Online Fault Detection for Networks-on-Chip Interconnect

Online Fault Detection for Networks-on-Chip Interconnect

Junxiu Liu, Jim Harkin, Yuhua Li and Liam Maguire
School of Computing and Intelligent Systems,
University of Ulster, Magee Campus,
Derry, Northern Ireland, UK
liu-j4@email.ulster.ac.uk

***Abstract*— A key requirement for modern Networks-on-Chip (NoC) is the ability to detect and diagnose faults and failures. A novel approach is proposed which addresses the challenge of fault detection using an online mechanism. The approach minimises online intrusion by employing dynamic rates of testing to maximize NoC throughput while still ensuring sufficient testing. This is achieved using a novel Monitor Module based on the back-off algorithm. The paper presents results on the minimal impact on the intrusion of the NoC for a range of test conditions and also highlights the low area/power overheads for scalability.**

Keywords*— *Networks-on-Chip; Fault detection; Online testing; Low area/power overhead.

I. INTRODUCTION

The NoC was first introduced in [1] and is composed of routers, channels and processing elements. The concept is similar to traditional computer networks where packets of information are communicated via paths across several routers, and routers define the path between the source and destination processing elements. NoCs are an active research field and widely used by many industrial applications [2]. Despite considerable NoC research, many issues remain including fault-tolerance, Quality of service policies, traffic modelling, latency and power minimization [3], [4]. According to ITRS, during operational lifetime, 1% of chips experience a fault per day, and in the near future, the manufacturing defect rate will reach the level of approximately 1,000 defects/m^2 [5], [6]. Therefore fault tolerance is a significant challenge for modern NoCs due to the increase in physical defects in advanced silicon manufacturing processes.

To ensure modern NoCs are able to accommodate faults and maintain operation post-manufacturing requires the development of autonomous systems that can self-detect and self-repair without human intervention. The fault detection of NoC systems is generally divided into the testing of processing elements, routers and interconnects [7]. For example, the processing elements can be tested by re-using the NoC as a test access mechanism [8]; scalable partial scan and test wrapper techniques were used to test the NoC routers [9] and the interconnect faults can be detected by a built-in self-test scheme [10], a broad detection strategy [11] or a token-based fault detection mechanism [12]. The repair techniques include a dynamically reconfigurable routing algorithm [13], adaptive routing algorithm (e.g. HLAFT [14], FTDR and FTDR-H [15]) and data-splitting strategy [16] to enhance the system fault-tolerance ability. The proposed work in this paper focuses on interconnect fault detection as modern NoCs now consume a significant level of resources, and therefore is now viewed as a major system component which is susceptible to faults. The output from the proposed work can be employed with an adaptive routing algorithm to provide a 'corrective' mechanism when faults are detected. Interconnect faults can cause loss of packets or even router crashes [17]. Faults that occur at the interconnect are mainly stuck-at, bridging, crosstalk, single event upsets and single event transient faults. Stuck-at faults are widely detected in digital circuits and can be divided into stuck-open and stuck-on faults, which result in non-conductive and conductive logic states respectively. Bridging faults can be defined as a short between two lines, which are otherwise unconnected. Crosstalk faults are also common in NoC interconnect as channel lines are placed physically close to each other, increasing the likelihood that logic signalling on one line can affect the logic values on neighbouring lines. The stuck-at, bridging and crosstalk faults that frequently occur in the NoC interconnect, pose a significant challenge in the design of reliable systems.

Typical in-field test techniques simply stop an executing application and test for possible failures. This is not permissible for mission critical systems as they cannot be interrupted. The ideal solution is an online fault detection mechanism however two key issues must be addressed: (1) test intrusion – detection should be minimal and not reduce the system performance and (2) test frequency – the level of testing should be balanced to ensure minimal dynamic power consumption for scalability. Testing with a high frequency provides early fault detection but introduces high power consumption. However, infrequent testing is not able to detect potential faults promptly. The area/power overheads are key impact factors in the scalability of NoC systems. Therefore, this paper proposes a novel online test mechanism for NoC interconnect based on the back-off algorithm, which provides a balance between frequent and infrequent testing while maintaining a minimal area/power cost. The approach supports the detection of both permanents and intermittent faults. The paper is organised as follows: fault testing/monitoring schemes and strategies in electronic systems are reviewed in section 2 with a focus on NoC. Section 3 discusses the proposed novel fault Monitor Module and online testing strategy. Section 4 presents results and performance analysis for a range of experiments, and section 5 provides a conclusion and highlights future work.

II. Motivation and Previous Work

Several methods can be used to enhance the capability of NoCs to detect faults. A fault test mechanism using test data generators and test error detectors has been explored in [10], [18] for NoC interconnect fault testing. Self-tests were implemented by transporting the testing data packets to the components. A router test was also integrated in [7], including the test of a router's internal blocks, the control logic and FIFOs. These approaches provide good methods of detecting channel faults, however, all the nodes are in functional mode which does not accommodate online tests and thus increases boot up time. Such offline testing approaches have a significant impact on the NoC as individual routers are pulled offline for testing and can no longer transmit data during the period of testing. In some computing systems, there is often a long period of time after power-up which cannot be pulled offline or interrupted, such as mission critical electronic systems. A distributed on-line test method was proposed in [12] to detect stuck-at and short wire faults in NoCs. In this approach, a token-based technique was used to put only one router in test mode while the others were in operational mode. However, the nodes are tested sequentially, implying long test times which do not support real-time detection. The online fault detection approach in [19] utilises several specialised nodes in a NoC called Test Infrastructure (TI) IP, which delivers test vectors to a normal node or active BIST component to detect the router fault. A node monitoring scheme must be provided in the system snapshot determination phase to aid in managing the test intrusion. The main drawback is that this testing strategy is not fully distributed and the entire testing procedure is terminated if TI-IP is faulty.

