Testable Design of Template based QDI Asynchronous Circuits

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

Abstract

Complexity of design and testing are the major obstacle for widespread use of asynchronous circuit in digital circuit design. Template based synthesis of asynchronous circuit is accepted as an effective way to decrease complexity of design of asynchronous circuit. One of the popular pre-designed templates that most synthesis tools use it to synthesis QDI asynchronous circuit is Pre-Charged Full Buffers (PCFB). In this paper we study the effect of single stuck-at fault in PCFB templates and categorize fault effects, then for each category introduce a circuit that detect faults. Experimental results show that using these additional circuits in PCFB buffers can improve the testability from 68.55 percent to 90.32 percent.

1. Introduction

Asynchronous design offers a number of promising features for low power, low electromagnetic compatibility, no clock skew problems and high-security applications [6][7]. 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.

Persia is an asynchronous synthesis tool developed for automatic synthesis of QDI asynchronous circuits [4]. Persia uses pre-designed templates. Each template has QDI structure which communicates with other templates using *Delay-Insensitive* (DI) handshaking. In [13], 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 simple circuits added to templates to increase testability of PCFB templates.

Fault detection in QDI (Quasi Delay-Insensitive) circuits requires newer techniques than those used in synchronous architectures [8]. 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 [8].

In this paper, circuits are analyzed at transistor-level abstraction, with single stuck-at-fault model.

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. Section 4 introduces Persia, A QDI asynchronous synthesis tool, and PCFB template. Section 5 explores single stuck-at fault effects on PCFB templates and categorization them. Proposed circuits to testable design of PCFB are described in section 6. Section 7 provides results of testing a ring buffer that is synthesized by the proposed testable PCFB templates.

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 [9]. Martin [10] 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 [11] showed that semi modular circuits are totally SC, Beerel [9] related the semi modular circuit testability results [11] 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 [12]. 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 [2].

An asynchronous circuit is called delay-insensitive if it preserves its functionality independent of the delays of gates and wires [2]. 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.

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).

Table1: Dual rail coding

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

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.

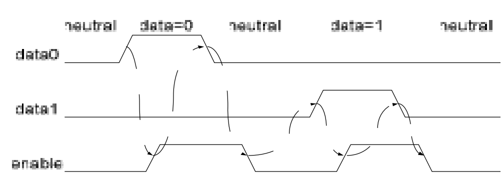


Figure 1: Four-phase protocol

4. Persia: A QDI asynchronous synthesis tool

Persia is an asynchronous synthesis tool developed for automatic synthesis of QDI asynchronous circuit [4]. The structure of Persia is based on the design flow shown in figure 2 which can be considered as the following three individual portions: QDI synthesis, layout synthesis, and simulation at various levels. The simulation flow is intended to verify the correctness of the synthesized circuit in all levels of abstraction.

CSP (Communicating Sequential Processes) is a well-known language for description of concurrent systems which is accepted as a good description language for asynchronous systems. Persia uses

Verilog-CSP [3], an extension of the standard Verilog which supports asynchronous communications as the hardware description language for all levels of abstractions except the netlist which uses standard Verilog. The input of Persia is a Verilog description of a circuit. This description will be converted to a netlist of standard-cell elements through several steps of QDI synthesis flow. For simpler synthesis first arithmetic operations are extracted from the code and the major steps of synthesis only works on the codes without any arithmetic operations. This is done by the AFE which also replaces the arithmetic functions by standard library modules. The two major steps in Persia synthesis are Decomposition and TSYN. In the following three subsections we briefly describe the functionality of these three stages.

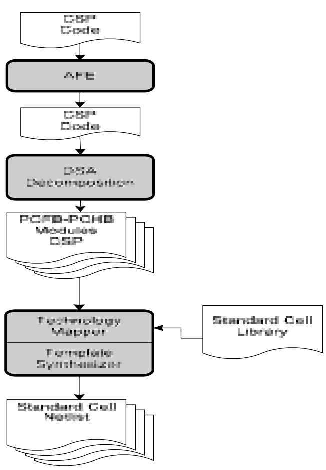


Figure 2. Persia synthesis flow [4]

4.1 AFE

Arithmetic operations are not synthesizable by TSYN (part of Synthesizer), so Persia extracts these operations from the CSP source code and then implements them with pre-synthesized standard templates. AFE extracts each assignment that contains arithmetic operations like addition, subtraction, comparison, etc and generates a tree of standard circuits which implement the extracted assignment.

