Low Overhead Monitor Mechanism for Fault-tolerant Analysis of NoC

Junxiu Liu, Jim Harkin, Yuhua Li, Liam Maguire
School of Computing and Intelligent Systems,
University of Ulster, Magee Campus,
Derry, Northern Ireland, UK
{liu-j4, jg.harkin, y.li, lp.maguire}@ulster.ac.uk

Alejandro Linares-Barranco
Robotic and Technology of Computers Lab,
University of Seville,
Seville, Spain
alinares@us.es

***Abstract*— Modern Networks-on-Chip (NoC) have the capability to tolerate and adapt to the faults and failures in the hardware. Monitoring and debugging is a real challenge due to the NoC system complexity and large scale size. A key requirement is an evaluation and benchmarking mechanism to quantitatively analyse a NoC system's fault tolerant capability. A novel monitoring mechanism is proposed to evaluate the fault tolerant capability of an NoC by: (1) using a compact monitor probe to detect the events of each NoC node; (2) re-using the exist NoC infrastructure to communicate analysis data of back to a terminal PC which removes the need for additional hardware resources and maintain hardware scalability and (3) calculating throughput, the number of lost/corrupted packets and generating a heat map of NoC traffic for quantitative analysis. The paper presents results on a case study using an example fault-tolerant routing algorithm and highlights the minimal area overhead of the monitoring mechanism (~6%). Results demonstrate that the proposed online monitoring strategy is highly scalable due to the compact monitor probe and the ability to reuse the existing NoC communication infrastructure. In addition, the traffic heat map generation and throughput display demonstrates benefits in aiding NoC system prototyping and debugging.**

Keywords— Networks-on-Chip, performance monitoring, fault tolerant, hardware adaption

I. INTRODUCTION

The complexity of modern Systems-on-Chip has seen the introduction of new interconnection strategies such as NoC which allow scalable on-chip communication between large numbers of processing components. The NoC strategy was first introduced in [1] and is composed of routers, channels and processing elements. The topology defines how to connect NoC routers on-chip and 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].

Fault tolerance and adaptive capabilities are challenges [3], [4] for modern NoCs due to the increase in physical defects in advanced manufacturing processes, as often faults occur post manufacturing. Such adaptation reflects the capability of a NoC system to maintain or decrease the performance gracefully in the context of internal faults or external interference. Some approaches have been proposed to enhance the NoC system fault-tolerant capability, such as fault tolerant and detection methods [5]–[8] and adaptive routing policies when faults occur [9]–[14]. A quantitative analysis of the fault-tolerant capability is very beneficial for design, evaluation and benchmarking. The key analysis tasks include heat map generation of NoC traffic, throughput calculation, communication path regeneration and counting the number of lost/corrupted packets due to the occurrence of faults. However, the quantitative analysis is a significant challenge due to the large scale (+250 CPU cores per device by 2015), limited test pins (e.g. using logic analyser) and extra area overhead introduced by a built-in debugger (e.g. using Altera SignalTap, Xilinx Chipscope or Lattice Reveal Analyser). Therefore to aid the NoC system design and evaluation requires the development of a monitoring mechanism that can detect the events in the NoC, visualize the traffic distribution, calculate the throughput, the number of lost/corrupted packets etc. for performance analysis.

In this paper, a monitoring mechanism is presented which aims to provide quantitative analysis of the fault tolerant capability of a NoC system. This approach is novel as (1) the NoC interconnection is reused to collect the system events, i.e. communication of monitoring data does not introduce large hardware area overhead; (2) the monitor probe of each node in the NoC system is area-efficient and (3) a generic communication protocol is designed to provide compatibility with other NoC systems. The proposed monitoring mechanism is complementary to conventional simulators. The conventional simulators can analyse the NoC system performance using the simulation data over short time periods, especially at the design time. However, the proposed monitoring mechanism has the capabilities to observe online the run-time behaviour of the entire hardware NoC system and aid the hardware debugging and implementation. For example it can observe real-time interactions with external sensor data. The remainder of the paper is organized as follows. Section 2 provides a summary of previous work and section 3 discusses the proposed monitoring mechanism and its hardware operation and section 4 introduces its software front-end. Section 5 provides results from a case study of a fault-tolerant NoC router, and section 6 provides a conclusion and highlights future work.

II. RELATED WORK

A NoC performance analysis monitor scheme was proposed in [15]. It included a hardware trace monitor in the NoC system and trace software on the host computer. The trace monitor collects the NoC system states and sends the data to the PC software for the performance evaluation. It uses Ethernet to collect the data in real-time. However, the NoC operates at a relative low clock frequency (25MHz) to allow the data collection in real-time. The monitor also needs to connect every channel through additional, dedicated wires which prohibits scalability for a large system. An industrial NoC emulation and verification environment, namely

NOCEVE, was proposed in [16]. It can analyse the performance of large-scale multi-FPGAs including traffic distribution, latency and throughput etc. Two models of real-time visualization and post-execution data analysis were proposed. A run-time monitoring mechanism was also proposed in [17] to decide the optimal NoC buffer size and capture system behaviour. A monitoring probe module was embedded inside each router. All the monitoring probe modules send traffic information to a global interface which is responsible for gathering all the snapshot data. They are connected by a dedicated point-to-point (P2P) connection which prohibits system scalability. Similarly, a configurable monitor was proposed in [18] where a probe component was embedded in the network interface to observe events between the router and processing element. The events were processed in the pre-processing module and then sent to a probe management unit for collection. However the probes introduced a high area overhead such that a 4 multi-purpose probe was approximately equal to 55% of the total area of the network interface and router.

