Journal of Systems Architecture xxx (2010) xxx–xxx

Contents lists available at ScienceDirect

Journal of Systems Architecture

journal homepage: www.elsevier.com/locate/sysarc

Investigation of transient fault effects in synchronous and asynchronous Network on Chip router

Pooria M. Yaghini, Ashkan Eghbal [*], Hossein Pedram, Hamid Reza Zarandi

Department of Computer Engineering and Information Technology, Amirkabir University of Technology, Tehran, Iran

A R T I C L E   I N F O

Article history:
Received 27 March 2010
Received in revised form 26 July 2010
Accepted 4 October 2010
Available online xxxx

Keywords:
Network on chip router
Asynchronous and synchronous design
Fault tolerance evaluation
Fault injection

A B S T R A C T

This paper presents comparison of transient fault effects in an asynchronous NoC router and a synchronous one. The experiment is based on simulation-based fault injection method to assess the fault-tolerant behavior of both architectures. The effort has been accomplished by employing fault injector signal (FIS) in asynchronous design and synchronous one. Different fault models such as Crosstalk, SEU, and SET have been applied in both architectures to evaluate their robustness. Glitch fault model has also been injected through the asynchronous scheme. The experimental results have been considered in different aspects to estimate the NoC router's robustness. Although asynchronous designs seems inherently fault-tolerant due to applying handshaking signals, up to 55% of the injected faults result in failure, and about 44% of injected faults are replaced by new values before turning into errors. Less than 1% of injected faults treated as latent error. Moreover, the failure rate of token generation is higher than token consumption effects. Furthermore, experiments show that asynchronous NoC router is more robust than the synchronous one by preventing the fault propagation.

1. Introduction

Utilizing more gates on a same silicon die has been pragmatic with the help of developed technology. Data transmission through a chip is becoming more difficult, as the complexity of a system has increased. Employing a new improved communication infrastructure is inevitable to provide a well-developed connectivity among the processing elements (such as CPU, DSP, or memory cell) on a single chip. Network on chip architecture is a practical alternative for traditional System on Chip (SoC) approach, supporting better modularity, scalability, and higher bandwidth [1].

Faulty operation of such interconnections might affect the functionality of connected processor elements. Controlling the physical parameters in the fabrication process is unexpected due to shrinkage of dimension [2,3]. Fault arising such as Crosstalk, Electro Migration (EM), Electromagnetic Interferences (EMI), Alpha particles hits, and also cosmic radiation can affect the functionality of a NoC router or, eventually lead it to failure [4]. The fault tolerance characteristic of a digital system has gotten a matter of considerable concern among designers because of these phenomena.

Based on definitions, a fault-tolerant system should operate correctly in presence of any hardware or software faults. In other words, a fault tolerance design identifies the potential causes of failures and recovers them. This identification is performed by means of hardware, time, or information redundancy.

With this end in view a comprehensive study is necessary to find the most sensitive component of a digital system. This analysis can be accomplished in two different ways which are formal and experimental analysis. In the first method a system is modeled by arithmetical formulas. Applying these sorts of rules, the reliability of a system can be estimated in a shorter time in contrast to second method. The experimental analysis is done by means of fault injection, providing a faulty environment to evaluate robustness of the system. Fault injection is a popular technique in evaluating the dependability attributes of a system [5].

This technique can be implemented in three main strategies consisting of physical, simulation, and software based fault injections. In physical method the original device is applied and infected by faults. Software strategy tries to produce the errors that might happen during its operation. The last method that is considered in this article is simulation-based technique. It is implemented by simulating a system on another computer. It is more popular due to its highest observability and controllability. It can also be used during the primitive steps of a system design, reducing the testing cost. In this method the sensitive parts are observed in a faulty environment [6].

To keep the cost of imposing redundancy low, an exhaustive research seems necessary to find the most tenuous components in a NoC router. A few researches have been performed to estimate the fault-tolerant property of synchronous NoC routers with

* Corresponding author. Tel.: +98 9121976404.
E-mail addresses: pooria.yaghini@aut.ac.ir (P.M. Yaghini), ashkanxy@aut.ac.ir (A. Eghbal), pedram@aut.ac.ir (H. Pedram), h_zarandi@aut.ac.ir (H.R. Zarandi).

1383-7621/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.sysarc.2010.10.003

simulation-based fault injection method. SEU and cross-talk fault effects have been evaluated on a synchronous NoC router [7]. In some cases the reliability of NoC router has been assessed by the help of qualitative methods. These analyses are based on probability equations [8]. However, fault injection with simulation method has been considered on a synchronous NoC router [1]. A similar effort has been accomplished to find the robustness of an asynchronous deign [9]. A simulation based method has been applied to compare the robustness of an asynchronous NoC router and a synchronous one.

Most of previous efforts have introduced reliable synchronous NoC architectures by means of employing fault-tolerant routing algorithms. In some experiences rerouting algorithms have been proposed to update the routing table to make the router more resilient against faults [10–12]. The effect of information redundancy as a fault-tolerant technique, in a NoC architecture has been examined [13]. The idea of bypassing faulty data paths within failed routers has been suggested as a light weight fault tolerant method [14]. Applying the analytical method usually accompanies with some assumption which can affect on result. In [8], an analytical model of a NoC architecture has been presented with the assumption of not being the faulty and destination router at same columns in a mesh topology.

