University of Utah, Oct. 1990.

[3] L. Lavagno, A. Sangiovanni-Vincentelli, “Algorithms for synthesis and testing of asynchronous circuits,” Kluwer Academic Publishers, 1993.

[4] S. B. Furber, P. Day, J. D. Garside, N. C. Paver, J. V. Woods, “AMULET1: A micropipelined ARM,” *Proc. IEEE Comput. Conf.*, March 1994.

[5] G. Russell, I. L. Sayers, “Advanced simulation and test methodologies for VLSI design,” Van Nostrand Reinhold (International), 1989.

[6] Chin-Long Wey, Ming-Der Shieh, D. Fisher, “ASCLScan: a scan design for asynchronous sequential circuits,” in *Proceedings of the IEEE Int. Conf. on Computer-Aided Design*, pp. 159-162, 1993.

[7] P. Thorel, J. L. Rainard, A. Botta, A. Chemarin, J. Majos, “Implementing boundary-scan and pseudo-random BIST in an asynchronous transfer mode switch,” *Int. Test Conf.*, 1991, pp. 131-139.

[8] M. Roncken, “Partial scan test for asynchronous circuits illustrated on a DCC error corrector,” in *Proc. Int. Symposium on Advanced Research in Asynchronous Circuits and Systems* (*Async94*), Nov. 1994.

[9] I. E. Sutherland, “Micropipelines,” *Communications of the ACM*, Vol. 32, no. 6, pp. 720-738, June 1989.

[10] P. Day and J. Viv. Woods, “Investigation into Micropipeline Latch Design Styles,” *to be published in IEEE Trans. VLSI circuits*, June 1995.

[11] S. Pagey, G. Venkatesh, S. Sherlekar, “Issues in fault modelling and testing of micropipelines,” *First Asian Test Symposium*, Hiroshima, Japan, Nov. 1992.

[12] A. Khoche, E. Brunvand, “Testing micropipelines,” in *Proc. Int. Symposium on Advanced Research in Asynchronous Circuits and Systems* (*Async94*), Utah, Nov. 1994.

[13] O. Petlin, S. Furber, “Scan testing of asynchronous sequential circuits,” in *Proc. 5th Great Lakes Symposium on VLSI*, New York, March 1995.

[14] P. A. Beerel, T. H.-Y. Meng, “Semi-modularity and self-diagnostic asynchronous control circuits,” *Proceedings of the Conf. on Advanced Research in VLSI / editor Carlo H. Sequin*, MIT Press, Santa Cruz, March 1991, pp. 103-117.

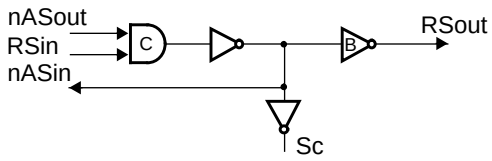

Figure 6: Scan test control logic for a 4-phase micropipeline

goes high and a rising event is generated on the request output *RSout*. If *nASout=RSin*=0 the C-element is set to zero. Thus, the signal *Sc* is reset to zero, rising and falling events are produced on the control lines *nASin* and *RSout* respectively. The delays of the control signals in the STCL block are presented in Table III. The results show that the use of the 4-phase STCL block improves the performance of the shift operation.

TABLE III:
4-phase scan test control delays

Path	Delay
RSin↑ to *RSout*↑	5.6nS
RSin↑ to *nASin*↓	3.1nS
nASout↓ to *nASin*↑	3.1nS
nASout↑ to *RSout*↑	5.6nS
Cycle time	17.4nS

6: Test strategy

The strategy we propose for testing stuck-at faults in micropipelines is very similar to that used in scan testing synchronous circuits. If *Sc*=0 the micropipeline shown in Figure 4 can perform in normal operation mode (*Tst*=0) or in test mode (*Tst*=1). In scan mode (*Tst*=0, *De*=0), the test patterns are loaded into the stage registers which are configured as a united scan register. The scan path is created by connecting the inputs *Sin* in series to the outputs *Sout* of all the stage registers. Clock signals *Sc* for controlling the shift operation are generated internally by the STCL. When the test patterns are loaded into the latches the micropipeline is set to test mode (*Tst*=1). A request signal is produced on the line *Rin* of the micropipeline. The responses from each processing block are stored in the registers. When *Tst*=0 the contents of the latches are shifted to the output *Sout* of the last stage register. The test results are compared with known good ones. Whilst shifting out the test results to the output *Sout* a new test pattern is loaded from the input *Sin*. The test procedure is repeated. Thus, the complexity of testing the micropipeline is reduced to the testing of its processing logic which comprises mostly combinational circuits.