In summary, current approaches have the aforementioned weaknesses of (1) they do not provide the quantitative analysis required for a fault-tolerant capability [17]; (2) prohibit scalability due to large area overheads of monitor trace module and data connection for events collection in the NoC system [15], [18]; and (3) commercial copyright as they are not open source [16]. To be available to the general public via open source can promote research by supporting independent review and evolution of code. For a modern NoC monitoring mechanism, several key functions need to be addressed: capability to provide analysis on NoC fault tolerance, scalability – do not have an impact on area overhead, and accessibility (open-source to NoC community) to aid others in designing and debugging the NoC. It is now timely to investigate such a monitoring mechanism as fault tolerance and adaptive capabilities are of paramount importance with ever increasing density of large scale electronic systems.

III. LIGHT WEIGHT MONITOR PROBE AND STATISTICS NODE IN THE NOC

In our previous work [8], [13], [14], [19], the authors developed a NoC router design, namely EMBRACE, which demonstrated a traffic-aware and online fault testing equipped router. This section outlines the monitor probe design based on the EMBRACE router and the communication protocol between the NoC and the computer.

A. Regular NoC node

Fig. 1 (a) presents the overall system structure. A remote server is used to compile the HDL code of the NoC system due to the long synthesis time of the hardware. The server is connected to a host computer via Ethernet. A JTAG server operates on the host computer which provides the bit-stream download and debugging service for the remote server. A NoC system is implemented on the FPGA board and the host computer is connected to the FPGA. The monitoring software runs on the host computer and provides analysis of the NoC fault tolerant capability based on traffic data from the NoC on the FPGA.

The example NoC system in the Fig. 1 (a) is a 2D-mesh ($DIM_x * DIM_y$) NoC system and is composed of NoC nodes, probe modules and a single statistics node. Each NoC node is positioned by one pair of coordinates and is connected to the neighbouring router nodes through channels at North/E/S/W directions. The NoC node includes one router and one processing element (PE). A probe module is embedded in the PE but it collects all the traffic events of the corresponding router on all of the directions including N/E/S/W and the events of the PE at the local connection. For example, in Fig. 1 (a) the probe in PE(2,2) collects all the events of router (2,2) on its N/E/S/W directions and the local connection events of PE(2,2). The PE obtains the event data from the probe and then constructs the statistics packets which are forwarded to statistics node (STAT). The STAT node is connected to the complete NoC system. The coordinate of the STAT node in Fig. 1 (a) is $(DIM_x + 1,2)$. It is responsible for collecting the statistics data from each NoC node (router/PE) and uploading it to the computer. In order to differentiate from the STAT node, all other nodes in the NoC except STAT are referred to as regular nodes.

Similar to the computer network, the NoC uses a specific

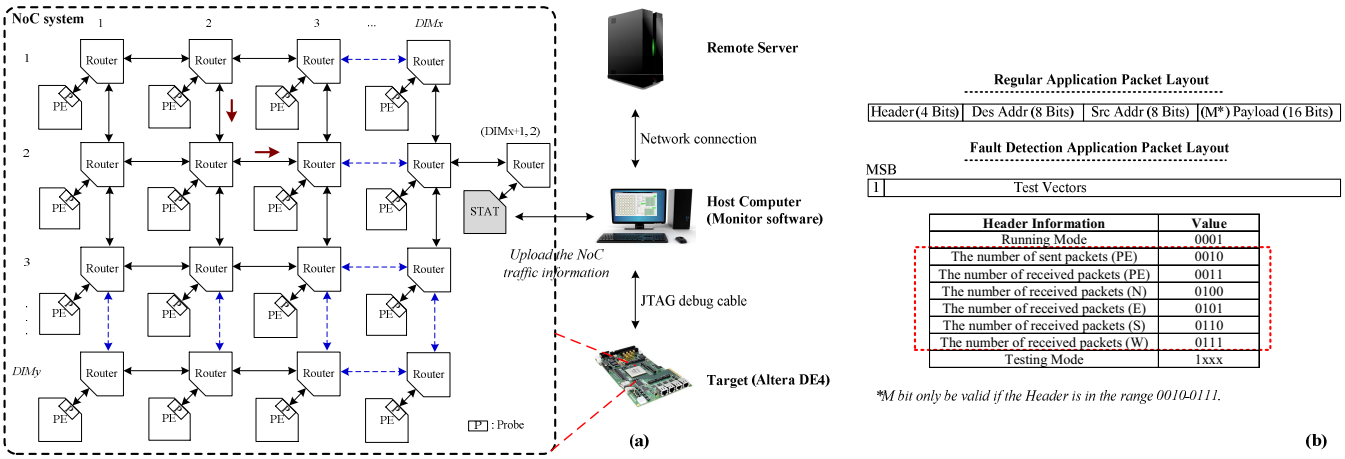

Fig. 1. Example NoC monitoring setup: (a) System structure; (b) The packet layout of the NoC