Applying an asynchronous NoC router instead of synchronous one can amend chip characteristics, such as power, performance, delay and also temperature. Eliminating clock in this sort of routers, make them more robust against clock skew phenomenon. Previous research has proved that asynchronous designs are more robust against the EM and also EMI faults in contrast to synchronous ones [15].

The rest of this paper is organized as follows: Section 2 is consisted of an introduction to the asynchronous and synchronous NoC router architectures which are employed through this experiment. Part 3 includes the fault concepts and terminologies. Injected fault models in this experiment have been introduced in Section 4. Fault injections framework is illustrated in part 5. Section 6 contains the result of fault injection experiments in different aspects. Finally, Section 7 presents some conclusion remarks.

2. Network on chip router architectures

2.1. Synchronous NoC router architecture

Fig. 1 indicates the architecture of a synchronous NoC router. It is implemented with five bidirectional ports, which is apt to be employed in the Mesh and the Torus topology. It consists of three main components including Buffer, Routing unit, and Crossbar switch. The state diagrams in Routing unit and Buffer illustrate the functionality of them. The Buffer component is sensitive to positive edge of clock. The routing process is accomplished as negative edge of clock has been found in Routing unit component and a new header flit is received.

Routing unit component would be aware of upcoming trailer flit by the help of Buffer component. Such a signal makes the Routing unit cease granting the selected output switch to transfer any more flits after the trailer flit has passed. The Buffer is implemented as a circular queue to optimize applying buffer efficiently. It also contains a buffer management sub-component which is in charge of controlling the empty and full spaces in order to preclude replacing the old flits by the new ones before transmission.

The Routing unit component is the central unit including sub-components such as header extractor, header processor, arbitration unit, and routing table. XY routing algorithm is implemented by the help of such component. Each of input channels can reserve one of the output ports when routing process has been accomplished. An arbitration unit locks the dedicated output channel until the end of packet transmission. It also shares a specific output channel fairly among input ports, requesting for it. Routing unit grants the requested input port and enables the selected output multiplexer to make them connected once the routing process has been accomplished successfully. Such a *grant* and *activation* signals are disabled as the trailer flit of a packet is transmitted to its desired output port.

The third component is a Crossbar switch. The control signals of Routing unit select one of output ports for an incoming header flit. By implementing the wormhole switching, all of the remaining flits of a packet follow the header flit in a pipeline manner. They are blocked if their header is impeded on its way toward destination. Once a packet transmission has finished, the switch is unlock to serve other input channels.

2.2. Asynchronous NoC router architecture

Fig. 2 indicates the architecture of the asynchronous NoC router which has been simulated very similar to the synchronous one to have more accurate comparison between them. The asynchronous router is implemented with five bidirectional ports, which are suitable to be employed in torus topology. There is no clock in this design and all of the data transmissions are put into action by the help of handshaking signals. A four-phased handshake protocol has been employed in this router.

In a four-phased communication protocol, a receive process consists of four following steps: (1) Wait for input to become valid. (2) Acknowledge the sender the transmission has been accomplished. (3) Wait for inputs to become neutral. (4) Make the acknowledge signal low. A send activity contains of four subsequent phases: (1) Send a valid output. (2) Wait for acknowledge. (3) Make the output neutral. (4) Wait for acknowledge to lower output. A dual rail encoding has been employed in such implementation in which the data channel contains a valid data (token) when just one of two wires is high. When the two wires are lowered the channel contains no valid data and is called to be neutral. Fig. 3 indicates such concepts in more details.

The asynchronous router like the synchronous router consists of three main modules which are Input buffer, Crossbar switch, and Routing unit.

Input buffer is responsible to store temporarily incoming flits from adjacent routers if it has free space. The capacity of Input buffer in all five ports is five flits (one buffer is dedicated per port). It is implemented by means of shift register to have better performance and lower area consumption. A counter has been dedicated to each input buffer to determine the number of received flits of current packet. This counter is accessed by a sub-component called header extractor which distinguishes the header flit among others in a packet. Header processor is another sub-component of routing unit which is responsible to process the incoming header of packet.

Routing unit is the main component to implement XY routing algorithm. Each router includes its own coordinates through X-ID and Y-ID variables. Routing process is implemented in this component once a new packet has been received. Each input channel can reserve one of the output ports when routing process has been accomplished. An arbitration unit is utilized to dedicate the output port with round robin algorithm if there is more than one request for a specific output channel. In other words, by the help of arbitration sub-component output ports are distributed fairly among the input channels. As soon as routing table set the switch component, the input port is connected to appropriate output channel with the help of asynchronous method. Such a component is activated once a time for each packet to make the routing decision.