For an autonomous system that can self-detect, several key functions need to be investigated: (1) the provision of online detection with minimal intrusions and (2) support for scalability via area/power-efficient fault detection mechanisms. Current approaches have the aforementioned weaknesses and do not fully address these requirements. This paper proposes such a fault detection approach. In this paper, a novel monitor module (MM) based on the back-off algorithm is proposed to achieve dynamic testing. The key advantages of using dynamic rates of testing include: (1) minimization of online intrusion from testing and (2) maximising NoC throughput while still ensuring sufficient testing is performed. The paper also highlights the key low area/power overheads of the MM in meeting scalability requirements.

III. Online Test Methodology for NoC

This section presents an efficient, online fault detection methodology for NoC interconnect across various faults.

A. Fault Models and Test Vectors

Stuck-at faults can be exercised by using two test packets of 'All-One' and 'All-Zero'. However, to fully detect all bridging faults, a set of walking-one data packets are required, where a single bit is set to one and the rest are set to zero. The Maximum Aggressor Fault (MAF) crosstalk model was proposed in [20] and models faults on a victim line between a two aggressor lines. In the MAF model, faults occur on the victim line due to the effects from the adjacent aggressors. The

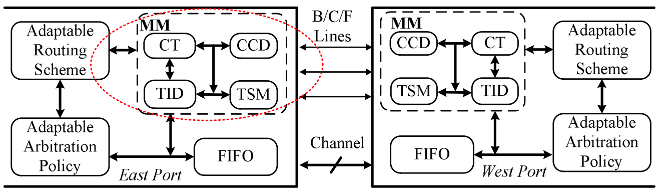


Fig. 1. Monitor Module (MM) architecture

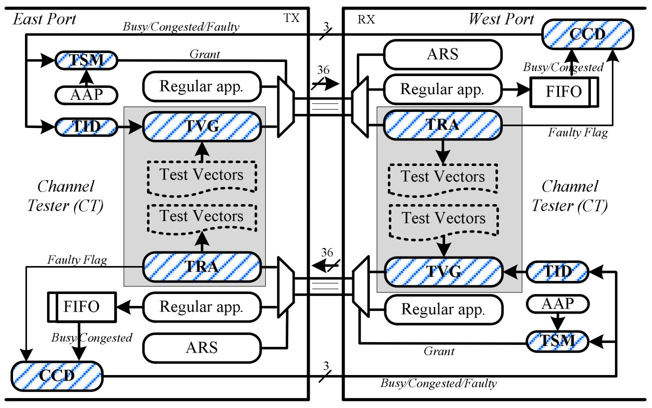
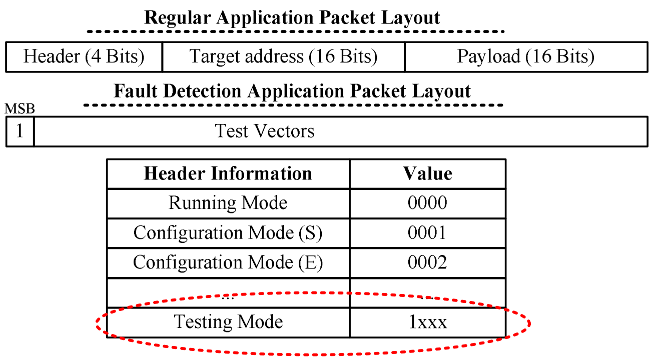


Fig. 2. Connections between the modules inside the MM



Regular Application Packet Layout

Header (4 Bits)	Target address (16 Bits)	Payload (16 Bits)

Fault Detection Application Packet Layout

MSB

1	Test Vectors

Header Information	Value
Running Mode	0000
Configuration Mode (S)	0001
Configuration Mode (E)	0002
...	
Testing Mode	1xxx

Fig. 3. Header information of packets

six most common crosstalk errors are positive/negative glitch, rising/falling delay, and rising/falling speedup. Eight test vectors are used to exercise the crosstalk faults, which include the test vectors for detecting stuck-at and bridging faults.

B. Online Fault Detection

The building block of the proposed fault detection architecture is the MM circuit which is presented in Fig. 1. The MM provides two main functions, which are (1) for the input channel, it analyses the received test vectors and judges whether the channel is faulty; if it is, the fault flag is raised to inform the neighbouring node and thus prevent packets from being corrupted and (2) for the output channel, it provides the channel arbitration between the fault testing and regular application and also it manages the testing frequency to reduce the power consumption while maintaining sufficient testing. The MM contains four sub-components: a Channel Tester (CT), a Channel Congested Detector (CCD), a Traffic Sensor Module (TSM) and a Test Interval Decision (TID) module.

The *CT* is the core module to test the channel and diagnose the faults. It includes the test vectors generator (TVG) and test response analyser (TRA) modules. Fig. 2 illustrates the connections between the modules inside the MM. It can be seen that the TVG in the transmitter (TX) side produces test vectors of all-ones/zeros, walking-one or MAF etc. to stimulate a NoC channel via transmission. The TRA in the receiver (RX) side compares the values of the received test vectors from the TVG. If they do not match, the present line under test is classed as faulty and a fault flag is raised to inform the CCD of a fault in its channel. The TRA in the RX side knows whether the received packet is a test vector or a regular application packet from the definition in the packets header. The general packet layout is defined in Fig. 3, e.g. if the MSB of the header is '1', the packet is for testing otherwise it's a regular application packet. The *CCD* module was proposed in our previous work [21], [22] which provided the Busy and Congested signals to inform the TX router of traffic conditions in the channel. In this approach, it also receives the fault flag from the TRA and outputs the three signals Busy/Congested/Faulty (B/C/F) to the TX router. In such cases the channel is shared by traffic from regular applications and from the fault detection application (CT). For online testing, an arbitration mechanism must be provided to decide whether the regular or fault application can use the channel. Therefore, a priority must be given to one application to allow access to the channel. This channel arbitration is achieved by the *TSM (discussed below)*. The TVG also receives a parameter from *TID* to control the frequency of the testing. The modules of TSM and TID are presented more detailed in the following sections.

