# Data-feedthrough faults in circuits using unclocked storage elements

W.K. Al-Assadi, D. Lu, A.P. Jayasumana,
Y.K. Malaiya and C.Q. Tong

*Indexing terms: Asynchronous sequential logic, Logic testing*

Some faults in storage elements (SEs) do not manifest as stuck-at-0/1 faults. These include data-feedthrough faults that cause the SE cell to exhibit combinational behaviour. The authors investigate the implications of such faults on the behaviour of circuits using unclocked SEs. It is shown that effects of data-feedthrough faults at the behavioural level are different from those due to stuck-at faults, and therefore tests generated for the latter may be inadequate.

*Introduction:* A significant proportion of physical failures in CMOS storage elements (SEs) do not manifest as stuck-at-0/1 faults. These include data-feedthrough faults and logic retention faults. The enhanced fault model proposed in [1] includes all the logical testable faults that do not manifest as stuck-at-0/1 as well as stuck-at faults and faults that manifest as delay faults. Table 1 shows the faulty behaviour of two unclocked SEs commonly used in asynchronous design, the NAND-pair latch and the C-element [2]. The enhanced fault model proposed for such SEs shows high explicit fault coverage compared with the stuck-at model. Here the term data-feedthrough faults refers to the faults that cause the SE to exhibit a combinational behaviour that can be described as a combinational function of the input data. The C-element latch exhibits conditional non-retention of logic 1/0 (CNR-1/0) behaviours, under which the SE fails to retain the logic level under a specific input vector. Indeterminate behaviours are modelled as parametric and can only be detected by observing the supply current ($I_{DDQ}$). In this Letter, implications of data-feedthrough faults in circuits using unclocked SEs are considered. The effect of such faults is shown to be different from that due to stuck-at faults.

**Table 1**: Faulty behaviour of two unclocked SEs

| Circuit | Enhanced model | | | | Parametric | Fault-free behaviour |
|---|---|---|---|---|---|---|
| | Stuck-at | Data-feedthrough | CNR | Delay | | |
| NAND-pair | 47% | 42% | – | 2% | 2% | 6% |
| C-element | 53% | 15% | 23% | – | 9% | – |

*Implications of data-feedthrough faults:* In this Section we show that low level physical failures that cannot be modelled as stuck-at faults can be functionally characterised at a higher level. Therefore accurate functional fault models for complex logic blocks can be derived considering the low level fault model of the primitives such as the elementary SEs without losing information about the structure of the circuit. This can reduce test generation efforts by allowing a low level failure to be tested functionally without the need to consider test generation at low level.

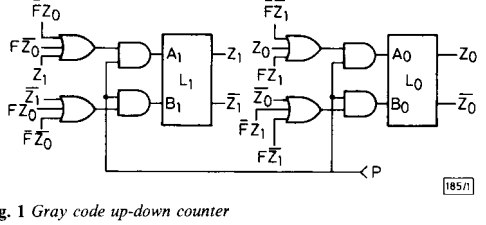


**Fig. 1** *Gray code up-down counter*

*(i) Example 1:* Consider the asynchronous Gray code counter shown in Fig. 1 using two NAND-pair latches $L_1$ and $L_0$. $F = 1(0)$ causes the counter to count up(down). Fig. 2 shows the state diagrams of the fault-free circuit as well as faulty circuits in the presence of data-feedthrough and stuck-at-0/1. Fig. 2 indicates that the faulty behaviour for stuck-at faults is different from that in the presence of data-feedthrough faults.

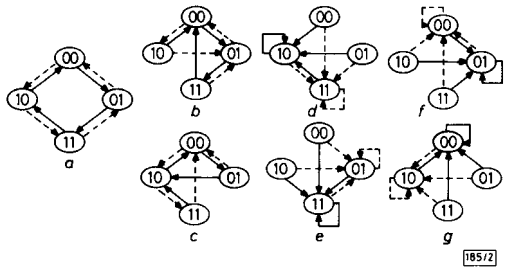


**Fig. 2** *Comparison of behaviour for stuck-at faults and data-feedthrough faults*

*a* State diagram of fault-free circuit
*b* $L1$ data-feedthrough ($Z1 = \overline{A1}$)
*c* $L0$ data-feedthrough ($Z0 = \overline{A0}$)
*d* $L1$ stuck-at-1
*e* $L0$ stuck-at-1
*f* $L1$ stuck-at-0
*g* $L0$ stuck-at-0

For asynchronous circuits with $n$ latches, $2^n$ states are possible. When one of the latches becomes combinational, the maximum number of states may decrease to $2^{n-1}$. This is not always true as shown in example 1. However, feedback in the circuit may be such that latching is still possible and it is possible to have more than $2^{n-1}$ states.

