Single Stack-at Fault Effect on PCFB Templates

Masoud Zamani **Mehrdad Najibi** **Hossein Pedram**
Amirkabir University of Technology
Department of Computer Eng. & IT
{m-zamani, najibi, pedram} @aut.ac.ir

Abstract - *Template based synthesis of asynchronous circuits is accepted as an effective way to high level design of them. One of the popular pre-designed templates that most synthesis tools use it to synthesis QDI asynchronous circuits is Pre-Charged Full Buffers (PCFB). Deferent synchronization methods in synchronous circuits and asynchronous ones cause deferent fault effects in these two types of circuits. Therefore methods used to test synchronous circuits are not directly applicable to asynchronous circuits. To have suitable test strategy in asynchronous circuits, it is necessary to explore fault effect in them. So In this paper, we study the effects of single stuck-at fault in PCFB templates and categorize fault effects. This categorization is useful for testable design and introducing test methods for template based circuits.*

Keywords: *Test, QDI asynchronous circuit, template based design, fault categorization.*

1 Introduction

Asynchronous design offers a number of promising features for low power, low electromagnetic compatibility, no clock skew problems and high-security applications [1][4]. However, it is not a well- established and widely-used design methodology. The main reasons for this can be enumerated as: 1) higher complexity of asynchronous circuit design, 2) insufficient development of testing methodology.

Fault detection in QDI (Quasi Delay-Insensitive) circuits requires newer techniques than those used in synchronous architectures [2]. In a clocked system where we assume the clock always works, one can always sample the output of the circuit at the clock edge and use those values to detect any fault. In contrast, asynchronous signaling protocols embed data validity in the signal values themselves. Therefore faults can lead to invalid circuit faulty state or in the simplest case deadlock (Note that a similar problem could arise in a clocked system when part of the clock network has a fault[2].

Persia is an asynchronous synthesis tool developed for automatic synthesis of QDI asynchronous circuits [3]. It is a high level synthesis tool based on martins synthesis method [5]. Persia uses pre-designed templates. Each template has QDI structure which communicates with other templates using *Delay-Insensitive* (DI) handshaking. In [6][7], it is proven that all DI modules are fully self-testable. So in template-based synthesis tools like Persia, test problem can be reduced to testing pre-designed templates. In this paper we propose new fault categorization for effect of single stack-at faults in PCFB templates.

This paper is structured as follows. We begin with an overview of related works on asynchronous circuit testing (Section 2). In section 3 asynchronous circuit design will be discussed. PCFB temples will be discussed in section 4. Section 5 explores single stuck-at fault effects on PCFB templates and categorization them. Section 6 offers some conclusions.

2 Related works

Based on our good knowledge there isn't any literature that targets template based asynchronous circuit testing, however there are some literatures that study testing of asynchronous circuits which will be reviewed here.

Self-synchronizing nature of some asynchronous circuits yields SC (Self Checking) property such that the circuit detects particular types of faults during operation[8]. Martin [7] demonstrated that a single stuck-at fault in a nonredundant delay-insensitive circuit causes a transition either not to occur or causes a transition on output to become enabled in an illegal state.

Varshavsky [9] showed that semi modular circuits are totally SC, Beerel [8] related the semi modular circuit testability results [9] to SI (speed-independent) circuits and extended the study of output stuck-at faults (OSAF's) to certain input stuck-at faults (ISAFs). These results are impressive and have led to new approaches to testing asynchronous circuits, based on the tendency of such circuits to halt in the presence of stuck-at faults. However, such SC abilities are limited to the scope of asynchronous control circuits. Previous analysis of faulty/mis-behaving asynchronous circuits has been done using the stuckat-0/stuck-at-1 fault model [10]. This work examines the effects of stuck-at faults in DI, QDI, and SI circuits.

3 Asynchronous circuit design

Asynchronous circuits represent a class of circuits not controlled by a global clock but rely on exchanging local request and acknowledge signaling for the purpose of synchronization. In fact, an asynchronous circuit is composed of individual modules which communicate with each other by means of point-to-point communication channels. Therefore, a given module becomes active when it senses the presence of an incoming data. It then performs the computation and sends the result via output channels. Communications through channels are controlled by handshake protocols[11].

An asynchronous circuit is called delay-insensitive if it preserves its functionality independent of the delays of gates and wires [11]. It is shown that the range of the circuits that can be implemented completely delay-insensitive is very limited[1].Therefore some timing assumptions exist in different design styles that must be hold to ensure the correctness of the circuit. Different asynchronous techniques distinguish themselves in the choice of the compromises to the delay-insensitivity. Quasi delay-insensitive (QDI) circuits are like delay-insensitive circuits with a week timing constraint: isochronic forks. In and isochronic fork the difference between the delay through the branches must be less than minimum gate delay. QDI implementations appear to be the most appropriate class of asynchronous circuits that can be synthesized automatically from large high-level behavior specifications. This is because of the week timing constraint that can be easily managed in this design style. Return to zero handshaking protocol with dual-rail data encoding that switch the output from data to spacer and back is the most common QDI implementation form. The most efficient QDI implementations are based on per-charge logic. That makes it easy to incorporate existing dynamic domino style power balanced structures in the QDI templates.