The last component, crossbar switch, is in charge of connecting all input ports to the suitable output ones. The control signals of

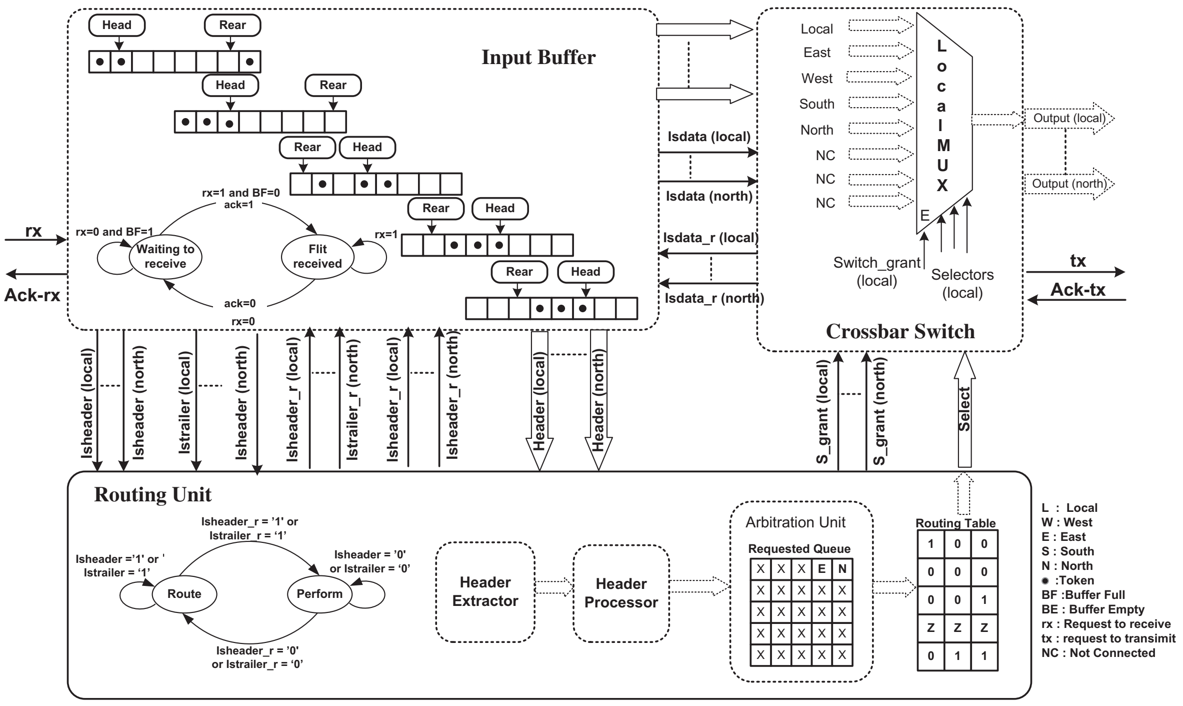


Fig. 1. Synchronous Network on Chip router architecture.

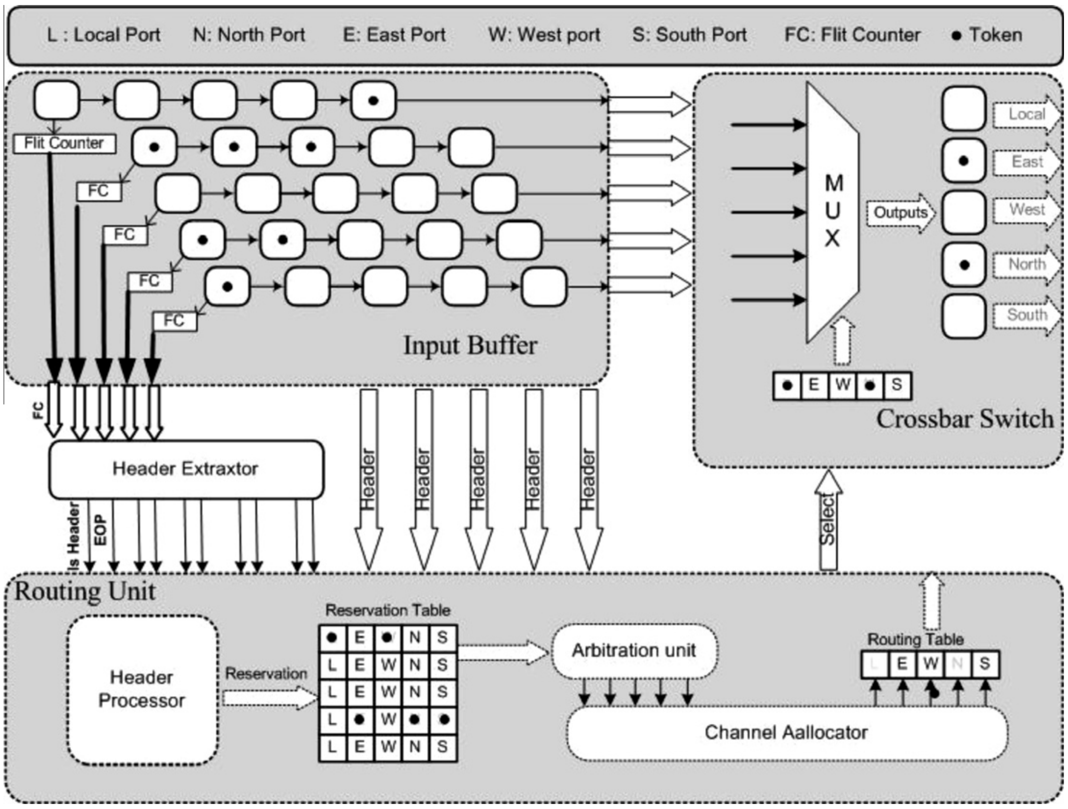
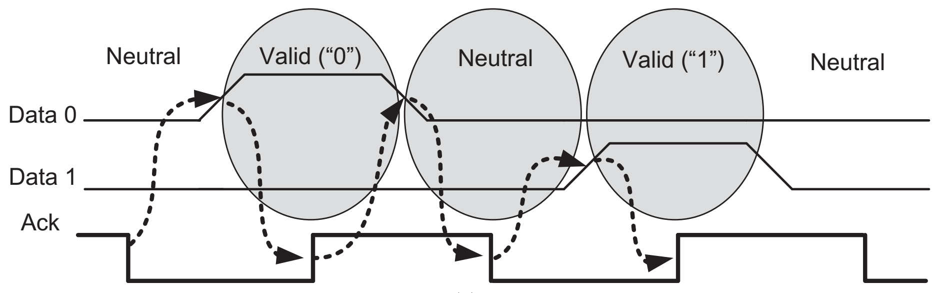