The role of *TSM* is to assess B/C/F lines and determine whether the regular application is attempting to transmit data by signalling from the Adaptable Arbitration Policy module (AAP) or, whether the channel has been detected as faulty by signaling from the CCD. From assessing the B/C/F lines, three channel states and three channel conditions are abstracted. Table I outlines all the channel states and conditions. The channel states include Fault-free, Faulty and Recovery. If the *Faulty* line is low, the state is '*Fault-free*' otherwise it's defined as '*Faulty*'. If the channel is faulty but the fault disappears after a period of time, this state is defined as '*Recovery*'. The channel conditions include *C*ongested, *B*usy and *I*dle. If the *congested* line is high, the condition is '*Congested*.' If the *Busy* line is high, the condition is '*Busy*'. If the *Congested* and *Busy* lines are both low, the condition is '*Idle*'. When the channel is in 'fault-free' state, the regular application is assigned priority and permitted access to the channel. In this state, traffic conditions C=1 or B=1 (i.e. Congested/Busy) are experienced and testing is forbidden. If the channel condition is 'idle', then the fault detection application is assigned priority and can test the channel. Otherwise, if the channel is in a 'faulty' state, the regular application is not permitted (no priority given) to transmit data and the traffic conditions are therefore not relevant. Table I outlines the basic priority rules and Fig. 4 illustrates the state transitions and application priority.

TABLE I. NoC CHANNEL STATUSES AND PRIORITY DEFINITION

Channel States (Status)	Conditions			Assigned Application Priority
	C	B	I	
Fault-free	--	--	√	Fault detection application
	--	√	--	Regular application
	√	--	--	Regular application
Faulty	--	--	--	Fault detection application
Recovery	--	--	--	Fault detection application for the duration of critical testing period

Fig. 4 presents the state transitions between the 6 states. The states names are presented in the key on the left and the application priority is presented in each state. It can be seen that after reset, the system is in state S1 i.e. Fault-free (regular), where the regular application can transmit data at any time through the fault-free channel; therefore the priority is assigned to the regular application (R_{egular}). This means that access to the channel is given to R_{egular} and the fault detection application T_{est}, is not permitted, i.e. ($R_{egular} > T_{est}$). After the regular application finishes transmitting data and the channel becomes idle, the state is changed to S3 i.e. Fault-free (test) where the fault detection application is then permitted to gain access and test the channel (i.e. $T_{est} > R_{egular}$). In addition, in state S1, if the

State	Name
S1	Fault-free (regular)
S2	Essential test
S3	Fault-free (test)
S4	Faulty
S5	Recovery (test)
S6	Recovery (regular)

FF	: Fault Flag (0-Fault-free; 1-Faulty)
Reg_req	: Regular application Request
RTC	: Required Test Counter
TIV	: Test Interval Value
CPC	: Check Period Counter

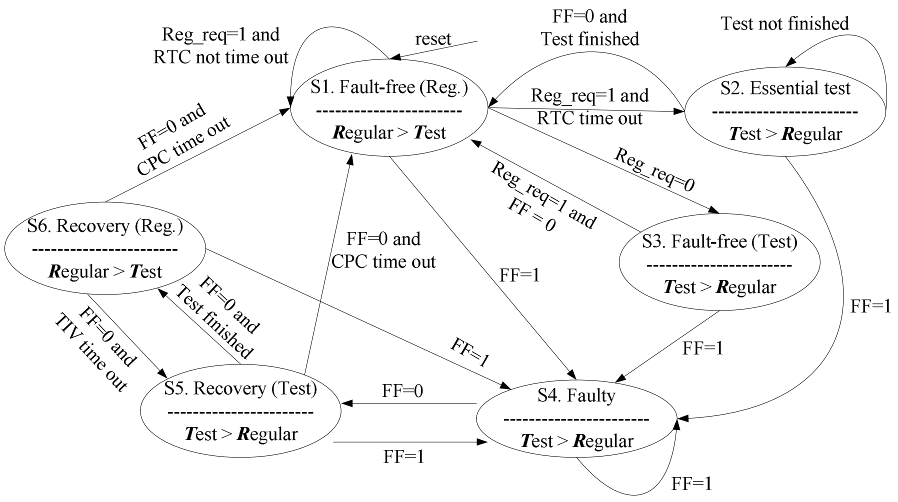


Fig. 4. Assigned application priority

regular application always transmits data but TID module times out and generates a test request, the channel access will still be given to regular application and the test will be postponed. However, if the regular application persists for a long time the channel still needs to be tested in order to guarantee the channel is fault-free. In such cases the state is changed to S2 i.e. Essential Test. In this state ($T_{est} > R_{egular}$), only one test needs to be initiated and the state will quickly return to S1; e.g. it returns channel access to the regular application if no faults occur.