packet format to forward the data across the network. The packet layout of EMBRACE is defined in Fig. 1 (b). When the MSB is '1', the packet is used as a test vector for online fault detection. The fault detection packets are generated at each router and sent across the local N/E/S/W channels. However, if the MSB is '0', the packet is a regular application packet [8]. In the regular application mode of operation, the packet can have several functions - defined by the packets header between values $(0001)_2$ and $(0111)_2$. When the packet header contains $(0001)_2$, it is classed as a normal packet containing data for the regular application mapped to the NoC; if the header is between $(0010)_2$ to $(0111)_2$, the packet is classed as a statistics packet which contains traffic event data on the number of packets a PE sent/received, the number of packets a router received at N/E/S/W, respectively. For the statistics packet, the most significant bit (*M*) of the payload (see top packet layout of Fig.1 (b)) is used to indicate the channel status. If it is '1', the channel is defined as faulty; if it is '0', the channel is fault-free. The setting of the *M* bit is controlled via the online test mechanism available in the example NoC [8]. Let's take an example to illustrate the packet layout definition. Assume the STAT node is (5,2) in a 4x4 2D-mesh NoC; regular node (2,2) sends packets to (3,2) and node (2,1) sends packets to (2,2) as depicted in Fig. 1 (a) where $\mathrm{DIM_x} = 4$, $\mathrm{DIM_y} = 4$. If node (2,2) sends a packet and the packet is $(800000001)_{16}$, it is defined as a test packet (MSB is '1') and therefore performs testing on the channel between NoC node (2,2) and (3,2); if the packet is $(132220251)_{16}$, then it is a normal packet which means the destination is (3,2), the source is (2,2) and the data application payload is $(0251)_{16}$. If the packet is $(252220005)_{16}$, then it is a statistic packet which means the packet is sending it to the STAT node (5,2), where it define the source node as (2,2) and that 5 packets were sent by PE(2,2). If the packet is $(452220008)_{16}$, then it is again a statistic packet where the source node is (2,2) and the router (2,2) has received 8 packets via its northern channel. If the packet is $(652228000)_{16}$, then it is classed as a statistic packet where the source node is (2,2) and its southern channel of (2,2) is defined as faulty as the *M* bit of payload is '1'.

Based on the packet layout definition, the regular application, test vector and statistics packets can be forwarded correctly across the same NoC network without any additional interconnect. Extra area overhead is not introduced as the statistic data is forwarded across the existing NoC network. The work flow of the proposed hardware monitoring system is presented in Fig. 2. After the NoC is reset, the regular (NoC) node is in the idle state until it starts to run an application task. After the application has run for a time period (defined as window time), the application is interrupted and the monitor probe inside the PE collects the statistics data and then sends it to the STAT node. In this approach, the payload of the packets is 16-bit in length. In addition, the *M* bit of the payload is used to indicate the channel status. Therefore, a maximum 15 bits can be used for counting the event, i.e. the maximum number of traffic events is $2^{15} = 32{,}768$, however this can be increased. Assume the NoC system clock frequency is 100MHz, therefore the window time can be up to ~328 µs. Note that the window time can be longer if the payload width is set to be wider (16-bit is used as an example size). Note that in this approach, the monitor probe collects and sends the statistics data to the STAT node synchronously. The example application running in the NoC hardware is a spiking neural network (SNN) [13], which operates at a very low frequency e.g. the inter-spike interval in the order of milliseconds. As the average firing rate is low, the channel traffic status is idle at most of time, which allows all the statistic packets reach the STAT node within the data collection time. However, if the regular application traffic is heavy, an independent physical communication infrastructure, e.g. the dedicated channels in [15], [16], should be employed to collect the statistics data.

B. STAT node

The STAT node is responsible for receiving statistics data from all regular nodes in the NoC and uploading it to the PC. The structure of the STAT node is presented in Fig. 3. It includes a FIFO, a statistics stream generator and a communication interface. The FIFO is required as the packet receive rate from the regular nodes and the byte stream send rate to the computer can differ; the statistics stream generator is responsible for converting the 36-bit statistics packets from the NoC to an 8-bit byte streams for uploading to the PC. The communication interface can be UART, SPI2USB or Ethernet. The UART is a light weight communication protocol which does not introduce much area overhead; SPI2USB was implemented in [20] which has a high speed rate but requires a microcontroller as a converter; Ethernet is the highest speed protocol however it requires a soft core processor on the FPGA due to its complex protocol, therefore it has large area overhead [15]. However, these three connection protocols can be selected based on the requirement of application. In this paper, the light weight UART protocol is used.

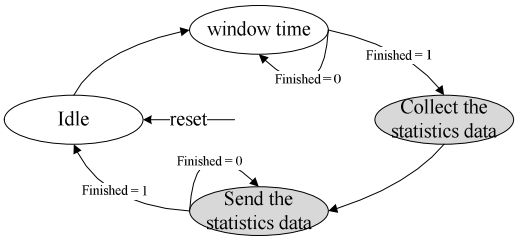

Fig. 2. The work flow of regular node in the NoC

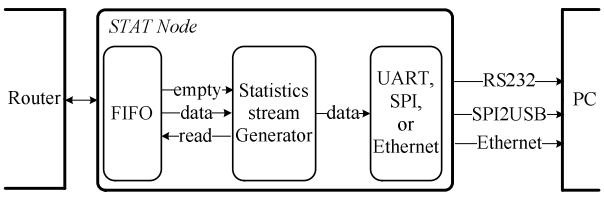

Fig. 3. The statistics node structure

The STAT node and computer communicate with each other based on a communication protocol which is illustrated

in Fig. 4. The communication begins with 0x42; the next byte is the type of statistical data which is consistent with Fig. 1 (b), e.g. the value of 2 represents the number of packets PE has transmitted; the third byte is the STAT node address; the fourth byte is the source node address which sent the statistics packets; the next two bytes of Data (byte 1 and 2) correspond to the payload of statistics packets; and the final byte is 0x4C. The software hosted on the computer can analyse the NoC traffic events and provide a quantitative result on system performance.

