violated, i.e. the outputs of the combinational block 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 ASC is similar to that exploited in delay testing of combinational circuits [15]. Basically, the pair of test patterns $<P_1, P_2>$ must be applied to the inputs of the combinational circuit to detect its delay path fault. The pair $<P_1, P_2>$ is made up of $<p_{1p} @ p_{1s}, p_{2p} @ p_{2s}>$, where

p_{1p} and p_{2p} are the test vectors applied to the inputs *PI* of the combinational circuit;

p_{1s} and p_{2s} are the state vectors initially loaded into the state registers (*Reg1* and *Reg2* respectively);

the symbol @ denotes the concatenation of bit vectors.

In test mode, the test pattern p_{1p} is supplied to the inputs *PI* and a request signal is generated on the input *Rin*. The combinational circuit is settled. After receiving an acknowledge event on the input *Aout* the test p_{2s} is copied into *Reg1*. The test vector p_{2p} is applied to the inputs *PI* of the ASC. The test control signal *Tst* is set to zero and a new request event is generated on the input *Rin*. As a result, a data path of the combinational block 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 ASCs built using the micropipeline design approach. The internal inputs and outputs of the combinational logic block are fully controllable and observable through the scan path. The test patterns are scanned into the state registers and the test results are shifted out from the register latches, united into one shift register. The test complexity of such a testable ASC is reduced to that of the combinational block. The scan path of the testable ASC is controlled by the STCL block which generates clock signals. The unique structure of the STCL block allows it 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 overall overhead can be estimated only for a particular case since it depends on the complexity of the combinational circuit.

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 *Proceedings of the Int. Symposium on Advanced Research in Asynchronous Circuits and Systems* (*Async94*), Nov. 3-5, 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 micropipelines,” *to be published in Proc. 13th IEEE VLSI Test Symposium*, New Jersey, Apr. 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.

[15] A. Pramanick, S. Reddy, “On the design of path delay fault testable combinational circuits,” *in Proc. 20th Fault Tolerant Computing Symp.*, pp. 374-381, June 1990.

TABLE II:
Scan test control delays

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

6: Test strategy

The strategy we propose for testing stuck-at faults in ASCs is very similar to that used in scan testing synchronous circuits. When φ_s =0, the ASC illustrated in Figure 4 can perform either in normal operation mode (*Tst*=0) or test mode (*Tst*=1). If *Tst*=0 and φ_d =0 the ASC is set to scan mode. In scan mode, the test patterns are loaded into the latches of the state registers which are configured as a united large scan register. The scan path is created by connecting the output *Sout* of *Reg1* to the input *Sin* of *Reg2* (see Figure 4). Clocks φ_s for controlling the shift operation are generated internally by the STCL. When the test patterns are loaded into the latches the ASC is set to test mode. A request signal is produced on the line *Rin* of the ASC. The results of testing the combinational logic block are stored in the register latches. In scan mode, the contents of the latches are shifted out to the output *Sout* of *Reg2*. The test results are compared with known good ones. Whilst unloading the test results a new test pattern is loaded from the input *Sin* of the ASC and the test procedure can be repeated. The complexity of the testing of the ASC is reduced to the testing of its combinational block.

6.1: Testing for faults in the STCL

The STCL unit of the testable ASC 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 during the test.

A stuck-at fault on any of the lines in the STCL block shown in Figure 5 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 ASC based on the micropipeline approach can be detected since they cause the ASC to halt. This happens because a micropipeline is an event-driven asynchronous circuit [11]. Such stuck-at faults can easily be identified either in normal operation mode or during the test.

6.3: Testing for faults in the processing logic

The internal inputs (*SI*) of the combinational circuit are controllable and its internal outputs (*SO*) are observable through the scan path. Tests for detecting stuck-at faults in the combinational logic block can be derived using well known test generation algorithms such as the D-algorithm, PODEM, FAN and others [5]. The test pattern is loaded into *Reg1* during scan mode. In test mode, when the data is stable on the inputs *PI* of the ASC, a request signal is generated on the input *Rin*. The responses from the outputs *SO* of the combinational circuit are stored in *Reg2*. The test results are analysed on the outputs *PO* of the ASC and an acknowledge signal is produced on its input *Aout*. The ASC is set to scan mode to shift the content of *Reg2* out to the output *Sout* and to load a new test pattern into *Reg1*.