Fig. 2. Asynchronous Network on Chip router architecture.

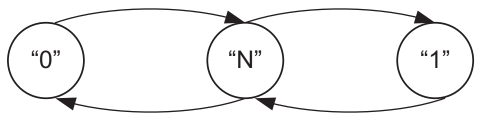
routing unit select one of the output ports for an incoming header flit simultaneously. By implementing the wormhole switching all of the remaining flits of a packet follow their header or they are blocked is their header is blocked on its way toward destination. Eliminating the clock skew problem, modularity, lower power consumption, and applying average delay instead of worst case delay, are some of asynchronous design benefits. The modularity of an asynchronous scheme makes it adaptable to future technologies and less vulnerable against changes of voltages or other environmental conditions such as temperature.



(a)

	data 0	data 1
Neutral “N”	0	0
Valid “0”	0	1
Valid “1”	1	0
Not allowed	1	1

(b)



(c)

Fig. 3. Four phase handshaking protocol (a) with dual rail coding (b) and RTZ sequence (c).

The router is implemented with CSP-Verilog (communicating sequential processes) language as an asynchronous design. CSP is a well-known language for description of concurrent systems which is accepted as a good description language for asynchronous systems [16]. A circuit in CSP is described as the composition of distinct processes that run in parallel and communicate with each other on channels by message passing. CSP-Verilog is an extension of the standard Verilog language which supports asynchronous communication as the hardware description language for all levels of abstractions except the netlist which uses standard Verilog.

CSP-Verilog is equipped with *READ* and *WRITE* programming language interface (PLI) macros to emulate CSP language communication actions on the channels. In order to send data, *Write* macro is employed like: ‘*WRITE* (*Port* name, value)’. If the sender wants to write another data on such port, it would be suspended until the previous one is read with the following command ‘*READ* (*Port* name, value)’. The receiver module’s port is also suspended until a data is written it. Read and write operation will be implemented by 4-phase handshake signaling [16].

3. Fundamental concepts of faults

A fault can be defined as a physical defect, imperfection, or flaw that occurs within some hardware or software component. An error is the manifestation of a fault especially, a deviation from a normal behavior. The error has turned into a system failure as the functionality of a system performs abnormally [17].

Evaluating the dependability of a system has become convoluted as the complexity of a system grows. Experimental or analytical methods are two main techniques for achieving such an assessment. Considering the fault effects and error propagation of an actual system is not an easy task in analytical method. Furthermore, the given assumptions in analytical technique have reduced the accuracy of the evaluating results. Fault injection has become popular in assessing the fault-tolerant behavior of a system although it takes longer to be accomplished [18,19]. This method can be implemented in three different approaches, physical, software, and simulation [20].

Physical fault injection stresses the hardware with environmental parameters or by modification of pin-level values. This technique needs a real instance of system, so it cannot be employed before the fabrication process is done. It cannot be utilized until a real instance of system is available. It is just employed to examine some special fault models effects such as EM, and EMI. Additional software is applied to inject faults into the system in software-based methods. This technique consists of reproducing the errors that would have been produced due to fault emergence at software level. It provides a cheaper and more flexible fault injection environment than previous method. Simulation-based is an admired method due to its full observability and controllability property. Evaluating the behavior of a system in designing steps and before fabrication process is the other advantages of it. In this method the model of a system is simulated in another system and the logical values are infected during simulation.

This experiment is based on simulation-based fault injection technique by the help of SINJECT tool [21]. It evaluates the fault-tolerant behavior of any modeled system in VHDL or Verilog languages in three different phases consisting of injection, simulation, and evaluation. Fault injection is accomplished by employing FIS (fault injector signals). Activating these signals forces the system to operate in a faulty environment. Similarly, deactivating the FIS signals makes the system’s running in a faultless environment.