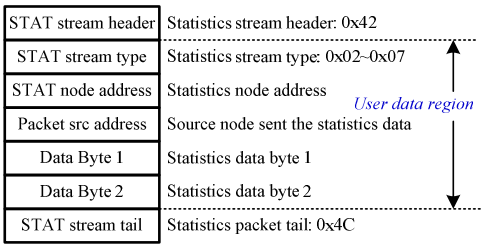

Fig. 4. Data exchange format between STAT node and computer

IV. NoC System Performance Analysis Software

Performance analysis software for the NoC system was investigated for the host PC. It provides several functions in quantitatively evaluating NoC performance including throughput, the number of lost/corrupted packets calculation and the heat map generation of NoC traffic. In particular, the software highlights where faults have been detected and the number of lost packets as a result. This enables design exploration and evaluation of adaptive NoC routing algorithms when faults occur.

A. Traffic heat map generation

The traffic heat map feature of the software enables researcher to quickly identify busy (hot-spot) nodes, congested and faulty channels. The clear view of the heat map helps designers in optimising the mapping of application tasks to the different PEs, and also designing more efficient routing algorithms etc. to enhance the fault-tolerant capability of the NoC.

Firstly in the software, the physical parameters of routers and channels need to be identified as part of the visualisation parameters of the data on the NoC. Each node consists of one router and 4 input channels. The size parameters of the routers and channels are defined in Fig. 5 - the width of a router is R_w, the height of router is R_h; the width of a channel is C_w, the length of channel is C_l; the margin between the channel and router top edge is M_u, the distance between the input and output channel is M_m and the margin between the output channel and router bottom edge is M_b.

To plot the router and channels in map, the start point of every router and channel should be identified. The physical position is defined by (1), assume that: one node (*i, j*) in a $DIM_x * DIM_y$ NoC system, (x_o, y_o) is the start point of the traffic heat map in the *x*/*y*-axis. R(*i, j*) is the start point of router (*i, j*) and N/E/S/WIC(*i, j*) is the start point of the Northern/E/S/W input channel of router (*i, j*). Based on the size parameters in Fig. 5 and coordinates in (1), any size of NoC system can be visualised successfully.

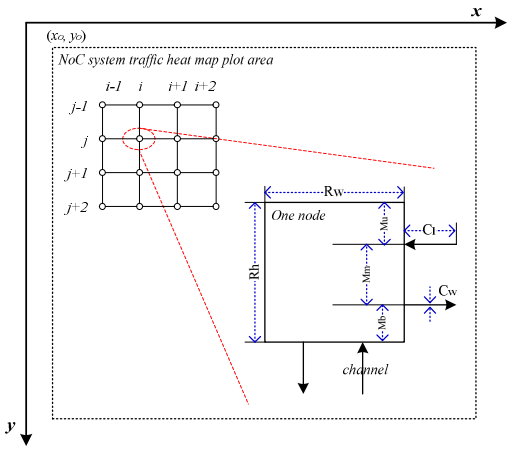

Fig. 5. The size parameters of components in the NoC system

Secondly, a colour filled scheme for the channel is employed to generate the heat effects in the map. The darker the colour of a channel, the heavier the traffic is. The colour is presented in the RGB data format, for example, if RGB = (0, 0, 0), the colour is black. The colours between red and yellow are used to present the traffic load in this approach. If the channel is red, the traffic is heavy and if yellow, the traffic is light. For the colour between red and yellow, the R and B values can be set to 237 and 47; only the Green value needs to be changed. Therefore, the $RGB = (237, G, 47)$ where *G* is the only value needs to be set and $0 \leq G \leq 255$. If *G*= 0, the colour is red; if *G*= 255, the colour is yellow. The NoC router sends one packet per clock cycle (a packet only contains one flit in this approach – as defined by the example SNN application [13] used in this study). For a given window time of *N* clock cycles, the maximum number of packets that can be sent or received is equal to *N*. Assume that the number of traffic events in Fig. 4 is R_{events}, the *G* value can be calculated by (2). For example, if the window time is *N=300* clock cycles and the number of events is $R_{events} = 260$, then *G=34* (by (2)) and RGB (237, 34, 47) will be used to fill the channel colour. This colour is close to red which means the traffic is heavy. In addition, the *M* bit of payload, shown in Fig. 1 (b), is used to indicate the status of a channel. If it is '0', the channel is fault-free. However, If it

$$\begin{cases} R(i,j) = \left(x_o + i * (R_w + C_l),\ y_o + j * (R_w + C_l)\right) \\ NIC(i,j) = (x_o + i * (R_w + C_l) + M_u,\ y_o + j * R_w + (j-1) * C_l), j \neq 0 \\ EIC(i,j) = \left(x_o + (i+1) * R_w + i * C_l,\ y_o + j * (R_w + C_l)\right) + M_u, i \neq DIM_y - 1 \\ SIC(i,j) = (x_o + i * (R_w + C_l) + M_u + C_w + M_m,\ y_o + (j+1) * R_h + j * C_l), j \neq DIM_x - 1 \\ WIC(i,j) = (x_o + i * R_w + (i-1) * C_l,\ y_o + j * (R_w + C_l) + M_u + C_w + M_m), i \neq 0 \end{cases} \quad (1)$$