In states S2 and S3, during T_{est}, if a fault is detected by the MM, then fault flag (FF) is '1' and the state will be changed to S4 i.e. Faulty. If this Faulty state occurs, the fault detection application tests the channel repeatedly and restricts the regular application from transmitting data on the detected faulty channel, i.e. $T_{est} > R_{egular}$. If the detection result is reported continuously as faulty, the channel is classed as having a permanent fault and the state remains the same. However, if the fault disappears after testing, the detection result indicates fault-free again and the channel is viewed as being free from faults (i.e. has recovered from an intermittent fault) and the state will be changed to S5 i.e. Recovery (test). If the channel is identified as faulty, the router in the TX side can forward the packets to other fault-free channels based on the adaptive routing algorithm [21].

In state S5, the channel is tested for an additional testing period to guarantee the channel has recovered from the fault. This testing period is defined as the 'check period'. The channel is said to be fault-free if it passes all the tests during the check period and the state is set to S1, otherwise, the state is set to 'Faulty' if a fault is detected again. In addition, in the S5 state ($T_{est} > R_{egular}$), regular application traffic is allowed to transmit data in the channel and the state will be changed to S6 i.e. Recovery (regular). However, the regular application transmission is performed at a restricted rate as testing is also performed. For example, between each channel test the regular application transmits traffic data through the channel in order to minimise intrusion from testing and increase throughput. This arbitration allows the access to the channel to be controlled by the TSM.

C. *Minimising Intrusion of Testing*

In the case when $T_{est} > R_{egular}$, the fault detection application can test the channel. However, it is not necessary to test the channels continuously and at short time intervals. In terms of scalability, the more frequent testing is performed the more dynamic power is consumed. The number of tests should be decreased to minimise power consumption but yet sufficient in quantity to ensure that the channel is tested adequately. Therefore, a test interval parameter generated by the *TID* module is employed to manage the frequency of performing tests on NoC channels, which allows the time period between tests to be varied. This period of time is defined as the test interval value (TIV), N, and it is used to increase or decrease the rate of testing.

The TIV can be regular or irregular. *Regular testing* initiates the tests every time after a pre-set interval period, where the pre-set interval period can be a random value. If the TIV is small, the testing is frequent with high power requirement but can promptly detect potential faults. If the TIV is large, testing is less frequent but the power consumption and intrusion of the test requirements are lower. When *irregular testing* is applied the TIV is not fixed but rather increases after each test is performed. In this paper, a novel approach is implemented by varying the test interval period based on a back-off algorithm [23]. In computer networks the back-off algorithm, known as binary exponential back-off, is employed to schedule retransmissions of the same block of data to avoid network congestion [24]. In this work, novelty resides in using the back-off algorithm in a NoC and also in simplifying the algorithm for an efficient implementation using a compact shift-register. The test interval period starts with a small value and gradually increases after each test, where the test interval value after the ith test is expressed by $N(\mathrm{i}) = 2^{(\mathrm{i}-1)}$, e.g. after the 3rd test (i=3), N=4. In this strategy a threshold value of 128 clock cycles is selected as the maximum test interval value, where N has a range $1 \leq N \leq 128$. Thus the pattern for *irregular testing* based on the back-off is that individual tests are frequent at the beginning but gradually become relaxed as the TIV increases towards N max.

The benefits of using the back-off algorithm are (1) congestion avoidance for NoC channels based on priority definition (2) a compact hardware implementation of the TID module (i.e. a shift-register) that supports scalability of the MM; (3) test frequency management to reduce power consumption. These key benefits are demonstrated in the result section of the paper. Typically, intermittent faults can return quickly after a channel recovers and often we seek to test more frequently just after the recovery for early detection of potential repetitions of a fault. However, as time progresses and no faults are detected, it is likely that the fault will not immediately re-occur and therefore the frequency of testing can be relaxed. This relaxing approach increases throughput performance by freeing the NoC channels for traffic. More importantly, it reduces power consumed by the testing process. The proposed implementation of the back-off algorithm provides a good balance in achieving such performances while maintaining sufficient testing for early detection of faults. The novelty of this paper resides in the use of the back-off algorithm and its implementation to achieve this balance.

IV. RESULTS

This section presents the methodology of evaluation for the online fault testing strategy and also gives a test environment to analyse the regular and irregular testing procedures. The area and power consumption for the implementation of NoC routers with these two mechanisms are also presented.

A. *Methodology of Evaluation*

In order to evaluate the performance of the online testing strategy, the EMBRACE NoC router in our previous work [21], [22] has been extended to include a MM in each of its four co-ordinate channels and has been implemented using VHDL. The modified NoC router is characterized by its ability to detect faults online while minimising intrusion to NoC traffic and resultant area/power overheads from testing. A 3x3 NoC prototype based on a mesh topology was developed as illustrated in Fig. 5. The clock frequency is 100MHz. (Note: The proposed approach is independent of the NoC topology and can be implemented in most general NoC topologies, such as torus, ring-bus and hypercube etc. The test environment in this paper uses a mesh topology as a demonstrator). Each node sends packets at a packet inject rate (PIR), i.e. the number of clock cycles to send one packet. In the system of Fig. 5, node [3,3] sends the packets more frequently than node [1,3]; therefore the test behaviours can be analysed under different traffic conditions, e.g. busy/idle.

