Microprocessors and Microsystems 39 (2015) 358–372

Contents lists available at ScienceDirect

Microprocessors and Microsystems

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

Low cost fault-tolerant routing algorithm for Networks-on-Chip

Junxiu Liu [a,*], Jim Harkin [a], Yuhua Li [b], Liam Maguire [a]

[a] School of Computing and Intelligent Systems, University of Ulster, Derry, Northern Ireland, UK
[b] School of Computing, Science and Engineering, University of Salford, Manchester, UK

ARTICLE INFO

Article history:
Available online 11 June 2015

Keywords:
Networks-on-Chip
Adaptive routing
Fault tolerance
Scalability
Hardware reliability

ABSTRACT

A novel adaptive routing algorithm – Efficient Dynamic Adaptive Routing (EDAR) is proposed to provide a fault-tolerant capability for Networks-on-Chip (NoC) via an efficient routing path selection mechanism. It is based on a weighted path selection strategy, which exploits the status of real-time NoC traffic made available via monitor modules. The key performance goal is to maintain throughput under congested and faulty conditions via effective routing path decisions. In the proposed EDAR, port weights are calculated in real-time according to the channel status – Idle/Busy/Congested/Faulty, and the port with the lowest weighting is ranked as the near-optimal route to forward packets. This mechanism enables the router to bypass congested ports and tolerate faulty ports. To assess the latency and throughput of the proposed routing algorithm, several traffic patterns for both fault-free and faulty NoCs were evaluated. Results show that EDAR can achieve higher throughput compared to other state of the art routing algorithms under various traffic patterns and levels of injected faults. In addition, the hardware area overhead for EDAR is demonstrated to have a reasonably low cost which maintains scalability for large NoC implementations.

1. 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. Communication latency and fault tolerance are challenges [1,2] for modern NoCs due to complex and large-scale application mapping structures for on-chip multiprocessors. In particular NoC reliability due to the increase in physical defects in advanced manufacturing processes is a critical challenge as often faults occur post manufacturing.

The NoC communication latency is mainly determined by the routing algorithms. Adaptive routing algorithms need to address contention with busy/congestion traffic detectors however, to address the challenge of reliability and provide fault tolerance, current algorithms must consider not only traffic loads but also permanent and/or dynamic faults in the NoC. Typically permanent faults exist for the life-time of the system and cannot be recovered; and external electrical fluctuations and radiation can cause transient faults, whilst unstable hardware can cause intermittent faults, such as stuck-at, bridging, crosstalk, single event upset and single event transient faults. 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 1000 defects/m^2 [3,4]. Testing results have demonstrated that even small numbers of faults in the NoC, e.g. ~30 logic gate faults, can cause between 5 and 50 faulty channels, which highly impairs the NoC network [5]. When a channel is faulty, the adaptive routing algorithms should choose a fault-free path to forward the packets and avoid corrupting packet data. Therefore, to ensure modern NoCs are able to accommodate faults and maintain operation post-manufacturing requires the development of autonomous systems that can make decisions according to the real-time traffic conditions. One step towards this is the development of an optimal path selection strategy which can operate under the presence of faults.

1.1. Contribution of this work

In this paper, the EDAR algorithm is presented which aims to minimise the degradation of NoC throughput, via dynamic routing, in the presence of faults. The approach is novel as it employs an area-efficient weight-based port selection mechanism to identify reliable and low-congested routing paths. The EDAR routing algorithm defines different weight values for the various traffic conditions (i.e. Busy/Congested/Faulty) where the busy and faulty

* Corresponding author.
E-mail addresses: liu-j4@email.ulster.ac.uk (J. Liu), jg.harkin@ulster.ac.uk (J. Harkin), y.li@salford.ac.uk (Y. Li), lp.maguire@ulster.ac.uk (L. Maguire).

http://dx.doi.org/10.1016/j.micpro.2015.06.002
0141-9331/

conditions are defined as the smallest and greatest weight value, respectively. Each NoC router port is weighted with the lowest weighting ranked as the near-optimal port to forward packets. By calculating the weights for the ports, routing decisions can be made to avoid faulty or congested channels between router ports. Therefore the EDAR can bypass faults in the NoC system and has the capability to make optimal routing decisions according to complex traffic conditions. Good system performance is obtained as routing decisions balance the overall traffic load by identifying the best ports under given traffic and fault conditions to maximise overall NoC throughput.

The main contributions of this paper include:

- EDAR: A novel fault-tolerant and congestion-aware routing algorithm centred on a weight-based port selection mechanism.
- Results and detailed performance analysis of latency, throughput and fault-tolerant capability of EDAR. Validation of these results against benchmarks.
- Validation of an area-efficient hardware implementation of the EDAR routing algorithm.

The remainder of the paper is organised as follows. Section 2 provides a summary of the previous work with a focus on the dynamic routing algorithm. Section 3 discusses the proposed novel EDAR algorithm and gives the process of weight calculation and routing decision making in detail. Sections 4 and 5 present results and a performance analysis on different traffic patterns and fault-tolerant ability for a range of experiments. Section 6 discusses the hardware implementation for EDAR using ASIC technology and presents an area overhead comparison with previous work. Section 7 provides a conclusion and highlights future work.

2. Background and previous work

Dynamic routing algorithms make decisions adaptively according to the traffic status. In this section, a review of current approaches is presented which include congestion-aware and fault-tolerant dynamic routing algorithms.