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 φ_d 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 in Figure 2 are detected during test and scan mode.

7: Testing delay faults

There is another class of faults in micropipeline structures which can be detected using the proposed scan test technique. These are delay faults in the combinational circuit. The output data of the combinational circuit 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

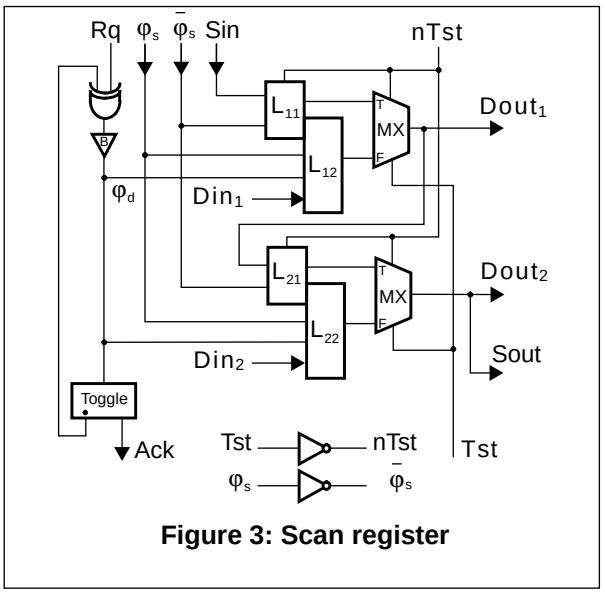

Figure 3: Scan register

the register all the latches L_2 are opened by a high signal φ_d driven by the buffer *B*. The data is transmitted from the inputs *Din* to the outputs *Dout*. Afterwards the request event is steered to the output (marked with a dot) of the toggle element and then to the input of the XOR gate. The output of the XOR gate becomes low (φ_d=0) and the latches L_2 are closed. An acknowledge signal is generated on the output ('blank' output) of the toggle element.

Scan mode. While *Tst*=0 and φ_d=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 φ_s.

Test mode. During the test (*Tst*=1 and φ_s=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 *Dout* of the state register. After receiving a request signal on the line *Rq* the test results from the inputs *Din* are stored into the second latches L_2 of the register (Figure 3). The test vectors and the test results are saved in different latches because the data flows through the ASC from left to right making a loop while the test vectors must be preserved during the test.

5: Scan test control

The design of the testable ASC is illustrated in Figure 4. It comprises the ASC and the scan test control logic (STCL) unit. The state registers of such an ASC are built using scan latches. The STCL block is used to make an asynchronous test interface for the ASC. It also generates clock signals φ_s for a united shift register. The STCL block uses a 2-phase signalling protocol. The STCL can either be a central control block or can be incorporated inside the 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. To reduce the number of external pins for the implementation of the testable ASC the pairs of signals such as *Rin* and *RSin*, *Ain* and *ASin*, *Rout* and *RSout*, *Aout* and *ASout* can be combined using multiplexers. These multiplexers are controlled by a Boolean signal switching between scan and normal operation mode. Note that, for 2-phase signalling, some of the multiplexers must contain state holding elements.

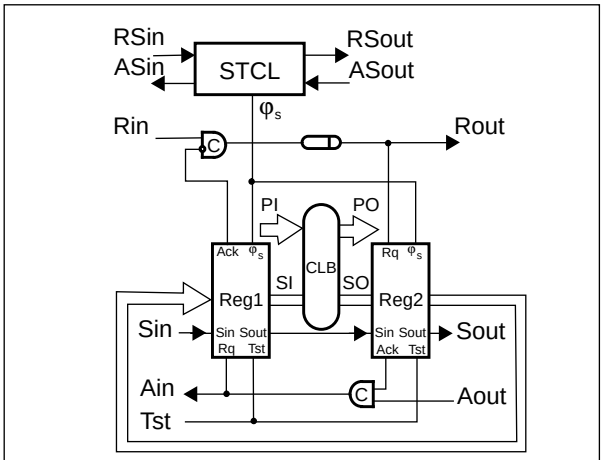

Figure 4: An asynchronous sequential circuit with scan features

An example of the STCL block for 2-phase transition signalling is shown in Figure 5. Such a STCL block processes the control signals in a manner similar to that of the control circuitry for generating the enable signals φ_d in the register illustrated in Figure 3. The C-element is used to ensure the delay insensitivity of the STCL block.