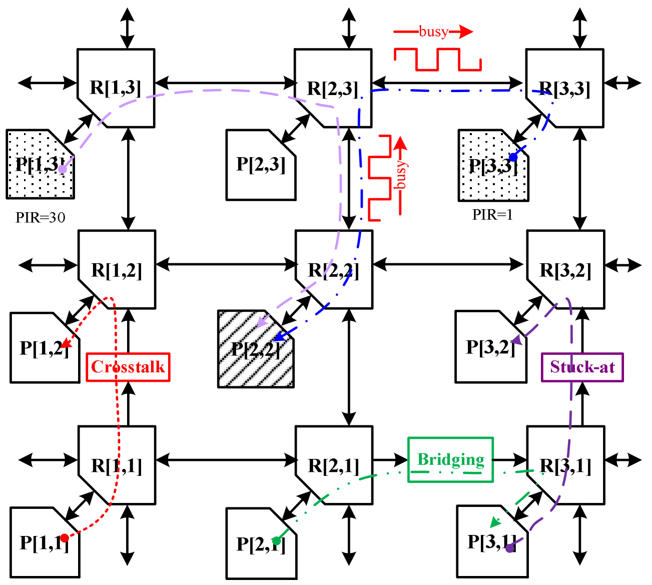


Fig. 5. Test environment (Prototype)

The common stuck-at, bridging and crosstalk faults are also considered, which can be enabled manually to inject faults. Based on this test environment, the fault detection behaviour can be analysed, especially the change in the test interval value during regular and irregular testing methods under different channel conditions.

B. *Results and Analysis*

In this section, the results of the proposed work are highlighted, including testing time, fault coverage, testing impact and testing behaviours.

(a) *Testing time*: Based on the test vectors of fault models [10], [20], the testing time of an *n*-bit channel for stuck-at, bridging and crosstalk fault detection is 2, 2*(n-1) and 9*(n-1) clock cycles, respectively. The MAF crosstalk model is complex; therefore the testing time for this model is longer than the bridging and stuck-at faults.

(b) *Fault coverage and scalability*: As each link is tested by one pair of MM, the proposed mechanism can test links simultaneously and provide up to 100% fault detection for all the links. The MM is independent of the topology and the area footprint per MM remains constant. The MM is scalable as the total area occupied by the MMs in a NoC grows linearly with the number of routers.

(c) *Testing impact*: Network latency and throughput are analysed under different channel conditions to examine how the online testing affects NoC performance. 1). When the channel is fault-free and not busy, based on the application priority definition in Fig. 4, the regular application can get access to the channel immediately without suspension as the fault detection application only tests the channel when it is idle. Therefore, the fault detection application does not affect the network latency and throughput; 2). However, if the channel is busy (i.e. the regular application always transmits data and lasts a long time), the MM will change to S2 state i.e. 'Essential Test' as shown in Fig.4, where one test needs to be initiated. The time period of a single test (denoted by N_t) for stuck-at, bridging and crosstalk faults are 2, 2 and 9 clock cycles. Therefore, during the essential test state, the router latency will be increased by 2, 2 and 9 clock cycles. Assume that the clock period is t_{clk}, the channel width is *w*-bits and the channel occupying time by the regular application before essential test is performed as N_o clock cycles. In this paper, one flit is transmitted every clock cycle. Therefore, the throughput during this time period (T_e) can be defined by (1), where $N_o * w$ is the total packet bits transmitted in the time period of $(N_o + N_t) \times t_{clk}$.

$$T_e = \frac{N_o * w}{(N_o + N_t) \times t_{clk}} \tag{1}$$

In order to calculate degradation we must first obtain the total max throughput (*T*), e.g. when time N_t is used to transmit application data and not perform testing. Equation (**2**) defines *T* where the total transmitted packet bits are increased to be $(N_o + N_t) * w$.

$$T = \frac{(N_o + N_t) * w}{(N_o + N_t) \times t_{clk}} \tag{2}$$

Therefore, throughput degradation (T_d) due to essential testing can be calculated by the relative difference equation (3), where the $T - T_e$ is the difference between the max throughput and the throughput when under essential testing, and T_d represents the essential test throughput degradation comparing to the max throughput *T*.

$$T_d = \frac{T - T_e}{T} = 1 - \frac{N_o}{N_o + N_t} = \frac{N_t}{N_o + N_t} \tag{3}$$

If we consider the essential test initiates when the channel is occupied for *0.1 ms*, then N_o = *10,000* (100 MHz clk). Therefore, the throughput degradation is 0.02%, 0.02% and 0.89% for stuck-at, bridging and crosstalk fault detection. This throughput degradation is very low if considering the added fault tolerance capabilities; 3). However, if the channel is faulty, the regular application is forbidden to transmit data and as a result of the fault the network latency increases and throughput decreases. Such performance degradation depends on the traffic patterns, fault rates and routing policies, which is beyond the scope of this paper. However the performance degradation can be minimised by repair techniques through a data splitting method [16], [25] and/or adaptive routing strategy [22] etc. For example, the HLAFT routing [14] has 0.9% and 12.61% throughput degradation under transpose traffic pattern for 5% and 10% fault rates, respectively.

(d) *Testing behaviours*: The testing behaviours should be analysed under different channel conditions. Section 3 outlined that when the channel is occupied by the regular application, the MM does not test the channel. When the channel is faulty, the MM tests the channel constantly. However, the test behaviour is different when the channel is (1) idle or (2) in the state of recovery as it uses the back-off strategy (irregular testing model). In the following experiments, crosstalk faults were used as they represent a comprehensive range of faults and include stuck-at and bridging type faults; e.g. the test vectors of stuck-at and bridging are included in the crosstalk.