2.1. Congestion-aware dynamic routings

Previous congestion-aware dynamic routings were investigated by various methods which are presented as follows. (a) *Combined routing algorithms*. A dynamic routing scheme, namely DyAD, can switch between the deterministic and adaptive routing algorithms based on the system congestion conditions, i.e. combined the advantages of these two routing algorithms [6]. Similarly, a DyXY adaptive routing algorithm made the routing decisions based on congestion conditions, and achieved a better performance than XY and Odd–Even routing [7]. Another selection strategy – PathAware selects the appropriate output port after getting the candidates from West-First or Odd–Even routing [8]. (b) *Using dedicated network or signals to collect traffic status*. A Neighbour-on-Path (NoP) routing algorithm used dedicated wires to sense the traffic status of neighbouring nodes [9]. It had a performance improvement on average delay and throughput, especially under the heavy traffic workloads. A traffic-aware NoC router used dedicated signal wires to indicate the status of channel traffic, and was adaptive to traffic congestion [10]. Enhanced dynamic XY routing algorithm [11] added two dedicated wires per channel, which indicates the congestion status of the channels in the same row or column as the current router. These congestion wires enabled the router to avoid a congested neighbouring path. A path-aware routing scheme employed a congestion aware sub-network to propagate the global traffic information and aid making routing decisions [12]. The main drawback is that it requires additional area overhead as it uses another sub-network to broadcast the traffic information. (c) *Piggyback routing*. A weighted priority deflection policy was proposed in [13] using the piggyback mechanism. Each packet has a weight which is calculated based on the age, distance and initial priority of the packet. This priority-based deflection policy achieved a good performance for most cases. It was only evaluated using a SystemC simulator and did not present hardware performance for scalability which is a key challenge to the success of any NoC strategy. Including piggyback routing, several indirect adaptive routing algorithm were proposed in the approach of [14]. They provided a good performance under steady-state and transient loads on the dragonfly topology. (d) *Table based routing*. The centralized adaptive routing mechanism was introduced in [15,16] where a routing table is employed at each node. It utilised a feedback module to monitor global traffic status and a control module to make routing decisions. It has the opportunity to be out of data if the traffic status changes during the source to destination propagation time of packets. A regional congestion awareness routing algorithm, namely RCA, makes routing decisions based on not only the local traffic information but also the regional congestion information [17]. The drawback is that the dedicated wire overhead is significant. In order to reduce the overhead, a global congestion awareness (GCA) routing algorithm [18] embedded the traffic status information in packet headers. After received the traffic status information, each node updates a congestion map including the link status of all the nodes in the system; it then finds the shortest path to forward the packet. The aforementioned congestion-aware dynamic routing algorithms can (1) avoid congested paths in the network; (2) have the ability to balance the traffic load and (3) achieve a higher throughput than typical deterministic routing especially for heavy traffic workloads. However, these algorithms do not function under fault conditions. It is very important for NoC routing algorithms to support fault tolerance and be able to make effective routing decisions in maintaining system performance.

2.2. Fault-tolerant dynamic routings

A reconfigurable routing algorithm in the approach of [19] has the ability to bypass the faulty router after the router is detected as faulty and deactivated. However, it is only tolerant of faulty routers (node) and cannot support faulty channels. A small-granularity fault-tolerant routing algorithm was proposed to support node and link faults [20]. It provided a fault-tolerant policy although the number of faulty links per router was limited to one. Dynamic routing schemes of ARIADNE [5] and uDIREC [21] employed routing tables to aid making decisions, where adjacent routers inform all other routers about any faulty links/routers and update routing tables with the identified failures. The Gradient fault-tolerant routing scheme [22] modelled the NoC across different zones. Routing directions are then established according to the zone where the destination node is located. A fault-tolerant adaptive routing algorithm [23] can forward the packets to the destination via the intermediate nodes to bypass the faulty links. The relative positions of source, destination and intermediate nodes are constrained. It cannot deal with complex fault patterns and did not present the hardware implementation results.

A fault-tolerant routing algorithm in the approach of [24] employed a localised re-routing to bypass the fault links and regions. Two fault-tolerant routing algorithms, FTDR, FTDR-H used routing tables to store the distance for every directions between the current and destination nodes [25]. The routing table are updated when the link status changes (e.g. from fault-free to faulty). Then the current node can choose a fault-free path to forward the packets. As the routing table of FTDR is not

area-efficient, the FTDR-H was developed which used a hierarchical structure to reduce the size of the routing table. The entire NoC system was divided into several regions and a separated routing table was used in each region. It has the key constraints that a faulty link has to be shutdown bi-directionally and two different regions must be connected. A Look-Ahead-Fault-Tolerant (LAFT) routing algorithm was proposed for the 3D NoC system [26]. LAFT receives the fault status from the immediate neighbouring nodes and selects the routing path based on this information. If there are several candidates, the path with minimum distance and large diversity is chosen. For most of these fault-tolerant approaches, the main drawback is that they do not provide an efficient congestion control mechanism to make optimal routing decisions for complex traffic conditions in order to maintain system performance under heavy traffic loads.

2.3. Summary

Current approaches of dynamic routing algorithms are summarised in Table 1. They have the aforementioned weaknesses of (a) unable to make routing decisions under complex traffic conditions and (b) the system throughput performance is degraded if faults occur. Therefore, they do not meet the required characteristics to provide an efficient routing strategy for modern NoC. For a NoC to be robust and that can achieve better performance and tolerate faults, two key functions need to be investigated: (a) the ability to avoid congested paths and balance the traffic workloads, and (b) the ability to tolerate faults and to proceed to provide system functionality in the event of a physical impairment. This paper investigated such a routing strategy as efficient adaptive routing and fault-tolerance requirements are of paramount importance with the ever increasing density of large scale electronic systems.

3. EDAR routing algorithm

In our previous work [10,27–29], the authors developed a NoC router design, namely EMBRACE, which demonstrated a traffic-aware, online fault testing capability, with Monitor Module (MM) components that indicate channel traffic states to neighbouring NoC nodes. However, this paper presents an extension to this work where a novel fault-tolerant routing algorithm (EDAR) is used to identify reliable and low-congested routing paths in order to minimise the degradation of NoC throughput in the presence of faults. This section outlines the EMBRACE router architecture and presents the EDAR routing algorithm in detail.

3.1. EMBRACE routing architecture

In this paper, a 2D-mesh topology is used to illustrate the principle of the EDAR algorithm as it is the most common topology used in many applications [8,11,19,22]. Fig. 1 presents a typical 2D-mesh NoC system, where each node is connected to other nodes through four directions (N/E/S/W) and processing elements are connected to the router via a local port. Every node is positioned using a pair of coordinates. The notation (x_s, y_s) is used to denote the coordinates of the source node which issues packets; (x_c, y_c) denotes the coordinates of the current router where the packet is located; (x_d, y_d) denotes the coordinates of the destination node, the final target node.

Table 1
Previous works summary.