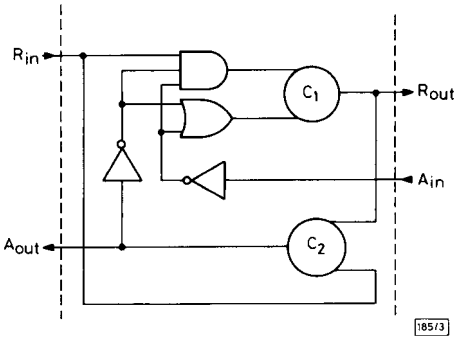


**Fig. 3** *Handshaking circuit using C-element*

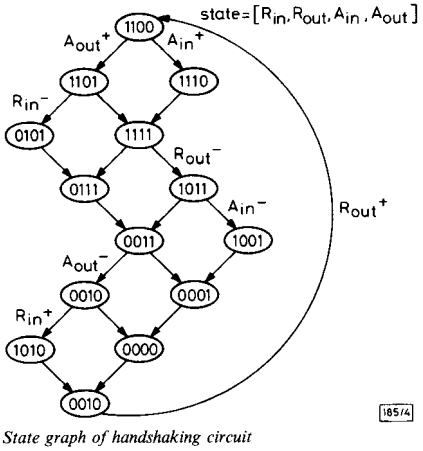


**Fig. 4** *State graph of handshaking circuit*

*(ii) Example 2:* Another example is the handshaking circuit implemented by the C-elements shown in Fig. 3. This circuit is semimodular which is the necessary and sufficient condition for an asynchronous circuit to be hazard-free. Such a circuit ensures that

every closed path in its state graph includes all possible transitions of all state variables as shown in Fig. 4. This circuit is self-diagnostic, i.e. any stuck-at fault within the circuit will halt the operation, and the detection of such faults can be guaranteed [3]. For example, a stuck-at-1 fault at the line $R_{out}$ of the handshaking circuit of Fig. 3 will make it impossible for transition $R_{out}^-$ to occur. Hence the circuit will eventually get stuck at a unique state of (0111) independent of the inputs and initial states (see Fig. 5). This implies that there is no need to generate any tests for detecting a stuck-at fault in this circuit. Consider the data-feedthrough fault that occurs when the two inputs of C2 in Fig. 3 are shorted. Because of this fault, C-element C2 is reduced to an AND gate. The effect of this fault can be seen from the faulty state graph in Fig. 6. The data-feedthrough fault destroys the proper sequence of transitions of circuit states. For example, transition $A_{out}^-$ should happen only after transitions $R_{out}^+$ and $R_{in}^-$ in the fault-free circuit. But because of the fault, $A_{out}^-$ can happen before $R_{out}^+$ or $R_{in}^-$. For this reason two new states are possible and several new transitions are created. Unlike the stuck-at fault, this fault is not self-diagnostic because the new state graph is no longer semi-modular and the circuit can have such an operating cycle as $[A_{out}^+, R_{in}^-, A_{out}^-, R_{in}^+]$. Therefore, specific effort is needed to generate a proper test sequence to detect it. Moreover, this data-feedthrough fault may mask the detection of some other stuck-at faults such as $R_{out}$ stuck-at-one fault.