According to the basic definition, faults are divided into two classes, propagated and not-propagated. In most of evaluations the propagated ones, real faults, are considered [17,20,21]. Following such a fact, the real faults have been taken into account in this article. The propagated faults can be partitioned into two subclasses, *Latent* and *Active*. *Active* faults relates to some which have been detected. *Latent* ones are pertinent to some sorts of fault, existing in the system. However, they have not been detected yet. The main property of *Latent* faults is that they remain in the system, and they might turn into *Active* ones once they have been detected. Some of faults might be overwritten by new values while they are *Latent*. A system never suffers from any replaced faults with new values since they are not able to be *Active* at all. There is a same story for errors and propagated errors. They can be divided into *latent* or *active* ones.

4. Fault models

The importance of transient faults is a matter of considerable concern comparing to permanent ones in digital systems. The frequency of transient faults is expected to increase in future systems due to technology scaling [22]. This experiment examines the effects of transient faults rather than permanent ones.

Different fault models have been injected into the VHDL and CSP-Verilog model of the synchronous and asynchronous NoC routers to compare the robustness of both deigns. SEU (Single Event Upset), SET (Single Event Transient), and Crosstalk fault models are considered in this experiment, which are well known faults among designers. Following paragraphs have given a brief description of them.

- *SEU* is a change of state caused by ions or electro-magnetic radiation striking a sensitive node in a micro-electronic device. These phenomena can affect on the behavior of sequential circuits like memory cells, register files, pipeline flip-flops, and also caches. These effects can invert the logical value of each of them. The SEU type can be modeled by inverting the statue of logical values in memory components randomly.
- *SET* is also a temporary variation in the output voltage or current of a circuit due to the passage of a heavy ion through a sensitive device would results in a SET. However, the SET faults are considered through combinational circuits. SINJECT supports different fault models to simulate the SET fault by means of Dead-clause, Stuck-then, and Micro-operation [21].
- *Crosstalk* is a phenomenon by which a signal transmitted on one circuit of a transmission system creates an undesired effect in another circuit or channel. SINJECT is capable of modeling Crosstalk fault model. According part both of synchronous and asynchronous design employs some control and data links which might be suffer from any sorts of Crosstalk fault models.
- *Glitch* is an undesired transition that occurs before the signal settles to its intended value. In other words, Glitch is an electrical pulse of short duration that is usually the result of a fault or design error, particularly in a digital circuit. Such a fault model result in Token Consumption and Token Generation effects in asynchronous designs. This fault model has been considered through the asynchronous NoC router.

Comparing the effects of similar fault models would assess the fault tolerance behavior of a synchronous design and an asynchronous NoC router architecture.

5. Fault injection framework

The importance and frequency of transient faults are expected to increase in future systems to technology scaling [22]. This article has considered the effects of transient faults to compare the robustness of a synchronous NoC router and the asynchronous design with the help of SINJECT fault injector. In SINJECT, the first step of a fault injection is the insertion of the FIUs (Fault Injection Units) into the VHDL (or Verilog) description of the target system. FIUs are units, which are added to the target system, in order to inject faults to it. Each FIU is activated by a fault injector signal (FIS). When an FIS takes the value 1, its corresponding FIU injects a fault to its target. SINJECT simulate transient faults by defining a life time for each FIS. Such a life span has been selected considering the required time for a router to receive, route and transfer a flit in a normal condition. The injected faults treat as a permanent one if such a life time equals the simulation time. One hundred and twelve targets of a synchronous NoC router are infected with the FISes, while 136 points are infected in the asynchronous design. The number of selected targets is based on the complexity of each design and their comprised components. More faults have been injected through the asynchronous scheme as it contains more handshaking signals and complicated circuits. Repeating each experiment for 100 times, result in 11,200 and 13,600 fault injections into the synchronous and asynchronous designs, respectively. The effects of single faults are taken into account in this experiment. In other words, the FISs are activated separately to observe their effects without any timing overlap.

In order to have a concise fault injection, we need to know the lifetime of activated transient faults, known as fault duration. In synchronous designs, the clock signal is used to control the fault duration, however, as there is no clock signal in any asynchronous designs; to find a specific lifetime for injected faults, flit delay estimation in asynchronous network is necessary. The requisite estimation for proposed asynchronous NoC router is as follows:

Average latency of NoC router for performing the receiving and forwarding of a flit through a single router can be estimated by Eq. (1). This calculation assumes the asynchronous NoC router is employed in a torus topology without considering the routing latency.

$$\textit{Router Latency}_{\text{avg}} = (m+1)d \tag{1}$$

Where m is the input buffer size in flit, and d is the delay of one read/write operation. By taking into consideration the routing latency that occurs for each packet (a packet contains n flits), the average router latency for one flit can be modified as Eq. (2).

$$\textit{Router Latency}_{\text{avg}} = (m+1)d + \frac{r}{n} \tag{2}$$

Where r is the routing latency and n is the packet size in flit. Considering torus topology, the maximum diameter in the network is obtained in the following formula as Eq. (3).

$$\textit{Network Diameter}_{\max} = 2\left|\frac{dim}{2}\right| \tag{3}$$

Where dim is the network dimension. The maximum latency of one flit crossing the entire network is thus:

$$\textit{Network Diameter}_{\text{avg}} = \left\{(m+1)d + \frac{r}{n}\right\}\left(2\left|\frac{dim}{2}\right|\right) \tag{4}$$

