High-Level Fault Simulation Methodology for Template-Based Asynchronous Circuits

Masoud Zamani , Hossein Pedram and Behnam Ghavami

Amirkabir University of Technology, Department of Computer Eng. & IT,
Tehran, Iran

Summary
Complexity of design and the lack of suitable test methodology are the major obstacles for widespread use of asynchronous circuit in digital circuit design. Template based synthesis of asynchronous circuits is accepted as an effective way to decrease the complexity of design. However, test frameworks such as fault simulator for synchronous circuits are not applicable for template based asynchronous circuits. In this paper we study transistor-level single stuck-at faults in traditional asynchronous templates and categorize their effects on the functionality of circuit. We prove by a mathematical specification that single stack-at fault in Pre-Charge Full Buffer templates has one of the three effects: deadlock, token generation and token dropping. This categorization is used to introducing a new high level fault simulation methodology for these circuits. Based on this strategy we develop a fault simulator and experimental results show the effectiveness of the proposed fault simulation methodology.

Keywords:
Asynchronous Circuit, Fault Simulation, Production Rule, Template Based.

1. Introduction

Asynchronous design techniques have been studied since the 1950s, and in the last 20 years, a series of successful chip-design projects have increased interest in the field [1]. Technological breakthroughs and increased emphasis on performance have motivated researchers and designers to reconsider the asynchronous design methodology[2]. Asynchronous circuits promise high performance gains and low power when compared to their synchronous counterparts. However, until recently these obvious advantages had been overlooked due to the inherent complexities associated with the design and testing of asynchronous circuits. Testing asynchronous circuits is a difficult task, because of two main reasons; first, the absence of a global clock does not allow the use of traditional test generation techniques used for synchronous circuits. Second, correct (i.e. hazard-free) operations of asynchronous circuits are usually obtained by introducing redundancies, that is, sacrificing the testability[3].

Unfortunately, methods used to test of synchronous circuits are not directly applicable to asynchronous circuits. This is due, in large part, to the absence of the global clock signal in the asynchronous circuits. New methods are required to adapt the rich knowledge on testing synchronous circuits to test asynchronous circuits.

In this paper we extend previous fault categorization of QDI (Quasi Delay-Insensitive) circuits to templates and study effect of transistor level single stack-at faults in the functionality of the templates then we change fault effect categorization based on functionality effect of faults used to introduce a high-level fault simulation methodology. In order to evaluate our method we develop a fault simulator tool by this strategy. This tool, to the best of our knowledge is first of its kind for simulating template-based QDI circuits. It provides a framework for monitoring tokens (valid data) in the circuit. Proposed fault simulator is added to Persia synthesis flow. Persia is a synthesis tool for QDI asynchronous circuit [4]. It synthesizes high-level description of circuit to pre-designed PCFB (Pre-Charged Full Buffer) templates, which produces high speed fine grained pipeline.

The rest of paper is organized as follows: In the next section, related works will be overviewed; Section 3 overviews the QDI asynchronous circuit design in brief. Persia synthesis tool and PCFB template will be introduced in section 4 and 5, respectively. Fault categorization and proposed fault simulator will be introduced in Section 5. Section 6 shows experimental results and then some conclusions are given in the last section.

2. Related works

In synchronous circuit domain, efficient fault simulation methods have been well established as the effects of multiple single-stuck-at faults can be propagated simultaneously from one gate to the next using only local information around the circuit nodes to which the fault effects have propagated [5]. Deniziak [6] presented a high level fault simulator for calculation fault propagation through High Level Primitives (HLPs). Reduced Ordered Ternary Decision Diagrams (ROTDDs) are used to

Manuscript received May 5, 2025
Manuscript revised May 20, 2025
https://doi.org/10.22937/IJCSNS.2025.25.5.16

describe HLPs. They compared this technique with gate level fault simulator and simulation based. These tools capture faulty data at clock intervals, so they are not useful for asynchronous circuits as in this type of circuits there is no global synchronization clock signal.

A.Lioy[8] and et al. in introduce an efficient test generator for asynchronous circuits which is based on a concurrent fault simulator. S.S. Kolay and et al. in [7] introduce Fsimac, a gate-level fault simulator for stuck-at and gate-delay faults in asynchronous sequential circuits.