4.2 Decomposition

Our synthesis approach is based on pre-design asynchronous four-phased dual rail templates. Each template can be considered as a simple pipeline stage. The most renowned form of these templates is named as pre-charge full buffer (PCFB)[2].A PCFB reads its data from input ports, performs the computations and writes the results on the output ports. A PCFB can have multiple inputs and outputs, have conditional inputs and outputs, and hold states. The circuit 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 QDI timing constraint (i.e. Isochronic fork) is local to each template. Figure 3 represents a PCFB buffer used in Persia synthesis tool.

The high-level Verilog-CSP description of even very simple practical circuits is not directly convertible to PCFB. The intention of Decomposition stage is to decompose the original description into a collection of smaller interacting processes that is compatible to these templates and is synthesizable in next stages of QDI synthesis flow

4.3 TSYN

Template Synthesizer, as the final stage of QDI synthesis flow, receives a Verilog-CSP source code containing a number of PCFB-compatible modules and optionally a top-level netlist and generates a netlist of standard-cell elements with dual-rail ports that can be used for creating final layout. The output of TSYN can be simulated in standard Verilog simulators by using the behavioural description of standard-cell library elements.

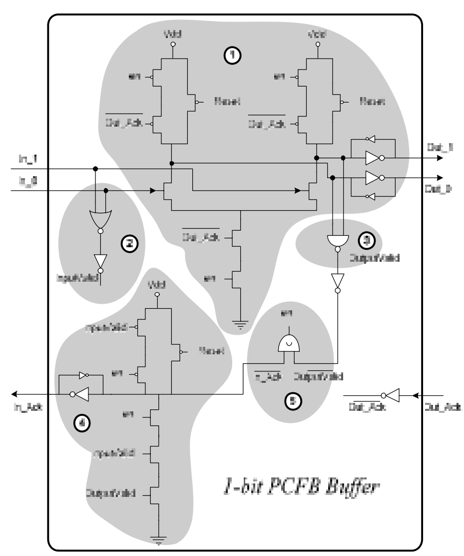


Figure 3. The PCFB 1-bit buffer

4.4 PCFB templates

At present, most QDI circuits are designed using PCHB and PCFB (Pre-Charge logic Full-Buffer) templates[5]Our synthesis Tool uses PCFBs for its predefined templates. The circuit 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 3: 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.

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.

5. Single stuck-at fault effects in PCFB templates

Testing QDI circuits, using the stuck-at model, is thoroughly explored in [13]. 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. By simulating faults which cause premature firing in PCFB template we categorized their effects as follow:

1. Premature firings those introduce isochronic fork problems in circuit: An isochronic branch occurs when a wire forks to multiple gates, and at least one of those gates doesn't acknowledge a transition on that wire. In PCFB templates there are some additional transistors for preventing isochronic fork in template (e.g. en transistors in Input-Ack generation unit of PCFB), when these transistors will be permanent ON because of fault then isochronic fork problems introduced.

2. 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)

3. 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).

By simulating faulty PCFB, percentage of each fault in PCFB is as table 2. Since delay of transistor and wires affect fault behavior in QDI circuits, we simulate our templates with respect to minimum and maximum extreme delays, which these extreme delays back annotated from library cell layouts.

Table2: Percentage of Fault effects in Various Delays

	Deadlock	Isochronic fork	Token Generation	Token Consume
Worst Case	66.13	9.68	11.3	12.9
Best Case	70.97	9.68	9.68	9.68
Average	68.55	9.68	10.49	11.29

Based on [13] faults those not inhibiting are not testable, so in the worst case 33.87 percent of faults in PCFB aren't testable. In the next section, proposed circuits for testable design of PCFB will be introduced.

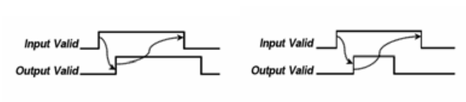
6. Testable design of PCFB

In these section based on fault category, proposed circuits to detecting faults that affect number of token in circuit will be introduced.

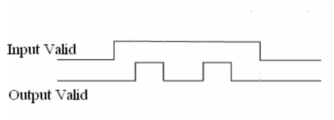
6.1 Detecting faults those cause redundant token generation in the circuit:

Redundant token generation means that for one input token, circuit produces more than one output token. In the other word for an up transition on Input-

Valid signal more than one up transition on Output-Valid signal been seen. Testable design for this type of faults requires to checking sequence of Input-Valid and Output-Valid signals. In Figure 5.a, correct sequences of Input-Valid and Output-Valid signals have been shown.



a



b

Figure 5. Correct sequences of Input-Valid and Output-Valid (a); Faulty sequence of Input-Valid and Output-Valid that cause redundant token in circuit (b)

For faults those cause redundant token in circuit sequence of these signals will be changed as figure 5.b. To detect this incorrect sequence proposed circuit is as figure 6.