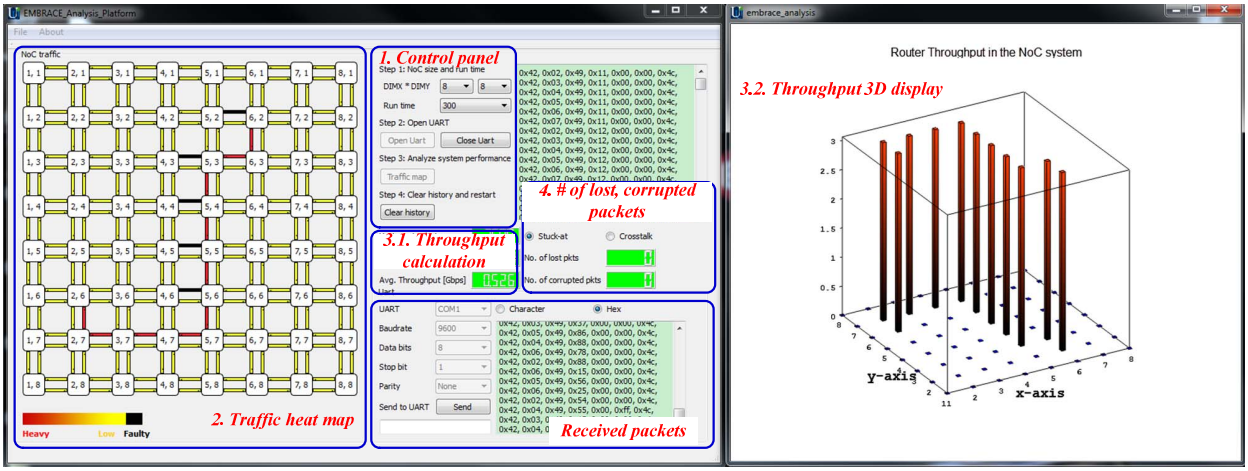

Fig. 6. Monitor software interface

is '1', the channel is faulty and the RGB is set to (0,0,0), i.e. the colour black.

$$G = 255 * (1 - \frac{R_{events}}{N}) \quad (2)$$

The router uses the distributed adaptive routing algorithm [8], [13]. Every port (N/E/S/W and local) can send and receives packets at the same time. The colour filled scheme is applied to the input and output directions of all the ports. Based on the coordinates definition and colour filled scheme, a traffic heat map is generated. The heat map shows the overall traffic status in a visualized manner and enables behaviour analysis of routing algorithm when faults occur.

B. Throughput calculation

Each NoC router has four ports to connect neighbouring nodes and a local port to connect the PE. The router works on a clock frequency of 100MHz and the channel width is 36-bit, therefore the maximum throughput for a single port is 3.6Gbps [13]. For a given window time of N clock cycles, the router throughput, T, can be calculated by (3), where $\sum R = R_N + R_E + R_s + R_w + R_L$, i.e. the number of received packets through North/E/S/W/Local ports, f_c is the clock frequency. The throughput of each node is visualised in a 3D column-format so clarity across the 2-D mesh.

$$T = \frac{\sum R \times 36}{N} \times f_c \quad (3)$$

Average throughput is also used to reflect the system performance. It is defined by (4), where the T_{avg} is the average throughput of the system and $T(i,j)$ is the throughput of router (i,j).

$$T_{avg} = \sum_{i=1}^{DIM_x} \sum_{j=1}^{DIM_y} T(i,j) \,/ (DIM_x * DIM_y) \quad (4)$$

C. The number of loss/corrupted packets calculation

The symbols of $R_{N/E/S/W/L}$ and $S_{N/E/S/W/L}$ are defined as the number of received(R)/sent(S) packets through North/E/S/W/Local port. The total number of faults experienced in the NoC is reflected by the number of packets which are lost in communication due to the fault, e.g. a fault in a channel etc. We calculate the impact, F, from the faults on packet transmission, using (5). F is equal to the total number of packets a router receives on the N/E/S/W/L ports, minus the total number of packet the router has sent. For a fault-free router, the number of total received packets is equal to the number of packets sent. For a faulty router the total number of transmitted and received packets will vary.

$$F = (R_N + R_E + R_S + R_W + R_L) - (S_N + S_E + S_S + S_W + S_L) \quad (5)$$

Stuck-at and crosstalk faults are employed in this paper as they are the most common faults in the NoC (the detailed fault models can be found in the authors' previous work [8]). When a packet is forwarded to a faulty channel with stuck-at fault, it will be corrupted. However, if the packet is forwarded to a faulty channel with crosstalk fault, it has a ~90% possibility of being corrupted and ~10% possibility of being lost (e.g. due to being overwritten inside the buffer, corrupted channels, soft errors) based on the study performed in [21]. Lost packets never reach their destination while corrupted packets do however the integrity of the packet payload is no longer guaranteed. Therefore, the number of lost (L_p) and corrupted packets (C_p) can be calculated by (6). The fault model can be selected in the software interface to allow performance evaluation under different fault conditions.

$$L_p = \begin{cases} 0, stuck-at\ fault \\ 0.1 * F, crosstalk\ fault \end{cases}$$
$$C_p = \begin{cases} F, stuck-at\ fault \\ 0.9 * F, crosstalk\ fault \end{cases} \quad (6)$$

The traffic heat map, throughput and the number of lost/corrupted packets provides efficient metrics to analyse the fault-tolerant capability of a NoC system. The current version of the tool supports mesh and torus topologies however other topologies can be supported via minimal modification. An example is given in the next section to illustrate how the physical hardware monitoring mechanism provides data to the software for analysis.