The approach	Congestion aware	Fault detection	Fault-tolerant routing
[9,6,10,7,8,12–18]	√	×	×
[26,22,5,19,20,23,24]	×	×	√
[11]	√	×	√
[21]	×	√	√
[43]	√	√	√

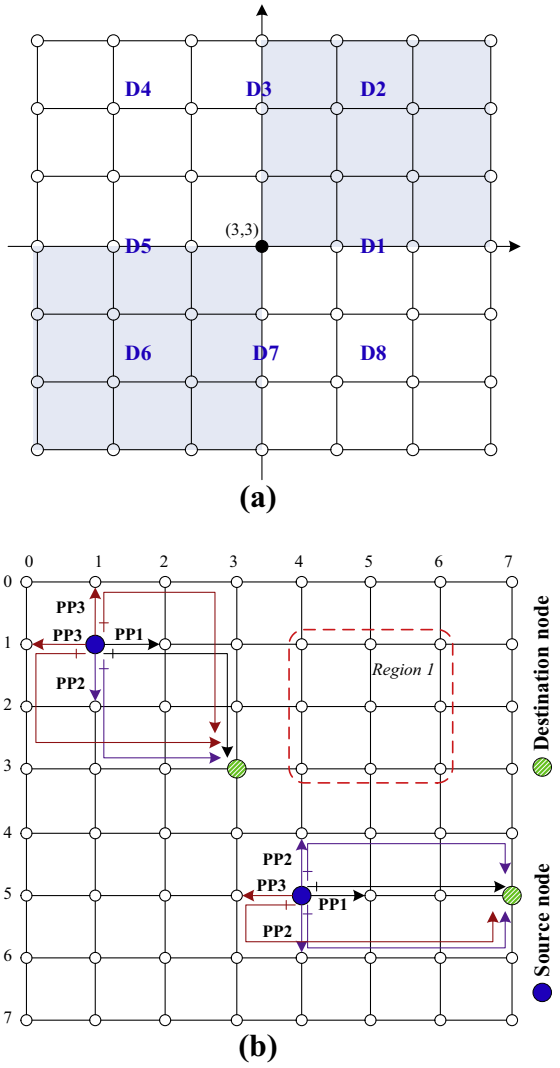


Fig. 1. 2D-mesh NoC system. (a) Relative directions between source node and destination node; (b) different preferred port definition.

In Fig. 1(a), assume node (3,3) is the current node (i.e. $(x_c, y_c) = (3,3)$), the destination node can be in the 8 directions denoted by direction D1–D8, i.e. from east–to south–east. However, if the coordinates of the destination node is equal to the current node (i.e. $(x_d, y_d) = (3,3)$), it indicates that the packet has arrived at the destination and should be forwarded to the local port. When the destination node is located in D1–D8, it can classed as two types – (1) diagonal position (i.e. D2, D4, D6, D8) and (2) direct position (i.e. D1, D3, D5, D7). In each type, one example is given to illustrate the preferred ports definition. The preferred ports are defined as the ports which the current node should preferably choose to forward the packets to the destination. The top part of Fig. 1(b) presents the destination in the *diagonal position* where the current node is (1,1) and the destination node is (3,3). The four ports N/E/S/W at node (1,1) are classed as three levels. The east is defined as Preferred Port 1 (PP1) port, the south is defined as Preferred Port 2 (PP2) and the west/north ports are both defined as Preferred Port 3 (PP3). The levels are set in this ranking

due to the number of hops required to transmit packets to the destination, i.e. it is smaller from the E and S directions than from W and N. The east port is defined as a higher level ranking than the south port as the EDAR algorithm firstly chooses the direction on the *x*-axis and then *y*-axis under the same traffic conditions. The bottom right of Fig. 1(b) illustrates the levels when the destination node is in the *direction position*. Note, in this example N and S are assigned the same level (PP2) as both have the same number of hops (i.e. 5 hops) to the destination node.

To understand how the nodes are being connected, the EMBRACE router architecture is introduced. Fig. 2(a) presents the EMBRACE router structure and the logical connections between the components. The Adaptive Routing Scheme (ARS) module is the core component for the router as it receives the traffic status information (i.e. Busy/Congested/Faulty) from all the four neighbouring nodes and makes routing decisions adaptively according to the channel situation. The Adaptive Arbitration Policy (AAP) module controls the output port selection and provides access arbitration for multi-requests for the same output channel. The packets go to a crossbar and are forwarded to the selected output port finally. The Monitor Module (MM) provides traffic status information for the channels [27], where each channel has one MM (Fig. 2(b)). The number of free slots (F_s) in the buffer of the receiver (RX) side is used to gauge the status of channel traffic. For example, if $F_s = 0$, the buffer is full and the channel is said to be 'Congested'. If $F_s \leqslant Threshold\_v$, the channel is said to be 'Busy', where *Threshold_v* denotes a threshold value (normally half the size of the buffer). However, for the 'Faulty' status, the MM in the transmitter (TX) side sends test vectors to the MM in the receiver (RX) side, and it compares values received from the outcome of the test vector with pre-defined values. If they do not match, the present channel under test is classed as faulty and a fault flag is raised to inform the TX router of a fault in the channel interconnect. The fault detection mechanism in our previous work of [27] is used in this paper where temporary and permanent faults can be detected promptly and the real-time detection results are fed to the EDAR routing algorithm to aid making the routing decisions. Therefore, the three traffic status signals from each direction are defined as Busy/Congested/ Faulty. Each group of 3 status signals are connected to the router's ARS to aid in making more effective routing decisions.

3.2. EDAR routing algorithm

The objective of the EDAR routing algorithm is to route the packets along the minimum congested path while bypassing identified faulty interconnections. For any given 2D-mesh NoC, when $(x_d, y_d) = (x_c, y_c)$, this signals that a packet has arrived at its destination node and can be forwarded to the node's processing element through the local port. If $(x_d, y_d) \neq (x_c, y_c)$, then the current node forwards the packets to its neighbouring node through N/E/S/W directions. EDAR uses a weight calculating mechanism to calculate the weights for each direction. The weight values to be calculated include (1) the direction priority weight w_p, (2) the channel Busy status weight w_b, (3) the channel Congested status weight w_c and (4) the channel Faulty status weight w_f. The direction priority weight w_p is determined by the relative direction between the current and destination nodes and the preferred port level. The pseudo code in Fig. 3 illustrates how the w_p is calculated for each direction. In this approach, if the port is a PP1 port, the w_p of this port is equal to 1; if it is a PP2 port, $w_p = 2$ and if it is a PP3 port, $w_p = 3$. Therefore, a lower direction priority weight value corresponds to a preferred port.