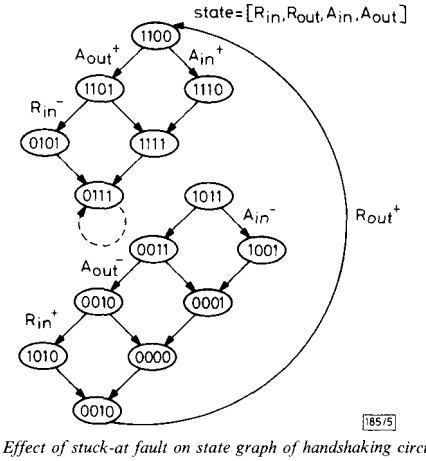


**Fig. 5** *Effect of stuck-at fault on state graph of handshaking circuit*

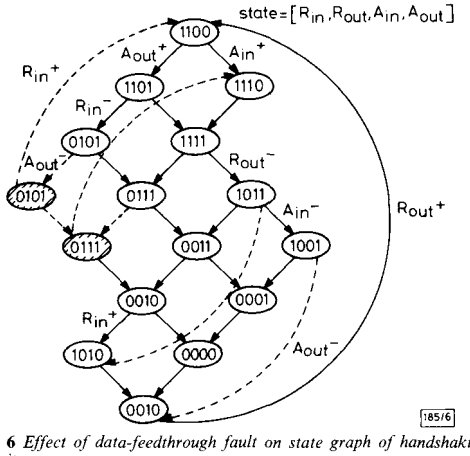


**Fig. 6** *Effect of data-feedthrough fault on state graph of handshaking circuit*

*Conclusion:* The testing of asynchronous circuits has received much attention. Existing testing methodologies are based on the stuck-at model. Analysis of the data-feedthrough behaviour indicates a need for developing techniques to predict the behaviour of asynchronous circuits in the presence of data-feedthrough faults and test generation for such faults.

*Acknowledgments:* This work was sponsored partly by an SDIO/IST funded project monitored by ONR.


*21 February 1994*
*Electronics Letters Online No: 19940514*

W.K. Al-Assadi, D. Lu, A.P. Jayasumana and C.Q. Tong (*Department of Electrical Engineering, Colorado State University, Fort Collins, CO 80523, USA*)

Y.K. Malaiya (*Computer Science Department, Colorado State University, Fort Collins, CO 80523, USA*)

Corresponding author: A.P. Jayasumana

**References**

1 AL-ASSADI, W.K., MALAIYA, Y.K., and JAYASUMANA, A.P.: 'Faulty behavior of storage elements and its effects on sequential circuits', *IEEE Trans. VLSI Systems*, 1993, **1**, (4), pp. 446–452

2 AL-ASSADI, W.K., LU, D., JAYASUMANA, A.P., MALAIYA, Y.K., and TONG, C.: 'Detailed examination of asynchronous storage elements'. Proc. NASA Symp. on VLSI Design, 1993, pp. 7.4.1–7.4.9

3 BEEREL, P.A., and MENG, T.H.-Y: 'Semi-modularity and testability of speed-independent circuits', *Integration, the VLSI Journal*, 1992, **13**, pp. 301–322

## Delay fault propagation in synchronous sequential circuits

P. Cavallera, P. Girard, C. Landrault and S. Pravossoudovitch

*Indexing terms: Logic testing, Sequential circuits*

Compared with the propagation of logic errors produced by stuck-at faults, the propagation of gate delay fault effects in sequential circuits poses some particular problems. The authors first describe the propagation conditions of such faults then define the propagation rules of these faults which are used in a new delay fault simulation process for synchronous sequential circuits.

*Introduction:* Different techniques have been proposed in the past to simulate stuck-at faults in sequential circuits. However, these techniques cannot be directly applied when gate delay faults are considered. The main reason is that timing instead of logic effects are induced by such faults during simulation. To the authors' knowledge, all the methods proposed earlier to simulate gate delay faults are limited to gross delay faults (transition faults) [3]. In this Letter, we assume that the problem of simulating gate delay faults in combinational circuits (or combinational blocks) is solved [1], and we focus on the propagation of delay fault effects through synchronous sequential circuits. To this end, we define rules for propagating delay faults irrespective of their size. These rules will be used in a new method for delay fault simulation in sequential circuits.

*Error propagation in sequentials circuits:* In synchronous sequential circuits, the effects of a delay fault are different according to whether we consider the fault effect in the faulty combinational block or in the blocks located after a memory element (flipflop). A delay fault produces delayed transitions in the faulty combinational block, but the delayed transitions are either filtered or changed into logic errors after propagation through a memory element. This depends on the fault size and the delay of the fault propagation path with respect to the sample time provided by the clock signal (Fig. 1).

From the above, it appears that different fault sizes may result in different faulty circuit behaviours. If the fault size is explicitly considered by the fault model, the circuit behaviour can easily be predicted. However, a fault simulation process that uses such a fault model is not realisable, because it is not possible to divide fault sizes into fine-grained ranges and simulate each fault of a given size separately. Therefore, the fault model used for simulation cannot explicitly consider the fault size, due to the high computational complexity obtained to simulate all faults.