6.1: Testing for faults in the STCL

The STCL unit of the testable micropipeline is an additional control block which is not used in normal operation mode. Nevertheless, it must be fault free as it controls the scan path of the micropipeline. A stuck-at fault on any of the lines in the STCL block prevents the generation of the control signals on its outputs. This is because the STCL is a fully delay-insensitive asynchronous circuit where every control signal handshakes with others. Such circuits are fully testable for stuck-at faults [14].

6.2: Testing for faults in the control logic

As was mentioned earlier, stuck-at faults on the control lines of the micropipeline can be detected easily since they cause the micropipeline to halt. This happens because a micropipeline is an event-driven asynchronous circuit [9]. Such stuck-at faults can be identified either in normal operation mode or during the test.

6.3: Testing for faults in the processing logic

It is assumed that all the processing blocks between the stages of the micropipeline are combinational circuits. The internal inputs of each combinational circuit are controllable and its outputs are observable through the scan path of the micropipeline. Tests for detecting stuck-at faults in all the processing blocks can be derived using well known test generation algorithms such as the D-algorithm, PODEM, FAN and others [5].

6.4: Testing for faults in the latches

Two types of stuck-at faults are considered for the register latches: stuck-at-capture and stuck-at-pass faults.

Stuck-at-capture (stuck-at-pass) faults of the scan latch (see Figure 2) can be caused by stuck-at faults on the control lines of the tristate buffers and inverters which disable (enable) them permanently. Most of these faults can be detected by shifting an alternating 0-1 test through the latches united in one scan register. A stuck-at-1 fault on the input *nTst* of the latch L_1 can be identified during test mode when the faulty scan latch and its predecessor are set to different states. In this case the state of the faulty latch L_1 will be changed. Stuck-at-0 and stuck-at-1 faults on the line *De* of L_2 are detected by driving the input *Din* with a different logic value to its current state during test mode and scan mode respectively.

Stuck-at faults on the data lines of the scan latch shown

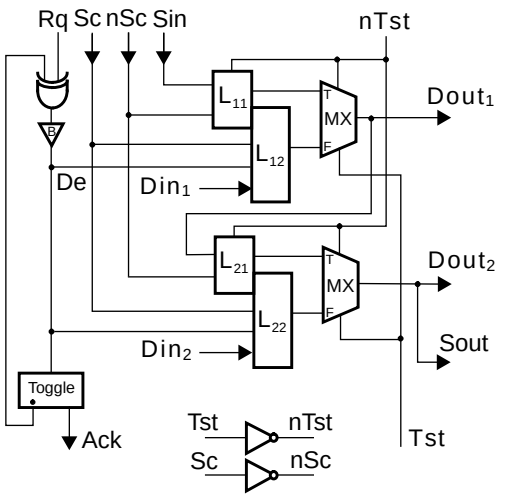

Figure 3: Scan register

Test mode. During the test (*Tst*=1, *Sc*=0) the test vectors are stored in the first latches L_1. The outputs of these latches are connected through the multiplexers to the outputs of the stage register. After receiving a request signal on the line *Rq* the data is stored into the latches L_2 of the register (see Figure 3). The test vectors and the test results are saved in different latches because the data flows through the micropipeline from left to right while the test vectors must be preserved during the test.

5: Scan test control

The testable micropipeline design is shown in Figure 4. It comprises a micropipeline and the scan test control logic (STCL) unit. The stage registers of such a micropipeline are built using scan latches. The STCL block is used to make an asynchronous test interface for the micropipeline.

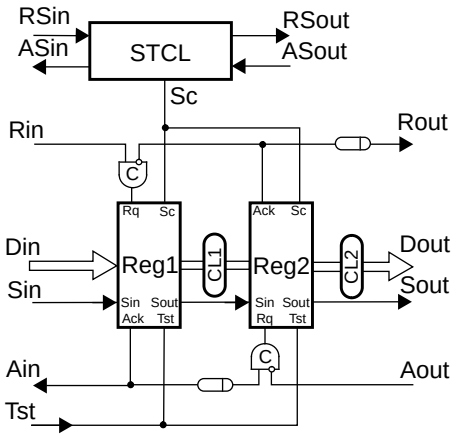

Figure 4: A micropipeline with scan features

It also generates shift clock signals *Sc* for a united shift register. The STCL block can serve either a 2-phase or 4-phase signalling protocol depending on its structure. The STCL can either be a central control block or can be incorporated inside some of the micropipeline registers. Similar STCL units can be used in different parts of the chip to arrange an asynchronous scan test control interface between different asynchronous blocks.