V. A CASE STUDY AND EVALUATION

This section presents an example case study using the proposed monitoring mechanism. Fig. 6 shows the monitor

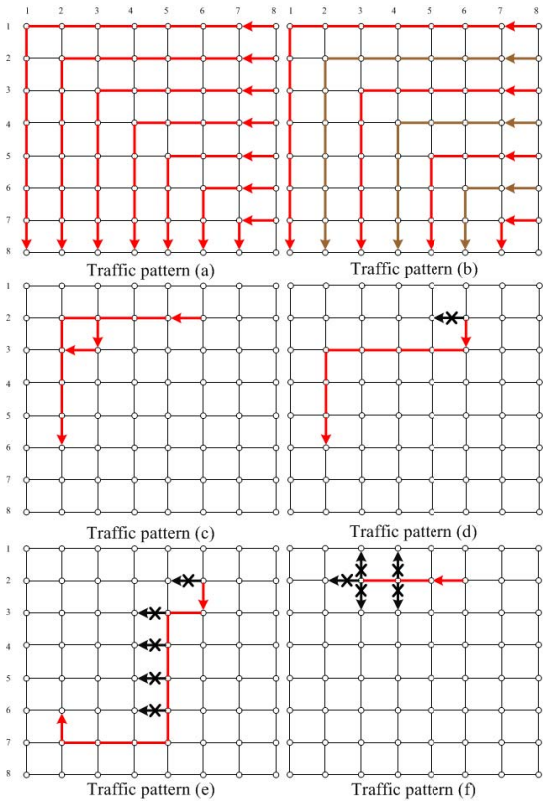

Fig. 7. Traffic heat map of different traffic patterns

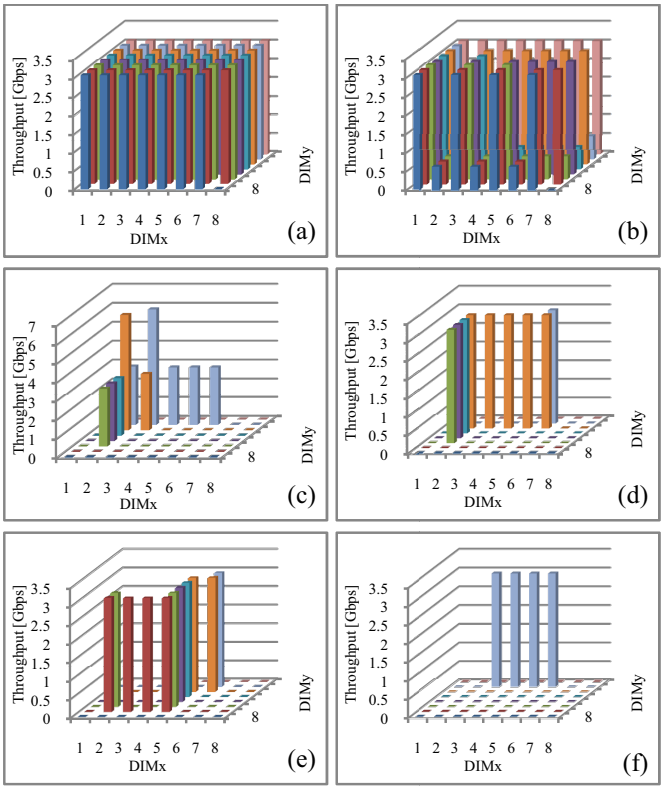

Fig. 8. The throughput of NoC routers displayed in 3-D column

software interface which runs on a host PC. Its main role is to provide quantitative analysis of the NoC performance, e.g. evaluate the NoC routing policy under real-time traffic and fault conditions. The software provides several facilities including 1). a control panel where the NoC system size and running time are defined; 2). a traffic heat map which is generated according to the NoC traffic; 3). average throughput calculation and 3D display of throughput, and 4). the number of lost/corrupted packets calculation to define the impact from faults in the NoC. Note faults are injected into the NoC channels using fault injectors [8], [22] and are not facilitated via the software interface. In this approach, the UART is employed as it has a small area overhead.

In this paper a case study on an 8x8 (64 cores) NoC system of 2D-mesh topology was implemented in the Altera DE4 development board (Stratix EP4SGX530KH4C2 FPGA). Each router implements a fault-tolerant NoC routing algorithm [8] developed by the authors. At the local port of each router, a PE is attached which encompasses a packet generator. The packets generator generates the packets according to the packet layout shown in the Fig. 1 (b) and controls the number of packets issued to the local router. Six traffic patterns listed in Table I were employed to evaluate the monitoring mechanism in this case study. These traffic patterns were chosen as they can evaluate the capability of congestion-avoid and fault-tolerance of the NoC system. All 6 traffic scenarios were injected into the NoC on the FPGA, executed in real-time and the resultant statistics data transfer to the software for analysis. Traffic pattern (a) presents a transpose traffic where node (8,i) sends packets to (i,8), i ∈ [1,7]; (b) presents a different varied packet injection rate (PIR) of 20 and 1 respectively (i.e. the packets are sent every 20 and 1 clock cycles); (c) present a congested traffic where (6,2) and (3,2) send packets to (2,6) and (2,3) respectively. A congestion will occur in the western output channel (WC) of (3,2); (d) presents a traffic pattern with a single faulty channel. Node (6,2) sends packets to (2,6) and WC of (6,2) is faulty; (e) presents a traffic pattern with multiple faulty channels. The source/destination nodes are the same as (d). However, the WCs of (6,2), (5,3), (5,4), (5,5), (5,6) are faulty and (f) presents a clustered faulty region. The northern output channel (NC) and southern output channel (SC) of (3,2) and (4,2) are faulty. The WC of (3,2) is also faulty. Note that 1). The PIR is 1 for all the traffic patterns except traffic pattern (b), which means the packets are sent every one clock cycle; 2). These traffic patterns are chosen to illustrate the proposed mechanism clearly however the

TABLE I. THE TRAFFIC PATTERNS OF BENCHMARKING

#	**Traffic patterns**
(a)	(8,*)->(*,8)
(b)	(8,*)->(*,8), varied PIR
(c)	(6,2)->(2,6), (3,2)->(2,3)
(d)	(6,2)->(2,6), WC(6,2) is faulty
(e)	(6,2)->(2,6), WC(6,2), WC(5,3~6) are faulty
(f)	(6,2)->(2,6), NC&SC(3~4,2), WC(3,2) are faulty

performance analysis of more complex traffic patterns can be found via the open source website, see the section conclusion and 3). The traffic patterns in Table I can be used to evaluate the system behaviours under different traffic scenarios, such as different packets injection rate, idle and congestion, fault-free and faulty, distributed and clustered faulty channel distribution. The traffic patterns are generated by traffic generators which are attached to the NoC on the FPGA; therefore new traffic scenario can be added to the NoC using modified traffic generators.