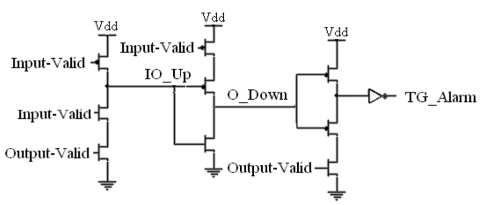


Figure 6. Proposed circuit to detect redundant token generation

This circuit operates as follow: when circuit start to work value of TG_Alarm, IO_Up and O_Down signals are 0, 1, and 0 respectively. IO_Up will be 0 when first up transition of Input-Valid and Output-Valid be detected, after that if there is another up transition on Output-Valid without any transition on Input-Valid, fault be detected by activating the TG-Alarm(Figure 7 illustrate this situation.).

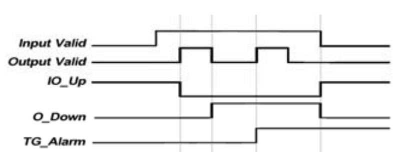


Figure 7. Sequences of signals in redundant token detection circuit

6.2 Detecting faults those cause token dropping

Token will be dropped when sequence of Output-Valid and Output-Ack change (correct sequence of these signals has shown in figure 8.a) as follow:

1) Negative edge of Output-Valid takes place before than positive edge of Output-Ack. Figure 8.b illustrates this situation.

2) Positive edge of Output-Valid takes place when Output-Ack is 1. Figure 8.c illustrates this situation.

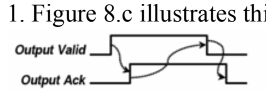
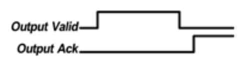
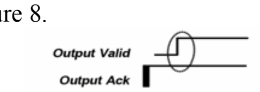


Figure 8.



b



c

Figure 8. Correct sequence of Output Valid and Output Ack signals (a); Faulty sequences of Output Valid and Output Ack signals that cause token drooping (b,c)

The circuit shown in Figure 9 is proposed to detect those faults illustrated in Figure 8.b

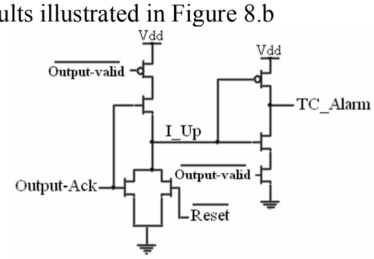


Figure 9. Proposed circuit to detect fault that illustrated in figure 8.a

In Figure 9 TC_Alarm activate when Negative edge of Output-Valid takes place before than positive edge of Output-Ack.

To detect fault which illustrates in Figure 8.c, proposed circuit is as Figure 10.

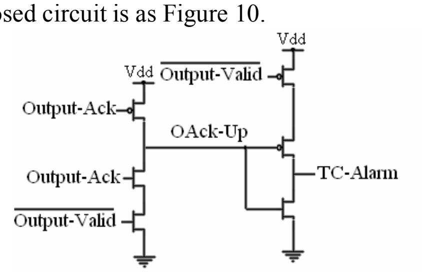


Figure 10. Proposed circuit to detect fault that illustrated in figure 8.c

In order to testable design of PCFB templates, it is enough to add proposed circuits to templates.

8. Conclusion

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, then for each category we introduced a circuit that detect faults. By using testable templates, testing will be easier in template based asynchronous circuit designs compared to other types of asynchronous circuit.

10. References

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

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

[3] Arash Seifhashemi, Hossein Pedram, "Verilog HDL, Powered by PLI: a Suitable Framework for Describing and Modeling Asynchronous Circuits at All Levels of Abstraction", *Proc. Of 40th DAC, Anneheim, CA, SA*, June 2003.

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

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

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

[7] V. Yakovlev, A. M. Koelmans, A. Semenov and D. J. Kinniment, Modelling, analysis and synthesis of asynchronous control circuits using {Petri} nets,Integration, the VLSI journal, Pages 143–170, Vol 21, No. 3, Dec 1996.

[8] 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.

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

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

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

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

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