Table1: Dual rail coding

	d.t	**d.f**
Neutral("E")	0	0
Valid '0'	0	1
Valid '1'	1	0
Not used	1	1

The encodings of the channels can be in a variety of ways. We use a dual rail encoding here the data channel contains a valid data (token) when exactly one of 2 wires are high. When the two wires are lowered the channel contains no valid data and is called to be neutral (Table1).

One of the major protocols used in asynchronous circuits is four phase protocol. In a four phase protocol's sequence a receive action consists of four steps. (1) Wait for input to become valid. (2) Acknowledge the sender after the computation performed. (3) Wait for inputs to become neutral. (4) And lower the acknowledgement signal. A send action consists of four phases: (1) send a valid output. (2) wait for acknowledge. (3) Make the output neutral.(4) wait for acknowledge to lower output .figure 1 shows a four phase handshake sequence.

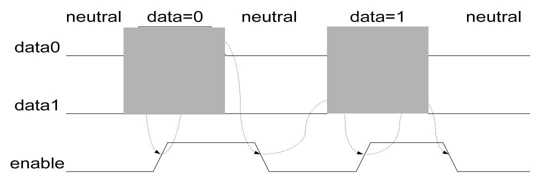

Figure 1: Four-phase protocol.

4 PCFB templates

At present, most QDI circuits are designed using PCHB and PCFB (Pre-Charge logic Full-Buffer) templates[11][3][1]. Circuit of PCFB is similar to pre-charge domino-logic style circuits in synchronous designs except that instead of a global pre-charge signal, local pre-charge signals are generated. The internal implementation of the simple buffer comprised five sub-circuits (Figure 2: 1- Output generation circuit. 2-Input validity check circuit. 3- Output validity check circuit. 4- A sub circuit that generates the acknowledgement for inputs. 5- A sub circuit that generates en (enable) signal.

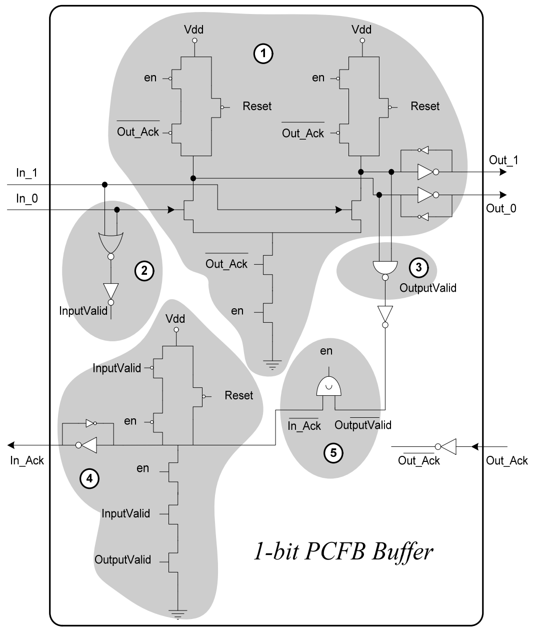

Figure 2: The PCFB 1-bit buffer

A PCFB template is an asynchronous buffer circuit that in each cycle of its operation reads some inputs, performs a particular calculation, and then writes the results to one or more of its output ports. A PCFB can have multiple inputs and outputs, have conditional inputs and outputs, and hold states. All I/O read or write operations are done using dual-rail four-phase handshaking protocol. In dual rail encoding, the data channel contains a valid data when exactly one of 2 wires is high. When the two wires are lowered the channel contains no valid data and is called to be neutral. In a four-phase protocol's sequence a receive action consists of four steps. Figure 1 shows a four phase handshake sequence.

The follow pseudo code describes how a PCFB template works:

```
Initial en=1;
always begin
    wait (~OutputAck);
    wait (Input_Valid);
    Output=F (Input);
    InputAck=1;
    en=0;
    fork
        begin
            wait (OutputAck);
            Output=Neutral;
        end
    begin
        wait (Input_Neutral);
        InputAck=0;
    end
join
en=1;
end
```

5 Single stuck-at fault effects in PCFB templates

Testing QDI circuits, using the stuck-at model, is thoroughly explored in [6]. This testing method classifies a fault as: 1) inhibiting (preventing an action) which causes circuit to halt during test, so these faults are testable 2) stimulating (causing an action) which cause a premature firing of a signal or signals, identifies faults that can't be observed easily. As PCFB templates have QDI nature so inhibiting faults in these circuits prevents or stops the four phase protocol between each template; therefore these faults are testable. But to have more suitable fault model in these circuits, it is necessary to study the effect of premature firing of signals on PCFB templates in detail. In follow we explore premature firing faults in PCFB templates.