The number and location of injected fault models through both the synchronous and asynchronous designs are depicted by Table 1 in details. More faults have been injected into the components of the asynchronous scheme except in Buffer. The reason is behind the fact of difference in Buffer implementation in each design. According to part 2, the Buffer component in asynchronous scheme is simulated by means of shift register, while the synchronous design employs the arrays and decoders. Employing shift register in asynchronous buffer implementation results in lower complexity; fewer numbers of faults are injected into this component. SET fault model is not considered in buffer component as it does not have any decision making sub-component. SEU faults are not considered in switch component since it consists of combinational circuits which are not affected by SEU fault models. SEU faults are injected into Routing unit and Buffer components through their sequential circuits, as shown in Table 1. Crosstalk fault models are injected through data and control links, among three main components. Glitch fault model is only injected through the asynchronous NoC router. It might lead to Token Consumption or Token Generation fault effects (which are explained in part 6) in any asynchronous design.

Table 1
The characteristics of injected faults.

	Buffer		Routing unit		Switch	
	Sync	Async	Sync	Async	Sync	Async
SET	14	–	19	10	12	6
SEU	18	12	18	35	–	–
Crosstalk	9	8	10	6	12	20
Glitch	–	9	–	12	–	18
Total	41	29	47	63	24	44

Sync: Synchronous; Async: Asynchronous.

Table 2
Results of fault injection experiments.

	Synchronous						Asynchronous					
	Replaced faults		Latent errors		Failure experiments		Replaced faults		Latent errors		Failure experiments	
	#	%	#	%	#	%	#	%	#	%	#	%
Input buffer	1622	52	63	2	1435	46	203	33.8	1	0.2	400	66
Routing unit	1105	39.2	23	0.8	1707	60	201	20.1	2	0.2	800	79.7
Switch	808	44.1	17	0.9	1022	55	633	80.6	1	0.1	151	19.2
NoC router	3535	45.7	103	1.3	4164	53	1037	44.84	4	0.16	1351	55

A comparable fault injection study has been done on a similar synchronous architecture [1]. The new synchronous router architecture has been modified to make it more similar to asynchronous architecture. Buffer component regulates itself in such a new design, while it was managed by central controller in previous approach. Handshake signals such as *tx*, *ack_tx*, *rx*, and *ack_rx* are not either controlled by the central controller anymore. They are managed by Buffer and Switch components as shown in Fig. 1. Some new control signals have been added to make different components be aware of the status of each other such as *switch grant*, and *is-data*. A new fault injection study has been accomplished according to the architectural amendments.

6. Experimental results

Table 2 compares the number and percentage of *Replaced* faults (which has been overwritten), *Latent* errors, *Failure* experiments, and also the average of fault latency for each of component separately. These results are based on comparison of the results from golden run (faultless run) and faulty run for each FIS. According to the result, the synchronous NoC router and the asynchronous behave in a similar way in terms of failure experiments, latent error, and replaced faults. However, their components treat different from each other. The failure rate of switch component is in asynchronous design is not as much as the same component in the synchronous scheme. The reason of such a difference is that the switch is responsible for connecting the input channel to the selected output one without any decision making process. The asynchronous switch prevents fault propagation due to handshaking signals, resulting in high ratio of replaced faults in this component. The routing unit of asynchronous design is more complicated in contrast to the same component in synchronous scheme; it has gotten higher failure rate in contrast to the similar component in the synchronous scheme. The difference in buffer component implementation in both design leads to higher failure rate in asynchronous one. Any fault occurrence in a shift register would propagate through the succeeding ones, resulting in lower replaced fault ration in Buffer component of asynchronous architecture.

Table 3
Failure rate of token consumption and token generation fault effects.

	Token consumption (%)	Token generation (%)
Input buffer	7.47	22.41
Routing unit	3.33	5.40
Crossbar switch	1.66	9.62
Total	12.47	37.44

The injected fault types in this experiment are apt to turn into two fault effects (error) related to asynchronous circuits, including Token Consumption and Token Generation. These fault effects are presented due to the inherent characteristics of handshaking signals in asynchronous designs. Fault occurrence such as SET or crosstalk affecting handshaking links or their sources may result in token (flit) consumption or generating useless tokens (flits).

If more than one output token (flit) is produced for one input token a token generation fault effect has been occurred. Token will be dropped if the sequence of *Output-Valid* and *Output-Ack* alters. It can be taken place as one of following scenarios. If negative edge of *Output-Valid* takes place before the positive edge of *Output-Ack* or when positive edge of *Output-Valid* takes place when *Output-Ack* is 1.

The failure rate of each component caused by token consumption and token generation fault effects are shown in Table 3. Multiplying the number of handshaking signals in a packet transmission period by the fraction of states which lead the router to failure, result in these failure rates. According to Table 3 the failure rate of token generation is higher than token consumption effects.