The weights ($w_b/w_c/w_f$) of the channel Busy/Congested/Faulty statuses are determined by the "B/C/F" input signals from the MM module, as shown in Fig. 2(b). If the status, *s*, of a channel is busy, then $s_b = 1$; if the channel is congested, status $s_c = 1$ and if the channel is faulty, status $s_f = 1$. Using this status information, the values of $w_b/w_c/w_f$ can be calculated using (1). It can be seen that $w_f > w_c > w_b$ when $s_f = s_c = s_b = 1$. The weight w_f is given precedence as the channel status of Faulty has the most significant performance impact on a channel. Similarly, Congested has more impact than Busy.

$$\begin{cases} w_b = \begin{cases} 0, s_b = 0 \\ 2, s_b = 1 \end{cases} \\ w_c = \begin{cases} 0, s_c = 0 \\ 3, s_c = 1 \end{cases} \\ w_f = \begin{cases} 0, s_f = 0 \\ 10, s_f = 1 \end{cases} \end{cases} \tag{1}$$

The values of $w_p/w_b/w_c/w_f$ are calculated based on the following four rules: (1) if the PP1 is fault-free, not congested and not busy, it will be chosen as preferred output port; (2) if the PP1 is faulty or congested, PP2 is chosen as the preferred port. The same rule applies for the PP2 and PP3 cases; (3) if the PP1 is busy and the PP2 is fault-free and idle, the PP2 is chosen as the preferred port. The same rule applies for the PP2 and PP3 cases; (4) the weight value should be as small as possible to allow compact hardware implementations of the EDAR.

After the weight values of w_p, w_b, w_c and w_f are generated, the total weight, *W*, for each port i ($i \in \{N, E, S, W\}$), is calculated by (2). The output port with the minimal weight value is selected as the final output port.

$$W[i] = w_p[i] + w_b[i] + w_c[i] + w_f \tag{2}$$

In summary, the EDAR routing algorithm is described by the pseudo code shown in Fig. 4. The inputs are coordinates of destination (x_d, y_d) and current nodes (x_c, y_c), the traffic status of B/C/F on all directions; the output is the selected channel to forward the packets. First, the weight values of w_p, w_b, w_c, w_f and *W* are calculated (line 11–24); then the *W*[*i*] with minimum value is selected and the corresponding *i* is the selected output channel number (line 26–32). The EDAR routing module is implemented in hardware using a combinational logic circuit. If the inputs change, the output is generated immediately.

To illustrate the application of the rules for path selection, consider region 1 (2D 3×3 array) shown in Fig. 1(b). For the disconnected direction of edge node, the busy, congested and faulty lines are set to be high. EDAR forbids packets to be forwarded to these directions, such as north of node $(*,0)$, east of $(7,*)$, south of $(*,7)$ and west of $(0,*)$. Fig. 5 provides an expanded view of region 1 to demonstrate how the EDAR routing algorithm operates. Assume that node (5,1) is the source node, (6,3) is destination node; then the nodes of (6,1), (6,2), (5,2) and (5,3) are the current nodes which packets pass through. When all the links are fault-free and not congested, the communication path is shown as Path #1. However, if the channel between the router (5,1) and (6,1), i.e. Link #1, becomes faulty, the EDAR in router (5,1) will re-calculate the port weights as shown by (3). As a result, the EDAR will choose the South direction as the new preferred output port as it has the minimum weight out of all four (*N*, *E*, *S* and *W*) ports. The port with minimum weight indicates this port has a better traffic conditions than others. Here, the south port is fault-free and idle, which is better than the east port to forward the packets.

$$\begin{cases} W[N] = w_p[N] + w_b[N] + w_c[N] + w_f[N] = 3 + 0 + 0 + 0 = 3 \\ W[E] = w_p[E] + w_b[E] + w_c[E] + w_f[E] = 1 + 0 + 0 + 10 = 11 \\ W[S] = w_p[S] + w_b[S] + w_c[S] + w_f[S] = 2 + 0 + 0 + 0 = 2 \\ W[W] = w_p[W] + w_b[W] + w_c[W] + w_f[W] = 3 + 0 + 0 + 0 = 3 \end{cases} \tag{3}$$