3. QDI Asynchronous Circuit

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[3][9]. An asynchronous circuit is called delay-insensitive if it preserves its functionality independent of the delays of gates and wires [10]. It is shown that the range of the circuits that can be implemented completely delay-insensitive is very limited [10].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 weak 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 pre-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' Valid '1'	0 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.

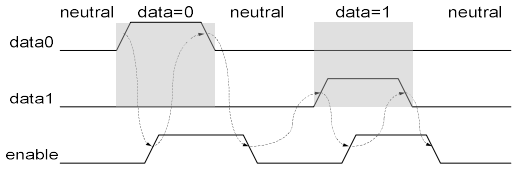

Fig. 1. Four-phase protocol

4. Persia: A QDI Asynchronous Synthesis Tool

Persia is an asynchronous synthesis tool developed for automatic synthesis of QDI asynchronous circuit[4][12]. 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[13], 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.

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.

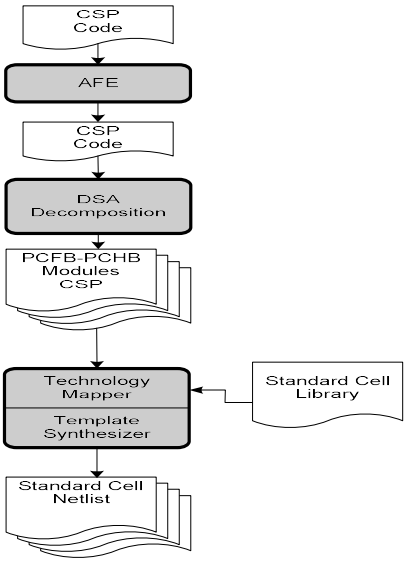

Fig. 2. Persia synthesis flow [4].

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)[12][14].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.

5. PCFB Templates

At present, most QDI synthesis tools like Persia [4] use pre-designed PCHB and PCFB templates to synthesis the high level specifications. 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.

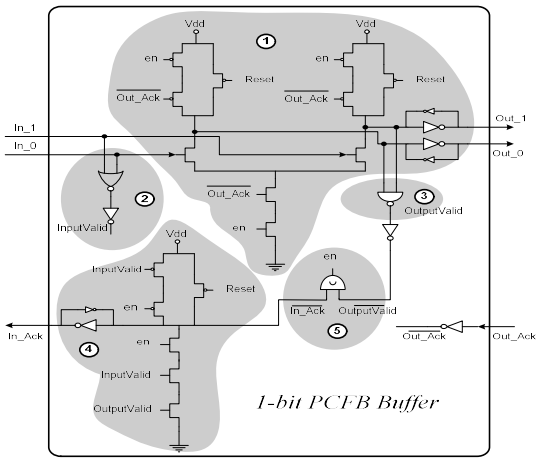

Fig. 3. The PCFB 1-bit buffer

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 [2]. 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 4 shows the necessary pre-defined sequences of PCFB internal signals to have correct operation of synthesized circuit. As seen in figure 4, data process starts by activating Input Valid and finished by activating on en signal.

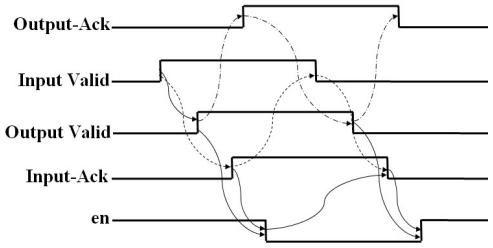

Fig. 4. Sequences of PCFB Signals

6. Fault Simulator

Fault simulation is the process of measuring the quality of a test. Test stimuli that will eventually be applied to the product on a tester are themselves first evaluated by applying them to circuit models that have been slightly altered to imitate the effects of faults. If the response at the circuit outputs, as determined by simulation, differs from the response of the circuit model without the fault, then the fault is detectable by those stimuli. After the process is performed for a sufficient number of modeled faults, an estimate T, called the fault coverage, or test coverage, is computed. The equation is

T = (# faults detected) / (# faults simulated)