According to the importance of transient faults in today's digital environment, the effects of such faults have been considered in this effort. Fig. 4 compares the percentage of fault propagation and also failure rate of three transient injected fault models in both synchronous and asynchronous designs. The percentage of fault propagation in synchronous design is higher considerably rather than

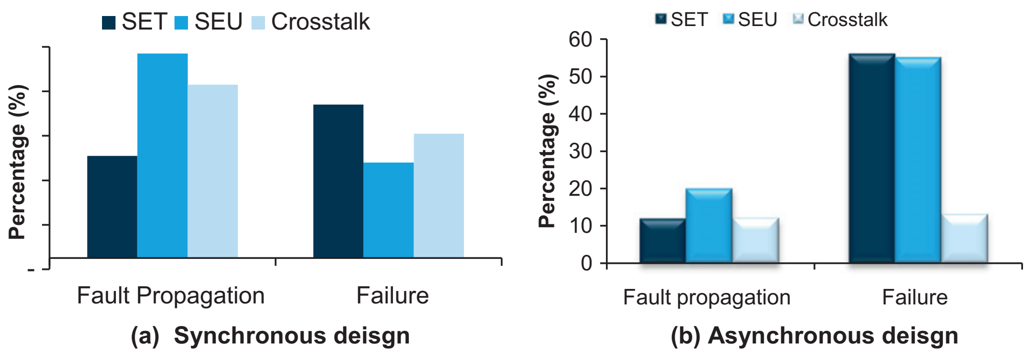


Fig. 4. Transient fault models propagation and failure.

Table 4
Failure rate of transient fault models.

	Asynchronous			Synchronous		
	SET	SEU	Crosstalk	SET	SEU	Crosstalk
Buffer	–	21	1	80	24	69
Switch	30	–	10	63	–	50
Routing unit	2	20	16	63.2	63.1	52.5

asynchronous one. That might be because of handshaking signals of asynchronous scheme, preventing fault propagation. According to these experimental results, an asynchronous NoC router is more robust comparing to the synchronous one. However, the high percentage of failure rate of SET fault models is important in asynchronous design. Although fault propagation ratio in SET and Crosstalk fault models is the same, the failure percentage of SET model is almost 4–5 times bigger than Crosstalk fault model. In order to get more accurate evaluation of different components of the asynchronous NoC router, the transient fault effects have been examined in each component.

Table 4 compares the failure rate of each similar transient faults in both synchronous and asynchronous designs. Buffer is infected by SEU and Crosstalk transient faults. Based on the results the failure rate caused by Crosstalk fault model in this component is less than 1%. On the other hand, propagated SEU fault models lead to failure rate of 30% in asynchronous scheme. But the experiments of synchronous design are totally different. It shows that SET fault model has a great influence over the failure rate among all of components of synchronous architecture. The high failure rate of SEU fault model in Routing unit component is also tangible. There is equilibrium between the effects of SET and Crosstalk faults in Switch component. Crosstalk fault models influence the functionality of Buffer component dramatically.

Such statistical information shows that the effects of SET faults are more tangible in both synchronous and asynchronous deigns. However, employing an SEU-tolerant method might be wise enough especially in synchronous NoC router. The obtained results are useful for designer to choose a best fault-tolerant approach for each of components separately. Following the experimental results of this effort may lead us in employing an optimized fault-tolerant mechanism wisely.

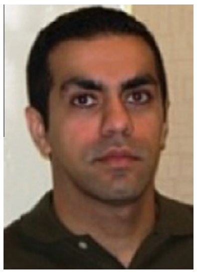
7. Conclusion remarks

In this paper, transient fault effect analysis is performed and compared between an asynchronous and a synchronous NoC router designs. To do this, a delay model was driven to estimate the maximum latency of the entire network provided that the asynchronous NoC router is utilized in a torus topology. Such delay estimation has been considered to find the lifetime of injected faults in fault injection experiments. The experiment results showed that asynchronous design eliminates the effects of clock skew problem and is more robust than synchronous one. Routing unit has been detected as the most vulnerable component in both synchronous and asynchronous designs. In addition, SET and SEU fault models have been detected as the most important faults among transient injected faults in both designs, so that about more than half of the propagated SET faults have caused system failures. The fault injection results showed asynchronous NoC router behaves more robust than synchronous one by preventing the fault propagation through the circuit, and is recommended to be used in fault-tolerant applications.