5.1: STCL for 2-phase transition signalling

An example of the STCL block for 2-phase transition signalling is shown in Figure 5. The STCL block generates the control signals in a manner similar to that of the control circuitry of the scan register illustrated in Figure 3. The addition of the C-element ensures the delay insensitivity of the STCL block.

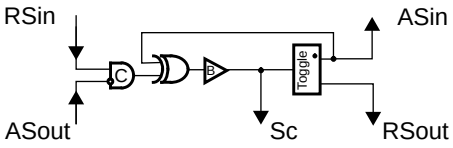

Figure 5: Scan test control logic for a 2-phase micropipeline

Some calculations of the typical delays in the STCL block have been carried out using SPICE analyses and are shown in Table II.

TABLE II:
2-phase scan test control delays

Path	Delay
RSin to *ASin*	6.3nS
RSin to *RSout*	11.8nS
ASout to *C-element primed*	1.2nS
Cycle time	19.3nS

5.2: STCL for 4-phase signalling

The implementation of the STCL block for a 4-phase communication protocol is simpler than that of the STCL block for 2-phase signalling (see Figure 6). Initially, the C-element is set to zero (*nASin*=*nASout*=1). When a rising shift request signal *RSin* arrives, the C-element changes its state to one. As a result, a falling event is produced on the negated acknowledge line *nASin*, the shift clock signal *Sc*

described allows the detection of all the stuck-at faults and bundling constraint violations in micropipelines. However, this scan test technique has been developed only for micropipelines which use a two-phase transition signalling protocol. The scan test interface uses clocks produced by a clock generator which is not always available in asynchronous VLSI designs.

A method to design and test asynchronous sequential circuits based on the micropipeline design style has been reported [13]. The test approach is implemented using specially designed scan latches manipulated by the scan test control logic. In this paper, we extend this test method to the testing of general micropipeline structures.

4: Scan test design

4.1: Scan latch implementation

Figure 2 shows a CMOS implementation of the scan latch structure which contains two latches (L_1 and L_2) and a multiplexer.

In normal operation mode (the test control signal *Tst* is low) the tristate inverter of L_2 is closed since the shift clock signal *Sc* is held at zero (*nSc*=1). When the data enable signal (*De*) is high the input data (*Din*) passes to the output *Dout* and is latched by L_2 when *De* is low.

In scan mode (*Tst*=0, *nTst*=1) the enable signal *De* is low so that the tristate buffer of L_2 is closed. When the clock signal *Sc* is high (*nSc*=0) the scan data from the scan-in input (*Sin*) is latched by the latch L_1 and passes to the tristate inverter of L_2. While *Sc*=1, L_2 is opened and the shift data is sent to the scan-out output (*Sout*) of the scan latch. When *Sc*=0 (*nSc*=1) the scan data bit is latched by L_2 and the latch L_1 is opened. This procedure is similar to that used for storing the data in a master-slave flip-flop.

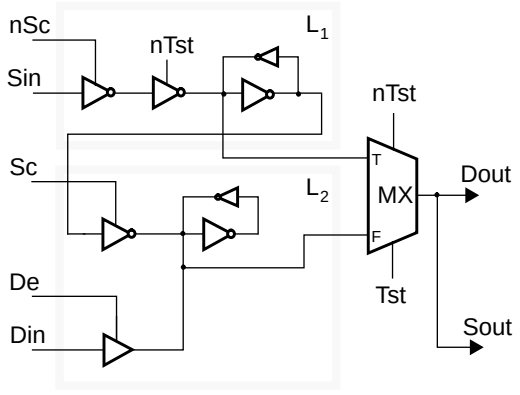

Figure 2: A CMOS implementation of the scan latch

In test mode (*Tst*=1, *nTst*=0, *Sc*=0, *nSc*=1) the scan latch performs as in normal operation mode. The only difference is that the response bit from the combinational circuit is stored in L_2 whereas the test bit is held unchanged in L_1 and stimulates the appropriate input of the processing logic of the next stage. Note that, during scan mode when the last test bit is shifted in the scan latch, the Boolean signal *Tst* (*nTst*) must be set to one (zero) before the signal *Sc* (*nSc*) goes down (high) in order to preserve the state of L_1.