When the data is presented to the software from the hardware monitor, the traffic heat map structure, shown in Fig. 7, is generated to enable visual analysis of the system performance. It can be seen that in the traffic pattern (a) node (8,i) sends packets to (i,8) where $i \in [1,7]$. The shape of communication paths are at right angles. The packets are forwarded along the x-dimension first and then y-dimension as the routing algorithm [8] follows the XY routing policy if there is no congestion or faults are not detected; (b) the channel colour is varied as the PIR varies in rate. For example, the colour of communication path from (8,2) to (2,8) is lighter than from (8,1) to (1,8) as the former PIR is 20 and the latter PIR is 1; (c) a congestion occurs in the western output channel of node (3,2) as (3,2) and (6,2) both forward the packets to the west port. The NoC routing algorithm [8] distributes traffic to the south port to relax the congestion instead of waiting at the west port; (d) it forwards the packets to the south port of node (6,2) as the WC is faulty; (e) the algorithm [8] forwards the packets to the fault-free port in the system with distributed faulty channels and the packets are delivered to the destination eventually, and (f) the packets are forwarded to a dead-end as routing algorithm [8] under evaluation does not have global traffic statuses of the nodes (e.g. >3 hops away). Based on the traffic heat map, the reliability of the NoC system can be analysed in a visualized manner, e.g. the congestion-avoid capability in the traffic pattern (c) and the fault tolerant capability in (d)-(f) can be viewed. This enables first order analysis of the effectiveness of the adaptive routing algorithm or any NoC routing policy to be evaluated under real-time traffic and fault conditions.

The throughput of each traffic pattern is presented in Fig. 8 and show the 3D column-format display. The 3D throughput display is used to alow the throughput of every node to be viewed and also provides a throughput distribution of the entire NoC system. The x/y-axis values are the router ID in the x/y dimension and the z-axis corresponds to the throughput of the router. It can be seen that for the traffic pattern – (a) each router has the same throughput as all the routers receive the same amount of packets. The node (8,8) does not send packets to itself, therefore the throughput of router (8,8) is zero; (b) two kinds of throughput are presented as the PIR is varied; (c) the throughput of router (2,3) and (3,2) are higher than others because they are the hotspots and lots of traffic pass through them, and in (d)~(f) the throughput of each node are present. Every node in the communication path has the same throughput. The throughput diagram provides several benefits, e.g. the throughput of each router can be obtained, the traffic of routers can be compared and analysed especially for the congested and faulty scenarios. It also provides real-time data on the entire NoC system performance and thereby aids system design in evaluating routing algorithm design and application mapping.

Average throughput T_{avg}, under traffic patterns (a)~(f) are presented in Table II. The data in the table illustrates that (1). the traffic pattern (a-c) have a high average throughput as they have more communicating nodes compared to other traffic patterns; (2). the average throughput of (e) is greater than (d) for the same source/destination node as the total throughput sum of (e) is greater than (d) and (3). for traffic pattern (f), the number of lost/corrupted packets is 25/230, 0/255 for crosstalk and stuck-at fault models, respectively. This outcome is expected as the example algorithm [8] does not have access to the global traffic status of nodes. From the traffic heat map and the throughput values presented in Fig. 7, Fig. 8 and Table II, it can be seen that the impact from clustered faults results in packets being forwarded to a faulty channel where they are either lost or the payload is corrupted and further forwarded to incorrect destination routers. Therefore, we can see under such fault conditions that the average throughput is low. The proposed monitoring mechanism and software interface enables the analysis of such fault scanerios in assessing the fault tolerant capabilities and effectiveness of routing policy in the NoC.

TABLE II. THE AVERAGE THROUGHPUT, THE NUMBER OF LOST/CORRUPTED PACKETS

Traffic	T_{avg} [Gbps]	Stuck-at		Crosstalk	
		L_p	C_p	L_p	C_p
(a)	3.012	0	0	0	0
(b)	1.979	0	0	0	0
(c)	0.573	0	0	0	0
(d)	0.430	0	0	0	0
(e)	0.526	0	0	0	0
(f)	0.191	0	255	25	230

For the example 8x8 2D-mesh NoC system, the area overhead of each component is presented in Table III based on an Altera FPGA device and Quartus II software. The hardware monitor probe only occupies 87 LC Combinational (LCCs) and 125 LC Registers (LCRs). The STAT node occupies 146 LCCs and 184 LCRs. The entire NoC system only needs one STAT node which is connected to a router. The two area overheads are used to evaluate the compactness of the monitoring mechanism – one is the probe area overhead of a single router (A_p) and the other is the monitoring mechanism area overhead of the entire NoC system (A_m). Therefore, the area overhead of A_p is 11.27% (87/772=11.27%) for LCCs and 13.27% for LCRs; A_m is 5.93% ((87*DIMx*DIMy+772+146)/109,451= 5.93%, where DIMx=8, DIMy=8) for LCCs and 6.01% for LCRs, i.e. A_p is ~12.27% and A_m is ~5.97% (right of Table III). To benchmark the monitor mechanism we can compare area overhead against an existing state of the art approach [18]. For example, the area overheads of [18] (using 4 multipurpose probes) are $A_p = 55\%$, $A_m = 31\%$. This demonstrates that [18] has a large area overhead as its architecture dictates that four type event detectors are included. The area overhead of our approach is relatively low

and it has the added advantage of monitoring fault tolerance capabilities and providing analysis. The low area overhead of monitors and analysis of fault tolerance capabilities are very beneficial for evaluating large-scale NoC in the era of reliable systems.