References

[1] A. Eghbal, P.M. Yaghni, H. Pedram, H.R. Zarandi, Fault injection-based evaluation of a synchronous NoC router, 15th IEEE International On-Line Testing Symposium (2009) 212–214.
[2] J.S. Murali, T. Theocharides, N. Vijaykrishnan, M.J. Irwin, L. Benini, G.D. Micheli, Analysis of error recovery schemes for networks on chips, IEE Design & Test of Computers 22 (5) (2005) 434–442.
[3] A. Acquaviva, A. Bogliolo, A bottom-up approach to on-chip signal integrity, Lecture Notes in Computer Science 2799 (2003) 540–549.
[4] C. Grecu, L. Anghel, P.P. Pande, A. Ivanov, R. Saleh, Essential fault-tolerance metrics for NoC infrastructures, 13th IEEE International On-Line Testing Symposium (2007) 37–42.
[5] Nakamura, K. Hiraki, Highly fault-tolerant FPGA processor by degrading strategy, Pacific Rim International Symposium on Dependable Computing (2002) 75–78.
[6] D. Gil, J. Garcia, J.C. Baraza, P.J. Gil, A. Study, 6th IEEE of the effects of transient fault injection into the VHDL model of a fault-tolerant microcomputer system, International On-Line Testing Symposium (2000) 73–79.
[7] A.P. Frantz, M. Cassel, F.L. Kastensmidt, E. Cota, L. Carro, Crosstalk and SEU aware networks on chips, IEEE Design and Test of Computers (2007) 340–350.
[8] A. Dalirsani, M. Hosseinabady, Z. Navabi, An analytical model for reliability evaluation of NoC architectures, 13th IEEE International On-Line Testing Symposium (2007) 49–56.
[9] P. M Yaghini, A. Eghbal, H. Pedram, fault injection-based evaluation of transient faults in an asynchronous NoC router, 18th Euromicro International Conference on Parallel, Distributed and Network-Based Computing (2010) 540–545.
[10] M. Ali, M. Welzl, S. Hessler, S. Hellebrand, An efficient fault-tolerant mechanis to deal with permanent and transient failures in a network on chip, International Journal of High Performance System Architecture 1 (2) (2007) 113–123.
[11] P. Rantala, T Lehtonen, J. Isoaho, J. Plosila, Fault-tolerant routing approach for reconfigurable Networks-on-Chip, International Symposium on System-on-Chip (2006) 1–4.
[12] M. Pirretti, M.G. Link, R.R. Brooks, N. Vijaykrishnan, M. Kandemir, M.J. Irwin, Fault tolerant algorithms for Network-On-Chip interconnect, Proceedings of the IEEE Computer Society Annual Symposium on VLSI (2004) 46–51.
[13] A.P. Frantz, F.L. Kastensmidt, L. Carro, E. Cota, Dependable Network-on-Chip router able to simultaneously tolerate soft errors and crosstalk, IEEE International Test Conference (2006) 1–9.
[14] M. Koibuchi, H. Matsutani, H. Amano, T.M. Pinkston, A lightweight fault-tolerant mechanism for Networks-on-Chip, Second ACM/IEEE International Symposium Networks-on-Chip (2008) 13–22.
[15] J.F. Chappel, S.G. Zaky, EMI effects and timing design for increased reliability in digital systems, IEEE Transactions on Circuits and System (1997) 30–142.
[16] A. Seifhashemi, H. Pedram, Verilog HDL, Powered by PLI: a suitable framework for describing and modeling asynchronous circuits at all levels of abstraction, Proceedings of IEEE 40th Deisgn Automation Conference (2003) 330–333.
[17] A. Avizienisv, J.C. Laprie, B. Randell, C. Landwehr, Basic concepts and taxonomy of dependable and secure computing, IEEE Transactions on Dependable and Secure Computing (2004) 11–33.
[18] A. Eghbal, H.R. Zarandi, P.M. Yaghini, Fault tolerance assessment of PIC microcontroller based on fault injection, 10th Latin American Test Workshop (LATW'09) (2009) 1–6.
[19] P.M. Yaghini, H.R. Zarandi, A. Eghbal, A. Jafarzadeh, S. Eskandari, An Investigation Of Fault Tolerance Behavior Of 32-bit DLX processor, The Second International Conference on Dependability (DEPEND 09) (2009) 93–98.
[20] J. Aidemark, J. Vinter, P. Folkesson, J. Karlsson, GOOFI: generic object-oriented fault injection tool, International Conference on Dependable Systems and Networks (2001) 83–88.
[21] H.R. Zarandi, S.G. Miremadi, A. Ejlali, Dependability analysis using a fault injection tool based on synthesizability of HDL models, Proceedings of the International Symposium on Defect and Fault Tolerance in VLSI Systems (2003) 485–492.
[22] R.C. Baumann, soft errors in commercial integrated circuits, International Journal of High Speed Electronics and Systems 14 (2) (2004) 299–309.

Pooria Mohammadi Yaghini was born in Tehran (Capital of Iran) in 1984. He has received his Bachelor of Science in Computer Hardware Engineering from Sadjad Institute of Higher Education in 2006 and his Master Degree in Computer Architecture field from the Amirkabir University of Technology in 2009. He has joined to the Design and Fabrication of a Customized Testable FPGA Project team since 2010.



Ashkan Eghbal was born in 1983 in Tehran (Capital of Iran). He has been graduated as a Bachelor with the major of Computer Hardware Engineering from Azad University Central Tehran Branch in 2007. He has received his Master of Computer Architecture from the Amirkabir University of Tehnology in 2009. He has been admitted as a Ph.D. student in University of California Irvine since 2010 in the field of Computer System and Software.

Hossein Pedram has received his B.S. and M.S. degrees in Electrical Engineering from Sharif university of Technology and Ohio State University in 1977 and 1980, respectively. He has also received his Ph.D. degree from Washington State University in 1992. He has joined Amirkabir University of Technology as a faculty member in 1992. He has been selected as a Chairman of Computer Engineering Department, in Amirkabir University of Technology since 2008.

Hamid Reza Zarandi has received his B.S., M.S., and Ph.D. degrees in Computer Engineering from the Sharif University of Technology, Tehran, Iran, in 2000, and 2002, and 2007, respectively. He has joined Amirkabir University of Technology as a faculty member since 2007.