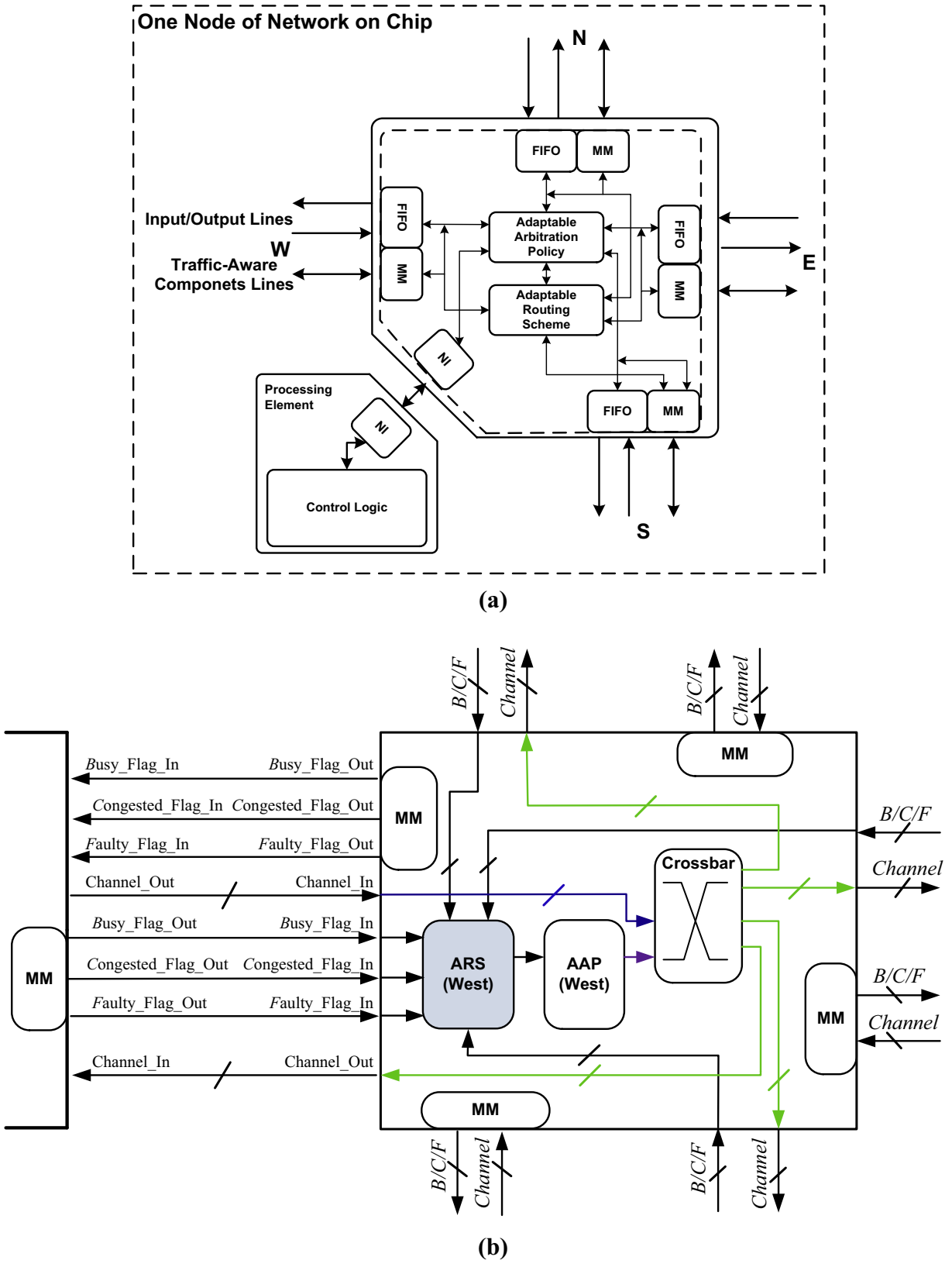


Fig. 2. EMBRACE Router. (a) Router structure; (b) signal connection (west port connection only shown).

Therefore, "Path #2" is the communication path if the Link #1 is faulty. Similarly, assume that the channel between the router (5,2) and (6,2), i.e. Link #2, is congested, in this example router (5,2) will re-calculate the port weights and the result will be $W[N/E/S/W] = \{3, 6, 2, 3\}$. In this instance the EDAR in router (5,2) will choose the South direction as the output port; therefore, "Path #3" is chosen to transmit packets. These examples illustrate that for a faulty interconnection, EDAR bypasses the faulty components to avoid packets being corrupted. Then for a congested or busy interconnection, EDAR selects another preferred idle port as

```
PSEUDO CODE.  Direction Priority Weight Calculation
01: entity DirectionPriorityWeightCalculation
02:    input  : CordinatesOfDestinationNode(Xd, Yd)
03:    input  : CordinatesOfCurrentNode(Xc, Yc)
04:    output : Wp[N/E/S/W]
05:
06: begin DirectionPriorityWeightCalculation
07:
08:    If(Xd > Xc && Yd == Yc) {                    //D1
09:       Wp[N/E/S/W] = {2,1,2,3};
10:    }
11:    elsif(Xd > Xc && Yd < Yc) {                  //D2
12:       Wp[N/E/S/W] = {2,1,3,3};
13:    }
14:    elsif(Xd == Xc && Yd < Yc) {                 //D3
15:       Wp[N/E/S/W] = {1,2,3,2};
16:    }
17:    elsif(Xd < Xc && Yd < Yc) {                  //D4
18:       Wp[N/E/S/W] = {2,3,3,1};
19:    }
20:    elsif(Xd < Xc && Yd == Yc) {                 //D5
21:       Wp[N/E/S/W] = {2,3,2,1};
22:    }
23:    elsif(Xd < Xc && Yd > Yc) {                  //D6
24:       Wp[N/E/S/W] = {3,3,2,1};
25:    }
26:    elsif(Xd == Xc && Yd > Yc) {                 //D7
27:       Wp[N/E/S/W] = {3,2,1,2};
28:    }
29:    elsif(Xd > Xc && Yd > Yc) {                  //D8
30:       Wp[N/E/S/W] = {3,1,2,3};
31:    }
32:
33: end DirectionPriorityWeightCalculation
```

Fig. 3. Pseudo code of direction priority weight calculation.

the output port. This avoids the packets waiting in the congested channel which will increase the packets delay and avoids the traffic status of busy channel becoming worse as continuously sending packets to a busy channel will make it become more congested. It can be seen that EDAR routing algorithm has the capability of being tolerant to the faults in the NoC interconnection and also adaptive to the traffic status which can reduce the average delay and balance the traffic load to achieve a better system performance.

3.3. Deadlock and livelock avoidance

Several assumptions are defined in this paper for deadlock and livelock avoidance and include: (a). a node cannot send a packet to itself; (b). a packet is absorbed when it reached its destination and (c). the source and destination nodes are in a connected region. The router architecture used in this paper is similar to a 5-stage NoC router in the approach of [30], where the technique of virtual channels (VCs) is employed to avoid deadlock. The virtual channel allocation is in stage 3 of our architecture where the packets are temporarily stored in VCs while the output channel is busy and cannot forward packets immediately. The strategies to handle the deadlock problems include deadlock prevention, avoidance and recovery. The VC implementation is one solution of deadlock avoidance. The VC is generally implemented as a stack using a first-in-first-output (FIFO) component. Fig. 6 illustrates an example of VCs. Two VCs (blue and red) between the east port of router #1 and the west port of router #2 multiplexed the same physical channel (black). The packets share the physical channel based on a flit-by-flit manner. Assume that packet P1 arrives at router #1 early and occupies the channel between the routers #1 and #2. If the packet P2 also requests access of the same channel, then the physical channel is multiplexed between the packets P1 and P2 on a flit-by-flit basis. Using the VCs, packet P2 would not be blocked while packet P1 is in transmission.