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

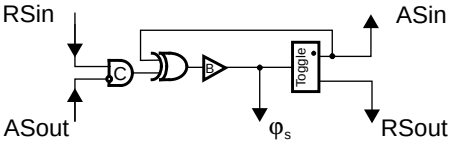

Figure 5: Scan test control logic

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 described allows the detection of all the stuck-at and delay faults in the micropipeline.

A similar scan test technique for testing micropipelines has been reported [13]. The implementation of this test approach is based on the use of specially designed scan latches controlled by the scan test control logic. The proposed scan test design can support the testing of micropipelines which use either 2-phase or 4-phase signalling protocols.

However, these scan test approaches have not been developed for the testing of ASCs built using the micropipeline approach.

4: Scan test design

The scan testing of ASCs requires the testable circuit to be set to scan mode to reconfigure all the latches into a united scan register where the test vectors are shifted in and the responses of the combinational circuit are shifted out.

4.1: Scan latch implementation

Figure 2 shows a CMOS implementation of the scan latch structure used in a testable ASC. The scan latch contains two latches (L_1 and L_2) and a multiplexer.

In normal operation mode (*Tst*=0) the tristate inverter of L_2 is disabled by the shift clock signal set to zero (φ_s=0).

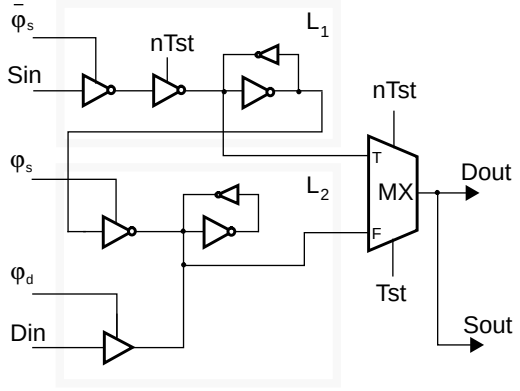

Figure 2: A CMOS implementation of the scan latch

As a result, the data from the outputs of L_1 is blocked. When φ_d=1 the input data (*Din*) passes to the output *Dout* and is latched by L_2 when φ_d=0.

In scan mode (*Tst*=0) the data enable signal φ_d is held at zero so that the tristate buffer of L_2 is disabled, preventing the input data (*Din*) passing through L_2. Shift clocks are generated on the input φ_s ($\bar{\varphi}_s$). When the signal $\bar{\varphi}_s$=0, the scan data from the scan-in input (*Sin*) is latched by L_1. While φ_s=1, L_2 is opened and the data is send to the scan-out output (*Sout*) of the scan latch. When φ_s=0 ($\bar{\varphi}_s$=1), the data latched in L_1 is copied into L_2 and sent to L_1 of the next scan latch of the scan path. This procedure is similar to that used for storing the data in a master-slave flip-flop.

In test mode (*Tst*=1, φ_s=0) the response bit from the combinational logic block is stored in L_2 while the stimulus bit is held unchanged in L_1 and applied to the appropriate input of the combinational circuit. Note that the signal *Tst* must rise before φ_s is set to zero in order not to change the state of L_1 when the new data is latched in L_2.

To analyse the performance of the basic (without scan features) and scan versions of the latch structure, SPICE analyses have been performed on extracted layout from the AMULET1 design implemented on a 1μm CMOS process. The basic latch cell used is similar to a single-phase static CMOS latch and requires 11 transistors [10]. 37 transistors are needed for the implementation of the scan latch. As a consequence, the redundancy of the scan latch is 236%. However, 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 ASC is shown in Figure 3. Compared with the basic register the scan register structure contains four additional wires: test control (*Tst*), scan-in (*Sin*), scan-out (*Sout*) and shift clock line φ_s.