To fully evaluate the back-off strategy, the test behaviours are analysed under the conditions that the channel is in the state of idle or recovery. Results in Fig. 6 demonstrate how the MM can provide a balance between providing rapid testing immediately after a channel recovers from a fault and, gradually relaxing the test interval time as the 'critical testing period' expires to allow the NoC to quickly reach its maximum throughput. This 'critical testing period' is defined as the time permitted for a channel to recover from a fault condition while it is still under critical monitoring. The number of additional tests is varied according to the application field, and the more additional tests that are initiated, the better the channel is monitored. Note that the additional tests in the critical testing period are only used to monitor if an intermittent fault re-occurs. They are not used for transient fault detection as a single-event transient pulse-width is in the range 900ps to 3ns for a 1.5μm technology and rapidly scale down with the technology [26]. However, the typical duration of intermittent faults caused by voltage fluctuations is from several to hundreds of nanoseconds [27]. In this study the clock period is 10ns and the critical testing period is defined as the time to complete seven tests (i.e. more than 1,000ns for irregular testing) which are sufficient to assess channel recovery.

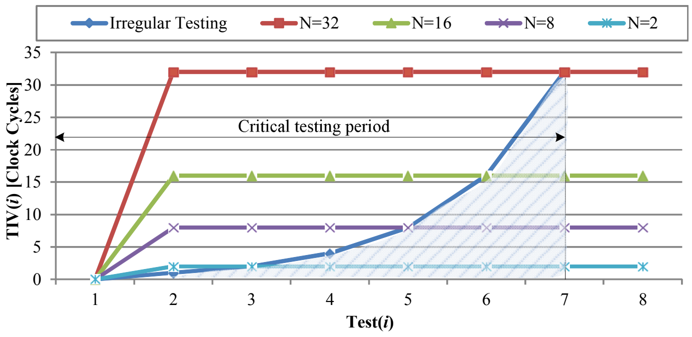


Fig. 6. Test procedures in the critical testing period

Fig. 6 shows the test interval time for regular and irregular testing with the back-off strategy; e.g. for regular testing TIV is set to a constant value, *N*, where $\mathrm{N} \in \{2, 4, 8, 16 \text{ or } 32\}$. The TIV for irregular testing increases in an exponential manner as it is generated by the back-off algorithm. In the critical testing period, the total time period required to complete the seven tests = $\sum_{i=1}^{7}\mathrm{Test(i)} + \sum_{i=1}^{6}\mathrm{TIV(i)}$, where Test(i) is the i^{th} test time and TIV(i) is the test interval value between Test(i) and Test(i+1). The time taken to perform one test for a crosstalk fault is a constant value of 9 clock cycles. Therefore, the time period for all seven tests is dependent on the total duration of time between tests, e.g. $\sum_{i=1}^{6}\mathrm{TIV(i)}$. Fig. 6 shows how the proposed irregular testing strategy minimises the test time (via increased TIV) to free the channel for NoC traffic while in the recovery state; i.e. the shaded area between the curve and X-axis. It can be seen that in tests #1 to #4, the TIV of the *irregular testing* method is very small (TIV < 5 clock cycles) and therefore rapidly tests the channel to provide sufficient testing when recovering from faults. However the rate of testing is relaxed in test #5~#7 after the channel shows complete recovery from the fault (TIV > 30 clock cycles).

In contrast, using the *regular testing* mode when N=2, the channel is tested very frequently. However, this is not ideal, for example, when a channel is diagnosed to be fault-free and in the 'Recovery' state, the channel should be relaxed after a period of time and not be tested consistently with such frequency as more power is consumed. On the other hand, using the *regular testing* mode with a higher value of N=32, the power consumption is low but may lead to sub-standard monitoring of faults due to low frequent testing. The use of the *irregular testing* strategy achieves a balance between the two cases. For example, the total time to complete critical testing of the regular and irregular testing methods are shown in Table II, where the total time required for irregular testing is smaller (126 clock cycles) compared with regular testing (159) when N>=16 for the regular mode. Table II highlights that using the irregular back-off approach, achieves a good balance between completing sufficient tests early after recovery in the test period while reducing the total test time.

TABLE II. COMPLETE CRITICAL TESTING PERIOD (CLOCK CYCLES)

Testing Method / Time required	**Regular**				**Irregular**
	N=2	***N=8***	***N=16***	***N=32***	***Back-off***
$\sum_{i=1}^{7} Test(i)$	63	63	63	63	**63**
$\sum_{i=1}^{6} TIV(i)$	12	48	96	192	**63**
Total time	75	111	159	255	**126**

C. Scalability Analysis

In this section results on the power consumption and area overhead for both regular and irregular testing methods were analysed for the duration of one test phase (defined as the critical testing period of the irregular testing mode, where N reaches a maximum value of 128). The router was implemented on a Xilinx Virtex-6 XC6VLX760 FPGA device for validation and the ISE design suite was used to estimate power consumption and area overhead.

Table III shows the dynamic power consumption of one router with 4 MM for both regular and irregular (back-off) testing and illustrates the dynamic power consumption for irregular and 5 examples of regular testing where the time interval is increased from N=2 up to N=64. Table III highlights the x2.9 reduction in dynamic power consumption between regular testing with a fixed value of N=2 and the proposed irregular testing. The dynamic power decreases in regular testing as the value of N increases, as tests are performed at a lower frequency. However, the power consumption for irregular testing is a constant value of 0.235W. It achieves a low power consumption which is almost equal to that of regular testing when N is fixed at 32 (with such infrequent tests it may be difficult to guarantee that any recurring faults can be promptly detected).

In addition the router was synthesised and mapped based on the SAED 90nm CMOS libraries to fully assess the scalability. The hardware areas occupied by the four MMs and one router are summarized in the bottom of Table IV for regular/irregular testing methods and the three different fault types: stuck-at, bridging and crosstalk. The total MM area overhead is calculated for four MMs occupied in one router. It illustrates that (1) the MMs and router with irregular testing occupied a larger area than those with regular testing irrespective of the fault type. This is mainly due to the TID module in the irregular testing implementation occupying additional area which does not exist in the regular testing; and (2) the router for the crosstalk faults occupied a larger area than those for the bridging faults which is larger than the one for stuck-at faults. As the crosstalk fault model is the most complex this additional area overhead is expected. It can be also seen that the area overhead of the MMs for the crosstalk is the largest, i.e. 16.24% and 18.84% for regular/irregular testing modes. However, the irregular testing model exhibits a small increase of area overhead (~2.5%) compared to the regular testing model.