A 4×4 NoC system in Fig. 7 is used to illustrate how the VCs can be employed to avoid deadlock in this paper. The NoC in Fig. 7(a) does not have any faults and four packets (P1–P4) are forwarded to the destination nodes #3, #4, #1 and #2, respectively. Based on the preferred direction definition, the packets are forwarded through the *x*-axis first and then *y*-axis, e.g. the path of packet P1 is #1–#2–#3. The paths of all the packets were marked as different colours in Fig. 7(a). It can be seen that for the fault-free NoC systems, the EDAR is deadlock-free as the channel dependence graph is acyclic. However, if the NoC system has faulty channels, the EDAR choose an alternative direction to forward the packets; e.g. when the west channel of node #3 is faulty in Fig. 7(b), the packet P3 cannot be forwarded to the west direction so the north direction (i.e. DS2 in Fig. 7(b)) is used instead. As packet P4 is occupying the north channel of node #3, the physical channel will be multiplexed between the packets P4 and P3 on a flit-by-flit basis. The west channel of node #2 has a similar status, i.e. the packet P2 and P3 also share the same physical channel. Therefore, all the packets can arrive at the destination nodes eventually without deadlock occurring. In addition, an adaptive arbitration policy in our previous work [10] is used for the arbitration of VCs requesting the same physical channel. The arbitration combines the fairness

```
PSEUDO CODE.  EDAR ALGORITHM
01: entity EDARAlgorithm
02:    Input  : CordinatesOfDestinationNode(Xd, Yd)
03:    Input  : CordinatesOfCurrentNode(Xc, Yc)
04:    Input  : Sb, Sc, Sf[N/E/S/W]
05:    Output : Selected output channel
06:
07: begin EDARAlgorithm
08:
09:    W[N/E/S/W] = 0;
10:
11:    For i = N to W {                    // N, E, S, W
12:       calculate Wp[i] using pseudo code in Fig. 3;
13:
14:       // calculate Wb/c/f[i], using Equation (1)
15:       if (Sb/c/f[i] == 1) {            // Sb, Sc, Sf
16:          Wb/c/f[i] = 2/3/10;           // Wb, Wc, Wf
17:       }
18:       else {
19:          Wb/c/f[i] = 0;
20:       }
21:
22:       // calculate W[i] using Equation (2)
23:       W[i] = Wp[i] + Wb[i] + Wc[i] + Wf[i];
24:    }
25:
26:    min = W[N];
27:    For i = E to W {
28:       if (W[i] < min) {
29:          min = W[i];
30:          output_channel = i;
31:       }
32:    }
33:
34: end EDARAlgorithm
```

Fig. 4. Pseudo code of EDAR algorithm.

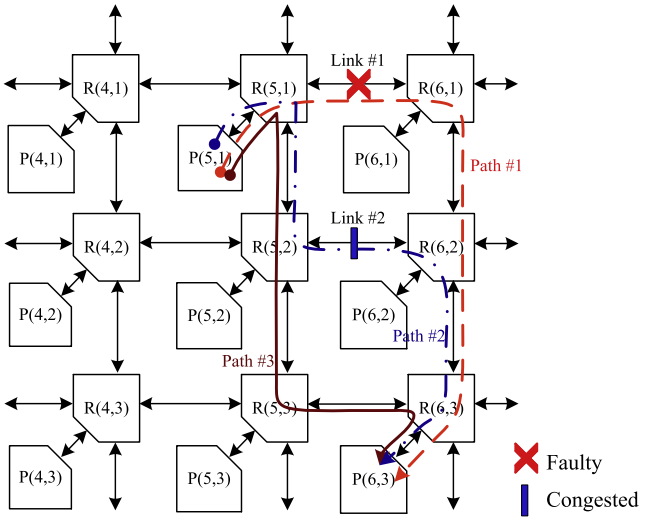


Fig. 5. Example application of the EDAR routing algorithm.

policy of a round-robin arbiter and a first-come first-served priority scheme to improve system throughput. At the same time, the free space in the VCs [31] and buffers [32] are also used to aid making routing decisions, where the fewer allocated VCs implies the multiplexing in the given link is less and therefore, a lower buffer count indicates less backpressure of the input port at the downstream node. Hence, if the preferred VC only has limited space or the input port of the preferred downstream node does not have enough space, the EDAR approach will choose an alternative direction to relieve the backpressure, e.g. if the north channel of node #3 is congested, the packet can be forward to south direction (i.e. DS3 in Fig. 7(b)) without waiting at the north channel.

The EDAR approach also imposes the restriction that data packets coming from any given direction cannot return on the same direction. The livelock avoidance includes three scenarios – (a). For a fault-free NoC system, EDAR is livelock-free as the packets will eventually reach the destination node although it will experience a longer path delay; (b). For convex faulty regions, the EDAR routes packets along the edge, then turn direction at the region corner and finally arrive at the destination node; (c). For a NoC system with concave faulty regions or other serious scenarios, a re-routing constraint mechanism [33] can be employed. It constrains the number of re-routing performed and discards packets if re-routing exceeds a threshold number. Discarding packets can relax the communication links, prevent traffic congestion, and maintain system traffic load balance. It avoids livelock, however, in the meantime it induces an issue of Quality of Service, as a packet can be dropped. An Automated Repeat reQuest (ARQ) mechanism can be employed to re-transmit the packet but with a different port, since the packet sent through the previous port is lost. Alternatively, a node de-activation mechanism [34] can be used to avoid livelock for the concave faulty regions, which converts the concave fault region to be convex. For EDAR, a minimal path is always chosen when the path is fault-free and not congested. If congestion occurs or the desired direction is faulty, the EDAR selects alternative directions to forward the packets without waiting on the desired direction becoming available. This leads to a non-minimal path length, however, it guarantees that packets can be successfully delivered.

4. Methodology and experimental results

This section outlines the methodology used in performing experiments and presents results on the performance of the EDAR under non-faulty conditions.