PCFB templates are designed based on dual rail protocol. Therefore, changing one bit of a valid data encoding (either (0,1) or (1,0)) results in one of the metadata states: quiet (0,0) or alarm (1,1). Thus single faults can not change the value of a token. In order to explore premature firing effects, we simulate faulty templates by applying stimulated faults to transistor-level model of PCFB templates in verilog-HDL. Results show that premature firing of a signal causes two types of error behavior in a template:

1- Premature firings those cause redundant token (valid data) generation: The fault causes some redundant tokens to be generated within the circuit due to following issues:

- Positive edge of Output-Valid takes place earlier than negative edge of Input-Ack (e.g. en stuck-at 1 in the pull-down network of Functional unit).
- Positive edge of Input-Ack takes place earlier than positive edge of Output-Valid (e.g. en stuck-at 0 in the pull-up network of functional unit).
- Negative edge of Input-Ack takes place earlier than negative edge of Input-Valid (e.g. Input-Valid stuck-at 0 in the pull-up network of Input-Ack generation unit)

2- Premature firings those causes some tokens within the circuit to be dropped due to following issues:

- Negative edge of Output-Valid takes place earlier than positive edge of Output-Ack (e.g. Output-Ack stuck-at 1 in the pull-up network of functional unit)
- Positive edge of Output-Valid takes place earlier than negative edge of Output-Ack (e.g. Output-Ack stuck-at 0 in the pull-up network of Functional unit).

So we can conclude that single stuck-at fault in PCFB template cause three effects:

- circuit to be deadlock
- redundant token generation
- token dropping

To have more results, we synthesized primitive gates to PCFB templates and injected single stack-at fault to these templates. Simulation results show that, percentage of each fault category is as table 2.

Table 2: percentage of fault effects in PCFB primitive gates

	Deadlock	Token Consume	Token Generation
And/Or	75.80%	11.9%	12.3%
Xor/Xnor	75.64%	11.68%	12.68%
Buffer/Not	75.22%	12.29%	12.49%
Nand/Nor	75.80%	11.9%	12.3%

As mentioned earlier deadlock is self testable, but to testing two other fault categories it is necessary to introduce some other circuits. These circuits can be based on token counting in circuits.

6 Conclusions

Complexity of design and testing asynchronous circuits are the major obstacles for widespread use of them in digital circuit design. Template based synthesis of asynchronous circuit decreases complexity of design. Using template based QDI asynchronous circuit decrease testing problem to pre-designed templates in library of synthesis tool this is due to the DI behavior to circuit outside of templates. One of the popular pre-designed templates that most synthesis tools use it as template is PCFB. In this paper we studied effect of single stuck-at fault in PCFB templates and categorized fault effects. Thus categorization is useful for testable design of templates. By using testable templates, testing will be easier in template based asynchronous circuit designs compared to other types of asynchronous circuit.

References

[1] A. M. Lines "Pipelined Asynchronous circuits" MSc Thesis, California Institute of Technology, June 1995, revised 1998

[2] C. LaFrieda, R. Manohar, "Fault Detection and Isolation Techniques for Quasi Delay-Insensitive Circuits," In Proc. of International Conference on Dependable Systems and Networks, Italy, June 28- July 01, 2004.

[3] Persia Site: http://www.async.ir/persia/persia.php

[4] Scott Hauck, Asynchronous Design Methodologies: An Overview, Department of Computer Science and Engineering, University of Washington, Seattle, TR 93-05-07, 1993.

[5] Alain J. Martin, "Synthesis of Asynchronous VLSI Circuits"Caltech,CS-TR-93-28, 1991

[6] P. J. Hazewindus. Testing Delay-Insensitive Circuits. PhD thesis, California Institute of Technology, Pasadena, California, 1996.

[7] A. J. Martin and P. J. Hazewindus, "Testing delay-insensitive circuits," in Proc. Univ. California Santa Cruz Conf.: Adv. Res. VLSI, 1991, pp. 118–132.

[8] P. Beerel and T. Y. Meng, "Semi-modularity and testability of speedindependent circuits," Integration, VLSI J., vol. 13, pp. 301–322, 1992.

[9] V. I. Varshavky, Ed., Self-Timed Control of Concurrent Processes. Dordrecht, The Netherlands: Kluwer, 1990

[10] H. Hulgaard, S. M. Burns, and G. Borriello. "Testing asynchronous circuits: a survey". Integr. VLSI J., 19(3):111–131, 1995.

[11] Jens Sparso, Steve Furber, "Principles of Asynchronous Circuit Design-A System Prespective", Kluwer Academic Publishers, 2002.

[12] A. J. Martin. The limitations to delay-insensitivity in asynchronous circuits. Beauty is our business: a birthday salute to Edsger W. Dijkstra, pages 302–311, 1990.

[13] R. Manohar and A. J. Martin. Quasi-delay-insensitive circuits are Turing complete. In Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems. IEEE Computer Society Press, 1996.