The basic and scan versions of the latch structure have been implemented in CMOS technology on a $1\mu m$ process and simulated using SPICE analyses. The basic latch cell used is similar to a single-phase static CMOS latch which requires 11 transistors [10]. 37 transistors were used for the implementation of the scan latch. As a consequence, the redundancy of the scan latch is 236%. This scan latch requires 12% fewer transistors than the one proposed by Khoche and Brunvand [12]. Table I shows the simulated delays through a single data path of the two latch structures.

TABLE I:
Data path delays for the basic and scan latches

Latch	Path	Delay
Basic	*Din* to *Dout*	3.7nS
Scan	*Din* to *Dout*	5.8nS

4.2: Scan register design

A two-bit scan register design for a testable micropipeline is shown in Figure 3. Compared with the basic register it contains five additional wires: test control (*Tst*), scan-in (*Sin*), scan-out (*Sout*) and shift clock line (*Sc*).

Normal operation mode (*Tst*=0, *Sc*=0). In the initial state the register latches L_2 are closed and the outputs of the toggle element and C-element are set to zero (reset control lines are omitted in Figure 3). When a request signal is received on the input *Rq* the latches L_2 are opened since the data enable signal *De* goes high on the output of the buffer *B*. The data is transmitted from the inputs *Din* to the outputs *Dout* of the register. The toggle element steers the rising event from its output (marked with a dot) to the XOR gate. The output of the XOR gate becomes low. As a result, the latches L_2 are closed (*De*=0) and an acknowledge signal *Ack* is produced by the toggle element on its 'blank' output.

Scan mode. While *Tst*=0 and *De*=0 the register can be used to scan the data into the latches from its input *Sin*. Simultaneously, the scan data comes to the output *Sout* supplying another scan register. The scan procedure is controlled by clock signals applied to the input *Sc*.

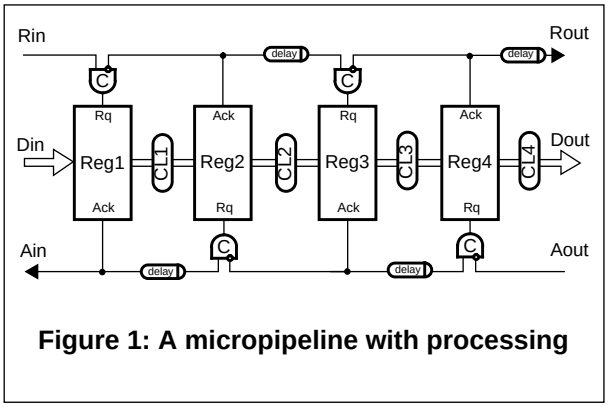

Figure 1: A micropipeline with processing

data comes to the micropipeline from outside and is transferred through the stage registers. If there are no processing blocks (usually combinational circuits) between stages the micropipeline performs as an ordinary first-in first-out (FIFO) buffer. The data processing procedure is controlled by C-elements which are state-holding 'rendezvous' elements performing the AND function for events.

Initially, all the C-elements are set to zero. When the data is ready on the inputs *Din* of the micropipeline the sender issues a request on the line *Rin*. The first C-element transfers a request signal to the first register (*Reg1*) which latches the data and generates an acknowledge (*Ain*) to the sender. A signal *Ack* from *Reg1* is delayed for the time required to complete the data processing performed by the combinational logic (*CL1*). When the data is stable on the outputs of *CL1* the second C-element sends a request to the next stage register (*Reg2*). As a consequence, the data is latched in *Reg2* and the first C-element is primed by an acknowledge signal generated by *Reg2*. New data can be written into *Reg1* and the process of transferring the data through the subsequent stages of the micropipeline is repeated. When the data reaches the last stage of the micropipeline a request (*Rout*) is produced for the receiver which completes the handshaking protocol by sending an acknowledge on the line *Aout*.

Every micropipeline stage works in parallel and sends the data to the neighbour stage only when the data is ready to be processed. The data is latched in registers. There are different ways to implement the control of latching and storing the data in the latches of the micropipeline registers. Basically, the latches are controlled by a pair of control signals such as 'pass' and 'capture' [9]. In the initial state all the register latches can be either transparent or in the capture mode depending on the latch transition controlling protocol. The use of 'normally closed' latches is preferable from the power consumption point of view since no transitions in the data paths can occur unless new data has been latched by the stage register [10].

3: Testing micropipelines

3.1: Faults in micropipelines

There are a few works devoted to fault modelling and fault testing problems in micropipelines [11-13]. Stuck-at faults in the control part, combinational logic blocks and latches of the micropipeline have been considered [11].