The variable T reflects the quality or effectiveness of the test stimuli[17][17]. Several fault simulation algorithms (e.g., Serial, Concurrent, Parallel, etc.) have emerged over the past three decades. In each instance the objective has been to reduce the number of computations and/or memory requirements in order to render the problem tractable. Some differences in approach result from differences in basic assumptions about the circuit being evaluated. When simplifying assumptions are made, it is possible to take advantage of those assumptions to produce a faster product, but one that will not function correctly when those assumptions do not hold. Hence, the user must understand the capabilities and limitations of the tool that he or she chooses to use in order to obtain maximum benefit from it. As mentioned earlier, fault simulation and in general test environments for synchronous circuits are not easily applicable for asynchronous circuits, this is due to deferent synchronization method and as a result deferent fault effect in these two styles of digital circuit design. In the follow we study effect of single stuck-at faults in transistor level for PCFB templates, then we propose efficient fault simulation methodology to circuits which synthesized by this template.

6.1 Single Stuck-at Fault Effects On PCFB Templates

Testing QDI circuits, using the stuck-at model, is thoroughly explored in[15]. 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. To have proper test strategy in PCFB templates, in follow we study effect of premature firing faults in this templates. To have mathematical model for these templates we use production rule, which is an acceptable specification model for QDI asynchronous circuits.

QDI circuits are implemented as a network of gates, where each gate consists of a pull-up network implemented with p-transistors, and a pull-down network implemented with n-transistors. Logically, we can think of a gate as corresponding to two Boolean predicates: G+, the condition that causes its output ν to be connected to the power supply (VDD, interpreted as the logic "true" or 1 value in any Boolean expression), and G−, the condition that causes its output ν to be connected to ground (GND, interpreted as the logic "false" or 0 value in any Boolean expression). We denote this gate using the production rule (PRS) notation [11] as follows:

$$G^{+} \mapsto v\uparrow$$

$$G^{-} \mapsto v\downarrow$$

Using this notation, a two-input NAND gate would be specified as follows:

$$\neg a \vee \neg b \mapsto out\uparrow$$

$$a \wedge b \mapsto out\downarrow$$

Where "Λ" denotes the Boolean AND, "ᵛ" denotes OR, and "¬" denotes logical negation. A restriction on production rules is that both G+ and G− must never be true at the same time, because this would result in a short circuit. This condition is known as non-interference. If G+ and G− are complements of each other, then the gate output is always connected to a power supply. This corresponds to a conventional static CMOS gate and is referred to as a combinational gate. If there is a state when both G+ and G− are false, then in this state the output does not change. If this occurs, then the gate is said to be state-holding. State-holding gates always contain a staticizer (a.k.a. a keeper) on their output to prevent the gate output from changing due to leakage or noise. A fork in a circuit corresponds to an output of a gate being used as the input to more than one gate. Each connection from a gate output to a gate input is referred to as a branch of the fork. We say that a branch of the fork is isochronic if we must make a delay assumption about the relative delay of the branch of the fork relative to the other branches of the same fork (a detailed technical discussion can be found in[16]). As mentioned earlier PCFB templates consist of five sub circuits, production rules of each sub circuit of one bit PCFB buffer is as follow:

1-Output generation sub circuit:

$$In\_1 \wedge \neg Out\_Ack \wedge en \rightarrow Out\_1\uparrow$$
$$(\neg en \wedge Out\_Ack) \vee \mathrm{Re}set \mapsto Out\_1\downarrow$$
$$In\_0 \wedge \neg Out\_Ack \wedge en \rightarrow Out\_0\uparrow$$
$$(\neg en \wedge Out\_Ack) \vee \mathrm{Re}set \mapsto Out\_0\downarrow$$

2- Output_Valid generator sub circuit:

$$Out\_1 \vee Out\_0 \rightarrow Output\_Valid\uparrow$$
$$\neg Out\_1 \wedge \neg Out\_0 \rightarrow Output\_Valid\downarrow$$

3- Input_Valid generator sub circuit:

$$In\_1 \vee In\_0 \rightarrow Input\_Valid\uparrow$$
$$\neg In\_1 \wedge \neg In\_0 \rightarrow Input\_Valid\downarrow$$