TABLE III. DYNAMIC POWER CONSUMPTION COMPARISON

Testing methods \ Power consumption		Dynamic (W)
Irregular	Back-off algorithm	0.235
Regular	N=2	0.685
	N=8	0.490
	N=16	0.354
	N=32	0.238
	N=64	0.151

The results demonstrate that the inclusion of irregular testing using back-off does not prohibit its scalability as the additional area overhead of [10.16~18.84%] is relatively low combined with its low power overhead. Therefore, irregular testing provides a good trade-off between achieving low area/power consumption while performing dynamic rates of testing during critical recovery periods to enable prompt detections of faults. Achieving low area/power consumption is a critical performance outcome to ensure that the scalability of any NoC-based system can be sustained. In the meantime, one extreme situation should also be considered, where the fault rate is very high. For this scenario, the regular testing with high frequency (e.g. N=2) can guarantee the system is monitored closely and faults are detected promptly. Therefore, the regular and irregular testing methods can be selected based on the application under consideration.

Table IV also presents the performance comparison of the proposed regular/irregular testing models against other approaches [7], [10], [12]. Only [12] and this proposed work provide an online testing capability. The advantage of [12] is that it can test the faults of both interconnect and routers; therefore the area overhead is a little high. However the fault detection mechanism of [12] uses a token-based strategy where each channel is monitored sequentially; this exhibits significant test times. Our proposed approach performs concurrent testing and therefore avoids such delays. For example, the tests for all individual channels can be initiated simultaneously as one pair of MM is allocated per channel. The approach in [12] can detect stuck-at and bridging faults, but the test units occupy a larger area overhead. The approaches in [7] and [10] also provide concurrent testing for NoC interconnects however, they only perform offline testing and must stop the executing NoC application during testing. In addition, the approach in [7] also requires a higher area overhead for test units. For the crosstalk fault detection of almost same channel width (36/32-bit), the area overhead of our work is a little higher (~5%) than the approach in [10]. However, this result does not prohibit its scalability as the additional area overhead of 5% is relatively low when we consider that the added advantage provided by online testing.

TABLE IV. PERFORMANCE COMPARISON/AREA OVERHEAD

Approaches \ Metrics	Online	Non-intrusive	Fault type	Link width	Area overhead
[12][a]	√	×	Stuck-at	32	23%
			Bridging		65%
[7][b]	×	N/A	Shorts	48	43.96%
[10][c]	×	N/A	Crosstalk	16	8.01%
				32	13.03%
				64	22.64%
This work (Regular testing)	√	√	Stuck-at	36	7.19%
			Bridging		11.17%
			Crosstalk		16.24%
Irregular testing	√	√	Stuck-at	36	10.16%
			Bridging		13.90%
			Crosstalk		18.84%

a. Modules used in the calculation of area overhead include Test Pattern Generator, Test Response Analyzer, Fault Diagnosis Module;

b. Area overhead is calculated by (TDG+TED)/RASoC= (341+401)/1688 = 43.96%;

c. Area overhead is calculated based on point-to-point MAF self-test methodology.

V. CONCLUSION

This paper presents a novel online fault detection strategy for NoC interconnect which uses a monitor module with a back-off capability to achieve irregular or dynamic testing. The reported benefits demonstrate an increase in the NoC system performance by providing dynamic testing to promptly evaluate the channel in critical recovery periods, and also to relax testing when it has completely recovered. In particular, the results demonstrate low area/power overheads of the MM to meet scalability requirements. Having a NoC that can detect faults efficiently is important however it is also important to establish approaches to restore functionality under the presence of faults. Therefore, future work will extend the current online fault detection strategy by investigating repair mechanisms.

ACKNOWLEDGMENT

Junxiu Liu is supported by University of Ulster's the Vice-Chancellor's Research Scholarship (VCRS) and the Vice-Chancellor's Student Fund. This research is supported in part by Altera University Programme.

References

[1] L. Benini and D. M. Giovanni, “Networks on Chips: A New SoC Paradigm,” Computer, vol. 35, no. 1, pp. 70–78, 2002.

[2] S. Madduri, R. Vadlamani, W. Burleson, and R. Tessier, “A monitor interconnect and support subsystem for multicore processors,” in Design, Automation & Test in Europe Conference & Exhibition, 2009, pp. 761–766.

[3] A. Agarwal, B. Raton, C. Iskander, H. Multisystems, and R. Shankar, “Survey of Network on Chip (NoC) Architectures & Contributions,” Journal of Engineering, Computing and Architecture, vol. 3, no. 1, pp. 1–15, 2009.

[4] R. Marculescu and P. Bogdan, “The Chip Is the Network: Toward a Science of Network-on-Chip Design,” Foundations and Trends in Electronic Design Automation, vol. 2, no. 4, pp. 371–461, 2007.

[5] R. A. Shafik, J. Mathew, and D. K. Pradhan, “Introduction to Energy-Efficient Fault-Tolerant Systems,” in Energy-Efficient Fault-Tolerant Systems, 2014, pp. 1–10.

[6] “ITRS.” [Online]. Available: http://www.itrs.net/.

[7] M. Hervé, P. Almeida, F. L. Kastensmidt, E. Cota, and M. Lubaszewski, “Concurrent Test of Network-on-Chip Interconnects and Routers,” in 11th Latin American Test Workshop (LATW), 2010, pp. 1–6.