4.1. Performance analysis metrics and experimental platform

The standard evaluation metrics from the approach of [9] are employed in this paper. For example, packet injection rate (*PIR*) refers to the rate at which packets are injected into the NoC network. For any given single node router in the NoC, the normalised number of sent packets per clock cycle is equal to *PIR* and has the range $0 < PIR \leqslant 1$. If $PIR = 0.2$, this means the node sent 0.2 packets per clock cycle or 2 packets every 10 clock cycles. The performance metrics of throughput, T, and average delay, D, are used and defined in (4) and (5) [9].

$$T = \frac{R_{flits}}{N_{nodes} \times N_{clk}} \tag{4}$$

In (4), R_{flits} is the total number of received flits, N_{nodes} is the total number of nodes and N_{clk} is the number of clocks cycles from the first generated flit to the last received flit. Therefore, the throughput is measured as a fraction of the maximum load that the network is capable of physically handling over a given path. Delay is defined as the number of clock cycles that elapses between the occurrence of a header flit injection into the network at the source node and the occurrence of a tail flit reception at the destination node. Eq. (5) defines the average delay, D, which is the average value for the total number of messages where K is the total number of messages reaching their destination nodes and D_i is the delay for the node i.

$$D = \frac{1}{K}\sum_{i=1}^{K} D_i \tag{5}$$

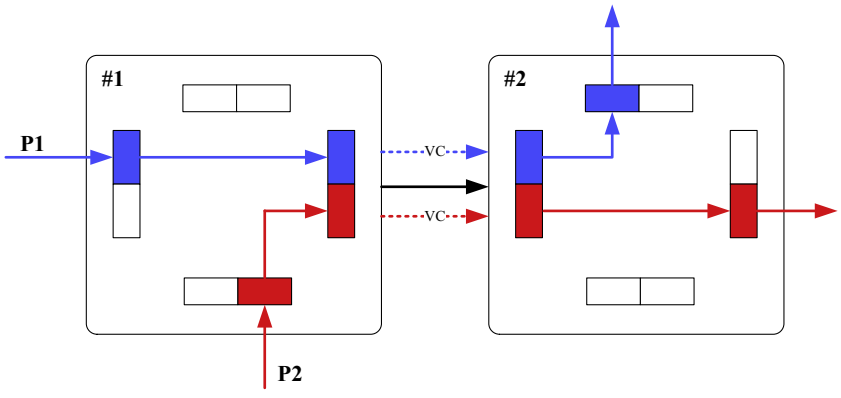


Fig. 6. An example of VCs in the NoC router.

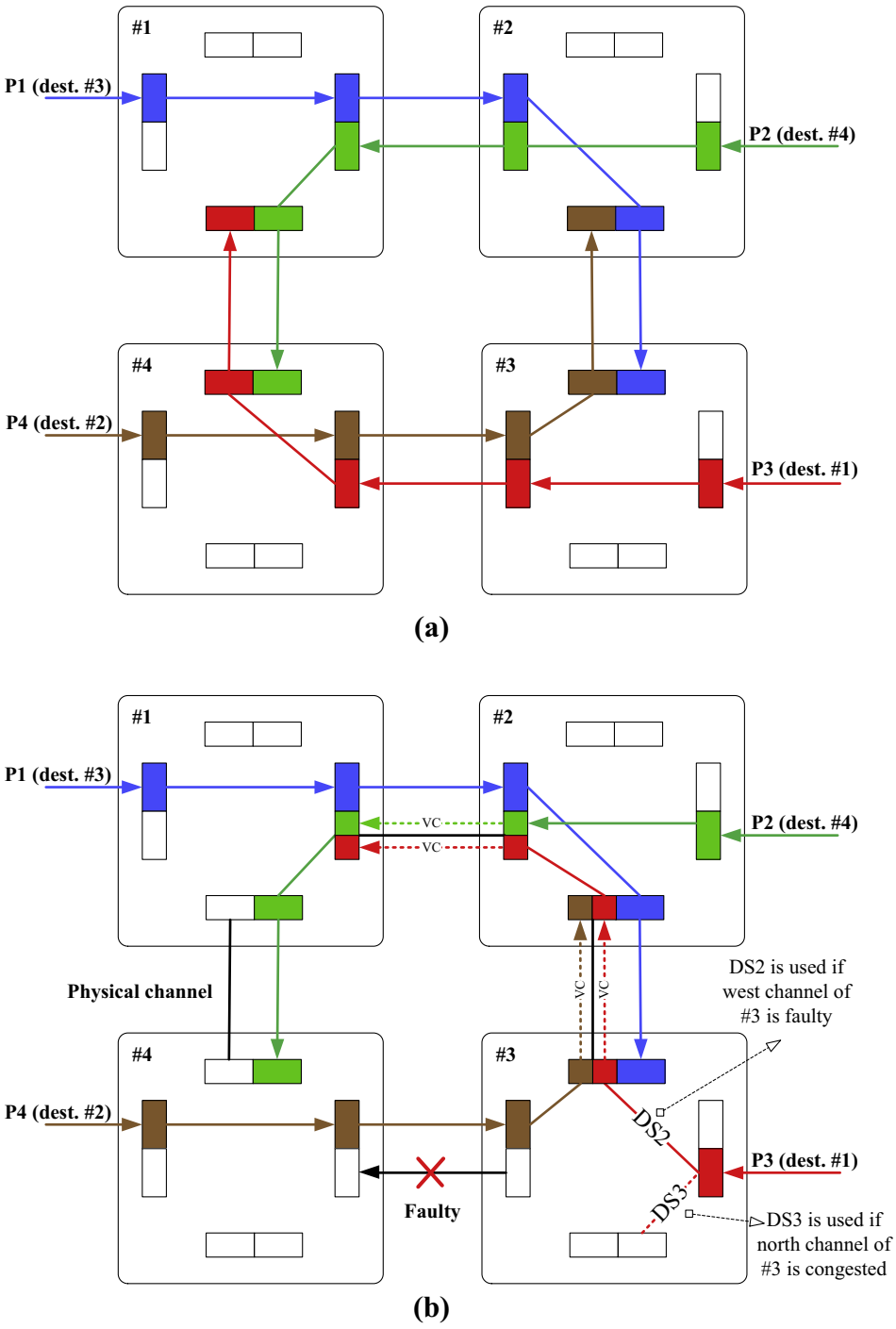


Fig. 7. A 4 × 4 NoC system: (a) fault-free system and (b) one link is faulty.