Faults in the control part

These are faults on the inputs and outputs of the C-elements and the request and acknowledge lines of the micropipeline. As was shown the micropipeline moves through at most one step and then halts in the presence of a stuck-at fault in its control part. Thus, such stuck-at faults can be identified easily during normal operation mode.

Faults in the processing logic

It was assumed that all the latches of the micropipeline are transparent initially. This allows the processing logic to be treated as a single combinational circuit. To detect any of the single stuck-at faults in such a circuit test vectors can be obtained using any known test generation techniques [5].

Faults in the latches

It was considered that a stuck-at fault inside the latch can put it permanently in capture (stuck-at-capture fault) or pass (stuck-at-pass fault) mode. Any stuck-at fault on the inputs or outputs of the stage register or stuck-at-capture fault of the transition latch is equivalent to the correspondent stuck-at fault in the combinational logic block. To detect a stuck-at-pass fault in the transition latch two test patterns are required.

3.2: Scan testing

An elegant scan test approach has been proposed by Khoche and Brunvand [12]. The micropipeline can work in two modes: normal operation and scan test mode. The micropipeline performs to its specification in normal operation mode. In test mode, all the latches are configured into one shift register where each latch works as an ordinary master-slave flip-flop. The stage registers of the micropipeline are clocked through the control lines where the input *Aout* is used as a clock input. The C-elements pass their negated inputs onto the outputs forming a clocking line for the scan path. As a result, the test patterns are loaded from the scan-in input into all the latches of the micropipeline. Afterwards the micropipeline is returned to normal operation mode in which only one request signal is generated. To observe the contents of the register latches the micropipeline is set to scan test mode. The contents of all the latches are shifted out to the scan-out output. The test technique

Scan Testing of Micropipelines

O. A. Petlin, S. B. Furber

Department of Computer Science, The University,
Oxford Road, Manchester, M13 9PL, U.K.

Abstract

The micropipeline approach to designing asynchronous VLSI circuits has successfully been used in the AMULET1 microprocessor. A method to design and test micropipelines is presented in this paper. The test strategy is based on the scan test technique. It allows the separate testing of all the data processing blocks by scanning the test patterns in and shifting the responses out of the stage registers. The proposed test approach provides for the detection of all single stuck-at and delay faults in the micropipeline. Tests for the combinational processing logic and state holding elements can be derived using standard test generation techniques.

1: Introduction

Asynchronous VLSI designs may have advantages over their synchronous counterparts. The clock skew problem no longer exists in asynchronous circuits since they do not use global clocks. In addition, asynchronous circuits have a potential for lower power consumption [1-3].

An asynchronous version of the ARM6 microprocessor (AMULET1) has been designed by the AMULET research group at the Department of Computer Science in the University of Manchester and fabricated by GEC Plessey Semiconductors Limited. AMULET1 was designed using the micropipeline design approach which offers a good engineering framework for the design of complex asynchronous VLSI circuits [4].

The design process for asynchronous circuits must take into account all hazards and races to ensure a proper signalling interface. From this point of view the testing of circuits without synchronization clocks is complex [3]. As a result, testing asynchronous VLSI designs presents new problems which must be addressed before their commercial potential can be realized. The most widely used fault models chosen to describe fault behaviours of asynchronous circuits are stuck-at and delay (transition) faults [3,5]. The scan test technique has been adapted well to the testing of asynchronous circuits [6-8]. Unfortunately, these results have been obtained for specific asynchronous designs and cannot be used for the testing of micropipelines.

2: Micropipelines

Micropipelines were introduced by Ivan Sutherland in his Turing Award lecture [9]. Micropipelines are asynchronous, event-driven pipelines based on the ‘bundled data’ interface. In micropipelines, the data is treated as a bundle, i.e. when the data produced by the sender is ready (the data outputs are stable) the sender issues a ‘request’ event to the receiver; the receiver acknowledges the receipt of the data by sending an ‘acknowledge’ event. This handshaking mechanism is repeated when further data is produced by the sender.

2.1: Transition signalling

The data transfer protocol in micropipelines is controlled by ‘transition’ signals. There are two types of signalling protocols used in asynchronous circuits: 2-phase and 4-phase signalling protocols. According to the 2-phase transition signalling protocol both rising and falling transition events have the same meaning. When the data is ready to be sent to the receiver the sender produces a rising (or falling) request signal which is acknowledged by a rising (or falling) signal on the acknowledge control line. 4-phase signalling differs from the 2-phase protocol in that both the control signals (request and acknowledge) must be returned to zero, i.e. new data can be transmitted only when both control signals are zero.

2.2: Micropipeline structures

Figure 1 illustrates a micropipeline with four stages. The