[8] É. Cota, L. Carro, and M. Lubaszewski, “Reusing an On-chip Network for the Test of Core-Based Systems,” ACM Transactions on Design Automation of Electronic Systems, vol. 9, no. 4, pp. 471–499, Oct. 2004.

[9] A. M. Amory, E. Brião, É. Cota, M. Lubaszewski, and F. G. Moraes, “A Scalable Test Strategy for Network-on-Chip Routers,” in IEEE International Test Conference, 2005, pp. 1–9.

[10] C. Grecu, P. Pande, A. Ivanov, and S. Res, “BIST for Network-on-Chip Interconnect Infrastructures,” in 24th IEEE VLSI Test Symposium, 2006, pp. 30–35.

[11] M. Botelho, F. L. Kastensmidt, M. Lubaszewski, E. Cota, and L. Carro, “A Broad Strategy to Detect Crosstalk Faults in Network-on-Chip Interconnects,” in 18th IEEE/IFIP International Conference on VLSI and System-on-Chip, 2010, pp. 298–303.

[12] M. R. Kakoee, V. Bertacco, and L. Benini, “A Distributed and Topology-Agnostic Approach for On-line NoC Testing,” in 5th IEEE/ACM International Symposium on Networks on Chip, 2011, pp. 113–120.

[13] Z. Zhang, A. Greiner, and S. Taktak, “A Reconfigurable Routing Algorithm for a Fault-tolerant 2D-Mesh Network-on-Chip,” in 45th IEEE/ACE Design Automation Conference, 2008, pp. 441–446.

[14] A. Ben Ahmed and A. Ben Abdallah, “Graceful Deadlock-free Fault-tolerant Routing Algorithm for 3D Network-on-Chip Architectures,” Journal of Parallel and Distributed Computing, vol. 74, no. 4, pp. 2229–2240, Jan. 2014.

[15] C. Feng, Z. Lu, A. Jantsch, M. Zhang, and Z. Xing, “Addressing Transient and Permanent Faults in NoC with Efficient Fault-Tolerant Deflection Router,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 21, no. 6, pp. 1053–1066, Jun. 2013.

[16] A. Vitkovskiy, V. Soteriou, and C. Nicopoulos, “A Dynamically Adjusting Gracefully Degrading Link-Level Fault-Tolerant Mechanism for NoCs,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 31, no. 8, pp. 1235–1248, 2012.

[17] F. A. Pereira, L. Carro, E. Cota, and F. L. Kastensmidt, “Evaluating SEU and Crosstalk Effects in Network-on-Chip Routers,” in 12th IEEE International On-Line Testing Symposium, 2006, pp. 191–192.

[18] C. Concatto, P. Almeida, F. Kastensmidt, E. Cota, M. Lubaszewski, and M. Herve, “Improving Yield of Torus NoCs through Fault-Diagnosis-And-Repair of Interconnect Faults,” in 15th IEEE International On-Line Testing Symposium, 2009, pp. 61–66.

[19] P. S. Bhojwani and R. N. Mahapatra, “Robust Concurrent Online Testing of Network-on-Chip-Based SoCs,” IEEE Transactions On Very Large Scale Integration (VLSI) Systems, vol. 16, no. 9, pp. 1199–1209, 2008.

[20] C. Michael, D. Sujit, B. Xiaoliang, and Z. Yi, “Fault Modeling and Simulation for Crosstalk in System-on-chip Interconnects,” in IEEE/ACM International Conference on Computer-Aided Design, 1999, pp. 297–303.

[21] S. Carrillo, J. Harkin, L. McDaid, S. Pande, S. Cawley, B. McGinley, and F. Morgan, “Advancing Interconnect Density for Spiking Neural Network Hardware Implementations using Traffic-aware Adaptive Network-on-Chip Routers,” Neural Networks, vol. 33, no. 9, pp. 42–57, Sep. 2012.

[22] S. Carrillo, J. Harkin, L. J. McDaid, F. Morgan, S. Pande, S. Cawley, and B. McGinley, “Scalable Hierarchical Network-on-Chip Architecture for Spiking Neural Network Hardware Implementations,” IEEE Transactions on Parallel and Distributed Systems, vol. 24, no. 12, pp. 2451–2461, 2013.

[23] M. M. Shurman, M. F. Al-Mistarihi, and Z. A. Alomari, “MAC Layer Back-off Algorithm for Ad hoc Networks,” in 36th International Convention on Information & Communication Technology Electronics & Microelectronics (MIPRO), 2013, pp. 446–451.

[24] IEEE standards, “ANSI/IEEE Std 802.11 - Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications,” 1999.

[25] A. Kologeski, C. Concatto, L. Carro, and F. L. Kastensmidt, “Improving Reliability in NoCs by Application-Specific Mapping Combined with Adaptive Fault-Tolerant Method in the Links,” in 16th IEEE European Test Symposium, 2011, pp. 123–128.

[26] B. Narasimham, V. Ramachandran, B. L. Bhuva, R. D. Schrimpf, A. F. Witulski, W. T. Holman, L. W. Massengill, J. D. Black, W. H. Robinson, and D. Mcmorrow, “On-Chip Characterization of Single-Event Transient Pulsewidths,” IEEE Transactions on Device and Materials Reliability, vol. 6, no. 4, pp. 542–549, 2006.

[27] P. M. Wells, K. Chakraborty, and G. S. Sohi, “Adapting to Intermittent Faults in Multicore Systems,” in 13th International Conference on Architectural Support for Programming Languages and Operating Systems, 2008, pp. 255–264.