4- en generator sub circuit:

$$\neg Output\_Valid \wedge Input\_Ack \rightarrow en\uparrow$$
$$Output\_Valid \wedge InputAck \rightarrow en\uparrow$$

5- InputAck generation sub circuit:

$$en \wedge Input\_Valid \wedge Output\_Valid \rightarrow InputAck\uparrow$$
$$\neg Input\_Valid \wedge \neg en \rightarrow InputAck\downarrow$$

Now for studying premature firing we inject fault to production rule of the template as follow: assume en signal in pull-down network of InputAck generation sub circuit is stuck-at 1, so production rule of InputAck will change as follow:

$$Input\_Valid \wedge Output\_Valid \rightarrow InputAck\uparrow$$
$$\neg Input\_Valid \wedge \neg en \rightarrow InputAck\downarrow$$

It means that if a signal in production rules stack-at 0, in guard which contain faulty signal, that signal will be replaced by 0 and for stack-at 1, it replaced by 1.

To have realistic results we replace $\rightarrow$ symbol in production rule by $\xrightarrow{\nabla}$ which means that delay of firing is arbitrary.

By these modifications we apply all possible single stack-at faults to production rules of the templates, by eliminating faults which cause deadlock, these faults categorized as follow:

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 Output generation sub circuit).

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 Output generation sub circuit).

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 InputAck generation subcircuit)

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. OutputAck stuck-at 1 in the pull-up network of Output generation sub circuit)

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 Output generation sub circuit).

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. As seen above it only can change the number of tokens.

So we can conclude that single stack-at fault in PCFB templates has only one of the three effects: Deadlock, Token dropping or Token generation.

6.2 Fault Simulation Methodology

Based on fault effect on templates, it is concluded that fault change number of tokens, in functionality view. So, to simulating faulty circuit in high level of abstraction, we can add property to each predefined templates of library, which counts number of tokens come in to template and come out from templates. This property compares number of input and output tokens to identify faulty circuit.

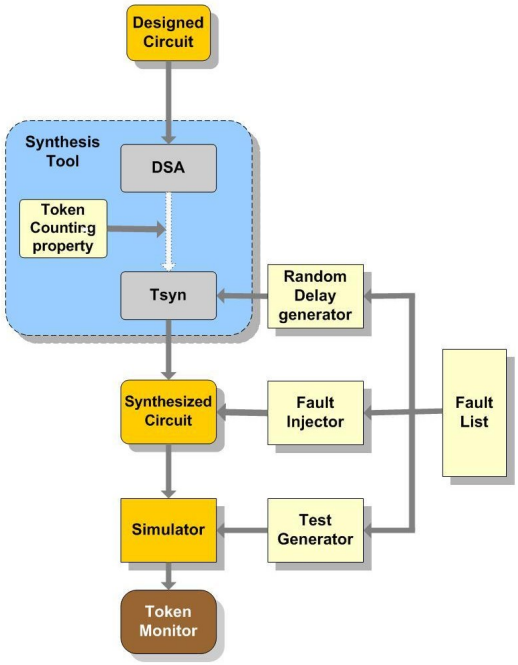

Fig. 5. Proposed Fault simulator

This property added to templates before TSYN stage and after of Decomposition phase of Persia tool. To simulate faulty circuit in worst and best cases in systemization of this circuit, random delays are applied. Test patterns are produced by pseudo-random tests generation approach modification of Cellular Automata [6]. By identifying location of fault in template (which sub circuit of template is under testing) efficiency of test vectors have been increased and time to produce them have been decreased. The test generator eliminates test vectors that are not relevant to the place of fault. Flow of the proposed fault simulator has been shown in figure 5.

7. Experimental results

We inserted token counting property to each pre-designed templates of Persia after decomposition phase of synthesis. To evaluate efficiency of our fault simulator, we implemented primitive gates, listed in table 2. Test vectors for simulating faults were generated as mentioned earlier by modification of CA. Table 2 shows results of the circuits testing.