Normal operation mode (*Tst*=0, φ_s=0). Initially all the register latches L_2 are closed. The outputs of the toggle element are set to zero (the reset control line is not shown in Figure 3). When a request signal arrives on the input *Rq* of

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: Basic structure of the ASC

Figure 1 illustrates the general structure of an ASC. This structure contains the combinational logic block (*CLB*) which performs the basic logic operations, and two registers (*Reg1* and *Reg2*) in the feedback loop which store the state of the ASC. The ASC works as a micropipeline. In the initial state, all the latches of *Reg1* are set to their initial states and both the C-elements are set to zero. The input data is generated on the primary inputs (*PI*) of the circuit by the sender which sends a request signal (*Rin*) to the ASC. The request signal is delayed by the delay element for long enough for the output data to stabilize on the primary (*PO*) and internal (*SO*) outputs of the combinational circuit. As a result, a request signal (*Rout*) is produced for the receiver by the sequential circuit. After receiving an acknowledge signal (*Aout*) and storing a new state in *Reg2* the circuit generates an acknowledge signal (*Ain*) for the sender simultaneously causing the copying of the contents of *Reg2* into *Reg1*. When a new request signal is sent by the sender the procedure of processing the data is repeated.

There are different ways to implement the control of latching and storing the data in the latches of the micropipeline registers. These latches are basically 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 opaque 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].

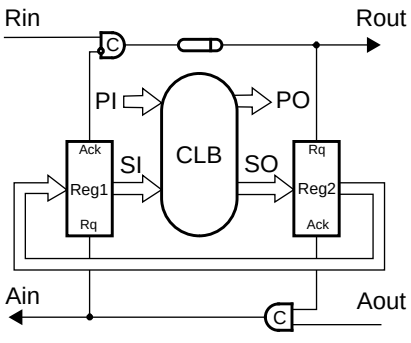

Figure 1: The general structure of an asynchronous sequential circuit

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]. The class of stuck-at faults in micropipelines has been considered [11]. Three types of stuck-at faults for a micropipeline have been identified:

- faults in the control part of the micropipeline;
- faults in logic blocks;
- faults in the latches.

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, in the presence of a stuck-at fault in the control part, the micropipeline moves through at most one step and then halts [11]. Thus, stuck-at faults in the control part of the micropipeline manifest themselves by preventing activities in the micropipeline.

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 combinational circuit test vectors can be obtained using any known test generation technique [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 corresponding stuck-at fault in the combinational logic. To detect a stuck-at-pass fault in the transition latch two test patterns are required.

Unfortunately, this approach can not be used for the testing of ASCs as shown in Figure 1 since the inputs of the state registers are not accessible.

3.2: Scan testing

An elegant scan test approach has been proposed by Khoche and Brunvand [12]. The micropipeline can perform 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

Scan Testing of Asynchronous Sequential Circuits

O. A. Petlin, S. B. Furber

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

Abstract

A method to design and test asynchronous sequential circuits (ASCs) based on the micropipeline design style is presented in this paper. According to the proposed scan test approach the combinational block is tested separately by scanning the test vectors in and shifting the responses out of the state registers. This provides for the detection of all single stuck-at and delay faults in the ASC under test. The complexity of the test procedure of such a testable ASC is reduced to that of the combinational circuit. Tests for the combinational circuit and state holding elements can be derived using standard test generation techniques.

1: Introduction

Asynchronous VLSI designs may have advantages over their synchronous counterparts. Asynchronous circuits have no global clock, so the clock skew problem no longer exists. In addition, asynchronous circuits perform according to typical processing times rather than worst case times and 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 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 cannot be used for the testing of ASCs built on the micropipeline design approach.

In this paper, a scan test methodology for the detection of stuck-at faults and delay faults in ASCs based on the micropipeline approach is proposed. In Section 2, we will describe the basics of the micropipeline design. Section 3 is a survey of results in the area of testing of micropipelines. Elements of the proposed scan test design are presented in Section 4. Section 5 contains implementation details of the scan test control logic. The test strategy for detecting stuck-at and delay faults in ASCs is described in Sections 6 and 7. Section 8 concludes the paper.

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. This means that, 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 signalling protocol both rising and falling transition events have the same meaning. When