TABLE III. THE AREA OVERHEAD OF MONITORING MECHANISM

Component	LCCs	LCRs	Approach	A_p (%)	A_m(%)
Monitor probe	87	125	[18]	55	31
NoC Router [8]	772	942			
STAT node	146	184	This paper	12	6
8x8 2D-mesh NoC	109,451	151,853			

VI. CONCLUSION

A monitoring mechanism was proposed in this paper and the novelty lies in exploiting the monitor probe to detect the events in the NoC router and employing the STAT node to transmit the statistics data to computer. In addition, the key aspect of the proposed approach is the quantitative analysis of fault tolerant capability and the traffic heat map generation to reflect the hardware adaption capability visually for the NoC system. Some metrics are provided by the monitor software to evaluate the NoC system performance, such as throughput, traffic heat map and the number of lost/corrupted packets. An 8x8 (64 cores) NoC system with a fault tolerant routing algorithm was evaluated for the case study. The results showed that the monitoring mechanism can evaluate the system performance successfully, especially the routing policy analysis. Moreover, the area overhead of the monitoring components in the NoC is low, only ~6% which does not prohibit NoC scalability. The monitoring software (Open Source) can be found at http://isrc.ulster.ac.uk/jliu/research.html. It can easily be used for other NoC platforms as long as the data exchange format between the NoC and computer are consistent. Future work aims to add more functions to the monitor software and to decrease the area overhead of monitoring components further.

ACKNOWLEDGMENT

Junxiu Liu is supported by the University of Ulster's Vice-Chancellor's Research Scholarship. This research is also supported in part by Spanish grant (with support from the European Regional Development Fund) BIOSENSE (TEC2012-37868-C04-02).

REFERENCES

[1] L. Benini and G. De Micheli, "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, 2009, pp. 761–766.

[3] M. Amin, M. Tagel, G. Jervan, and T. Hollstein, "Design Methodology for Fault-Tolerant Heterogeneous MPSoC under Real-Time Constraints," in 7th International Workshop on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC), 2012, pp. 1–6.

[4] A. Agarwal, B. Raton, C. Iskander, *et al.*, "Survey of Network on Chip Architectures & Contributions," Journal of Engineering, Computing and Architecture, vol. 3, no. 1, pp. 1–15, 2009.

[5] A. Das, A. Kumar, and B. Veeravalli, "Fault-tolerant Network Interface for Spatial Division Multiplexing based Network-on-Chip," in 7th International Workshop on Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC), 2012, pp. 1–8.

[6] C. Grecu and A. Ivanov, "Testing Network-on-chip Communication Fabrics," IEEE Transactions on CAD of Integrated Circuits and Systems, vol. 26, no. 12, pp. 2201–2214, 2007.

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

[8] J. Liu, J. Harkin, Y. Li, and L. Maguire, "Online Traffic-Aware Fault Detection for Networks-on-Chip," Journal of Parallel and Distributed Computing, vol. 74, no. 1, pp. 1984–1993, 2014.

[9] P. Lotfi-Kamran, A. M. Rahmani, M. Daneshtalab, A. Afzali-Kusha, and Z. Navabi, "EDXY - A low cost congestion-aware routing algorithm for network-on-chips," Journal of Systems Architecture, vol. 56, no. 7, pp. 256–264, Jul. 2010.

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

[11] 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.

[12] 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.

[13] S. Carrillo, J. Harkin, L. McDaid, *et al.*, "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.

[14] S. Carrillo, J. Harkin, L. J. McDaid, *et al.*, "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.

[15] A. Alhonen, E. Salminen, J. Nieminen, and T. D. Hämäläinen, "A Scalable, Non-Interfering, Synthesizable Network-on-Chip Monitor," in NORCHIP, 2010, pp. 1–6.

[16] O. Hammami, X. Li, and J.-M. Brault, "NOCEVE : Network On Chip Emulation and Verification Environment," in Design, Automation & Test in Europe Conference & Exhibition (DATE), 2012, pp. 163–164.

[17] A. Ben Ahmed, T. Ochi, S. Miura, and A. Ben Abdallah, "Run-Time Monitoring Mechanism for Efficient Design of Application-Specific NoC Architectures in Multi/Manycore Era," in 7th International Conference on Complex, Intelligent, and Software Intensive Systems, 2013, pp. 440–445.

[18] L. Fiorin, G. Palermo, and C. Silvano, "A Configurable Monitoring Infrastructure for NoC-Based Architectures," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, pp. 1–5, 2013.

[19] J. Harkin, F. Morgan, S. Hall, *et al.*, "Emulating Biologically Inspired Architectures in Hardware: A New Reconfigurable Paradigm for Computation," in Reconfigurable Communication-Centric Systems-on-Chip Workshop, 2008, pp. 10–19.

[20] A. Jimenez-Fernandez, G. Jimenez-Moreno, A. Linares-Barranco, et. al, "A Neuro-Inspired Spike-Based PID Motor Controller for Multi-Motor Robots with Low Cost FPGAs," Sensors, 12(4), pp.3831-3856, Jan. 2012.

[21] 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.

[22] M. Birner and T. Handl, "ARROW - A Generic Hardware Fault Injection Tool for NoCs," 12th Euromicro Conference on Digital System Design, Architectures, Methods and Tools, pp. 465–472, Aug. 2009.