Table 2: Percentage of fault effects in PCFB primitive gates

	Deadlock	*Token Consume*	*Token Generation*
AND/NAND	75.80%	11.9%	12.3%
XOR/XNOR	75.64%	11.68%	12.68%
OR/NOR	75.48%	11.73	12.79
Buffer/ Inverter	75.22%	12.29%	12.49%

8. Conclusions

Complexity of design and testing are the major obstacle for widespread use of asynchronous circuit in digital circuit design. To overcome design complexity of these circuits, designers can use template based synthesis tools. So it is necessary to develop test frameworks which suitable for this type of circuits. Because of pre-designed property of these circuits we can use some special study on fault effects of templates and categorized them to develop test tools. In this paper we present an efficient high-level fault simulation strategy for template-based QDI asynchronous circuits. We study transistor-level single stuck-at faults in templates and categorize their effects. By use of mathematical specification of circuit, this fault categorization has been proved. This categorization is used to introducing high level fault simulation for this type of circuits. Based on this strategy we develop a fault simulator, Experimental results on a set of circuits have shown the effectiveness of the fault simulator.

References

[1] M.Nystom, E. Ou, A.J.Martin, “An Eight-bit Divider Implemented in Asynchronous Pulse Logic”, Proceeding of the 10th International Symposium on Asynchronous Circuits and Systems(Async’04), IEEE, 2004.

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

[3] D. Koppad, A. Bystrov, A. Yakovlev, “Off-line testing of Asynchronous circuits,” Proc. of the 18th International Conference on VLSI Design, pp. 730 - 735, Jan. 2005.
[4] Persia Site: http://www.async.ir/persia/persia.php
[5] E.W. Thompson and S.A. Szygenda, “Digital logic simulation in a time-based, table-driven environment part 2, parallel fault simulation,” Comput., Vol. 8, pp. 38-49, Mar, 1975.
[6] Deniziak and K. Sapiecha, "FAST HIGH-LEVEL FAULT SIMULATORS", IEEE , 2004
[7] S.Sut-Kolay, M.Roncken, K.Stevens, P.P Chavdhuri, R.Roy, “ Fsimac: A Fault Simulator for Asynchronous Sequential Circuits”, IEEE, 2000.
[8] A.Lioy, F.Maino, G.Odass, M.Poncino, “Testing Hyperactive Faults in Asynchronous Circuits”, IEEE, 1995.
[9] Jens Sparso, Steve Furber, “Principles of Asynchronous Circuit Design-A System
[10] Alian .J. Martin, “The limitation to delay-insensitivity in asynchronous circuits”, In W.J.Dally,ed,Sixth MIT Conference on Advanced Research in VLSI. Cambridge, Mass, MIT Prwss, 1990.
[11] 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.
[12] A. J. Martin, “Synthesis of Asynchronous VLSI Circuits,” Department of Computer Science California Institute of Technology, Pasadena, California, CA, Tech. Rep. TR-A267744, pp. 1-143, MAR 2000.
[13] 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.
[14] M.Najibi, K. Saleh, H.Pedram, "Using Standard ASIC Back-End for QDI Asynchronous Circuits: Dealing with Isochronic Fork Constraint" In Proc. Of GLSVLSI/07,Italy, March 11-13, 2007,
[15] 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.
[16] 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.
[17] Alexander Miczo, "Digital logic testing and simulation", John Wiley & Sons, New Jersey, 2003.

Masoud Zamani was born in Mianeh in Azerbaijan-E-Sharghi of Iran, on 11 Jun, 1983. He received B.S. and M.S. degrees in Computer Engineering from Shahed University and Amirkabir University of Technology in 2005 and 2008, respectively. He is interested in Digital Design, Bio-inspired Design and Test.

Hossein Pedram Received his BS degree from Sharif University in 1977 and MS degree from Ohio State University in 1980, both in Electrical Engineering. He received his PhD degree from Washington State University in 1992 in Computer Engineering.
Dr Pedram Has served as a faculty member in the Computer Engineering Department al Amirkabir University of Technology since 1992. He teaches courses in Computer architecture and distributed systems. His research interests include innovative methods in computer architecture such as asynchronous circuits, management of computer networks, distributed systems, and robotics.

Behnam Ghavami was born in Esfarayen in North Khorasan of Iran, on April 9, 1982. He received his BS degree in Computer Engineering from Bahonar University in 2005. He graduated from the Tehran Polytechnic University. He is a member of Asynchronous Design Laboratory in the same school.

.