The evaluation environment is presented in Table 2. The Noxim simulator [35] was extended to evaluate the EDAR routing algorithm's performance. The NoC application mapping strategies of NoCTweak [36] and NoCmap [37] were used to generate the real and non-synthetic traffic patterns. The evaluation platform of the EDAR NoC algorithm in this paper was based on a 2D-mesh system. To guarantee the accuracy of results, the simulation at each *PIR* point has been repeated several times (five times in this approach). The warm up and execution times are 1000 and 20,000 clock cycles, respectively. The EDAR routing algorithm is evaluated under various traffic patterns including (1) uniform, (2) transpose, (3) shuffle and (4) a real NoC application traffic load Multi-Media System (MMS), which are common traffic patterns used in evaluating routing performance [11,9,22,38]. A generic MMS includes an H.263 video encoder, H.263 video decoder, mp3 audio encoder, and an mp3 audio decoder [37]. The MMS application is partitioned into 40 distinct tasks and then these tasks were assigned and scheduled in the NoC system. MMS is a typical traffic load of real NoC applications and has been used in many approaches as a testbench framework [9,39,40]. The performance evaluation and comparison follows two steps. The first step uses the extended Noxim simulator to evaluate the EDAR congestion-aware capability. Noxim integrates the routing algorithms of DyAD [6], Odd–even [41], XY, Negative-first. The standard simulator allows a

Table 2
Evaluation environment.

Simulator	Extended Noxim simulator
Topology	2D mesh
Routing	XY, DyAD, Odd–even, Negative-first,FoN, Cost, FTDR, FTDR-H, LAFT, HLAFT, EDAR
Traffic load	Uniform, transpose, shuffle, MMS
Simulation warm up [cycles]	1000
Execution time [cycles]	20,000
Simulation repeat time	5
Fault rates	0%, 5%, 10%, 15%, 20%

performance comparison between the proposed EDAR and integrated routing algorithms under the same traffic conditions. This evaluation method, i.e. using the standard platform to evaluate the proposed algorithm and compare with integrated routing algorithms, has been widely used in other studies, e.g. [9,11,26,42]. However, the integrated routing algorithms are not fault-tolerant. Therefore, a second step was undertaken, i.e. several state-of-the-art approaches including FoN, Cost, FTDR, FTDR-H [43], LAFT, HLAFT [26] were chosen as benchmarks. As the router architectures of EDAR and these approaches are different, the results generated by the first step are normalised (e.g. throughput degradation value) in order to give a comparison. The experimental results across various *PIR* were obtained for each algorithm using different traffic patterns.

4.2. Experimental results

This section presents the results from experiments on the performance of the EDAR algorithm compared against the listed benchmark routing algorithms under varied traffic loads. The results of the average delay, *D*, and throughput, *T*, under uniform traffic conditions for EDAR and all benchmark algorithms are shown in Fig. 8. This type of traffic is used for simulation in many approaches. It can be seen that EDAR and XY routing algorithms achieves a lower average delay and a higher throughput compared to other algorithms. The result from the XY routing algorithm is consistent with the results of other approaches [9,41,44] and is due to the fact that XY embodies global long-term information about the uniform traffic pattern [44]. By routing packets first along the *X*-axis and then *Y*-axis, it spreads the uniform traffic as evenly as possible across the channels in the long term. The adaptive routing algorithms on the other hand, select channels for transmission based on local short-term information. This type of decision making benefits only the packets in the immediate future, which can, in the longer term interfere with other network traffic/packets. The result is increased contention and decreased performance at higher *PIR* rates. However, EDAR still yields a low average delay and a high throughput in this traffic pattern, in particular it outperforms the adaptive benchmark DyAD.

Fig. 9 presents the results under transpose traffic load. At the point of *PIR* = 0.017, the average delay for DyAD, XY and Negative-first increase sharply and at the same point, their throughputs begin to decrease. Odd–Even is the only algorithm which achieves a marginally higher performance under this traffic pattern. The main reason is that the turn model of Odd–Even can spread the transpose traffic as evenly as possible across the channels, which is consistent with the result in the approach of [41]. However, the EDAR also has a very similar performance to Odd–Even under this traffic pattern as it spreads the traffic to the less congested channels adaptively and achieves a good traffic balance.

Fig. 10 presents the average delay and throughput for the traffic shuffle. For DyAD, Westfirst and XY routing, their average delays begin to increase at *PIR* values of 0.017 and 0.025, respectively. At the same *PIR* values their throughputs begins to decrease as well. However, EDAR and Odd–even maintain a low average delay and high throughput with Odd–even only providing a marginal gain.

However, it should be noted that the real performance benefit of EDAR is achieved under more difficult circumstances when both increased traffic loads and faulty conditions are present. The next section present results on EDAR under such conditions.

5. Performance results under fault conditions

A set of experiments were carried out to evaluate the performance of EDAR under traffic patterns while faulty channel links were present. The effects of faults on the NoC interconnection are different and depend on the fault type. In this approach, the stuck-at fault [45] is considered as it is the most prevalent fault model for NoCs and was used in previous approaches [46–49]. A fault rate is used to present the percentage of faulty links in a NoC system. A NoC system with different fault rates is employed as the test bench framework in evaluating EDAR performance. Similar to the approaches of [43,38], ten fault patterns are chosen for each fault rate and the average value is used as the result.

5.1. EDAR performance under fault conditions

Stuck-at faults mostly occur at the logic level, which can be divided into stuck-open and stuck-on faults resulting in non-conductive and conductive logic states respectively. In this approach, if the packets pass through a channel having stuck-at

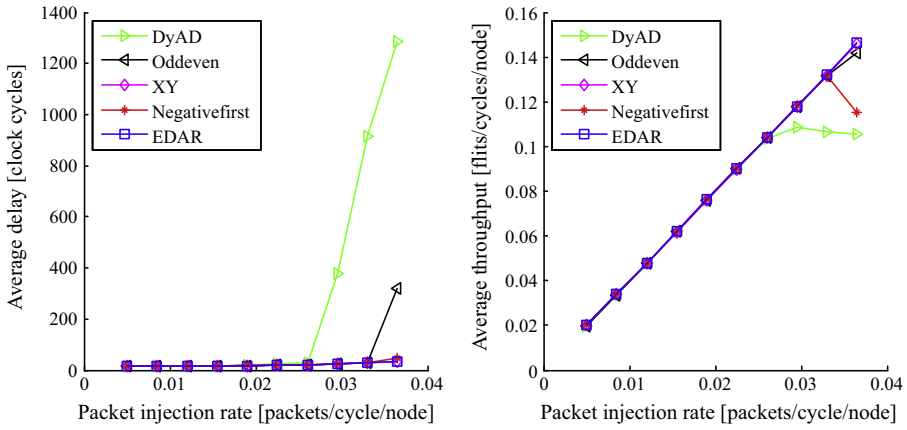


Fig. 8. Performance under uniform traffic load.

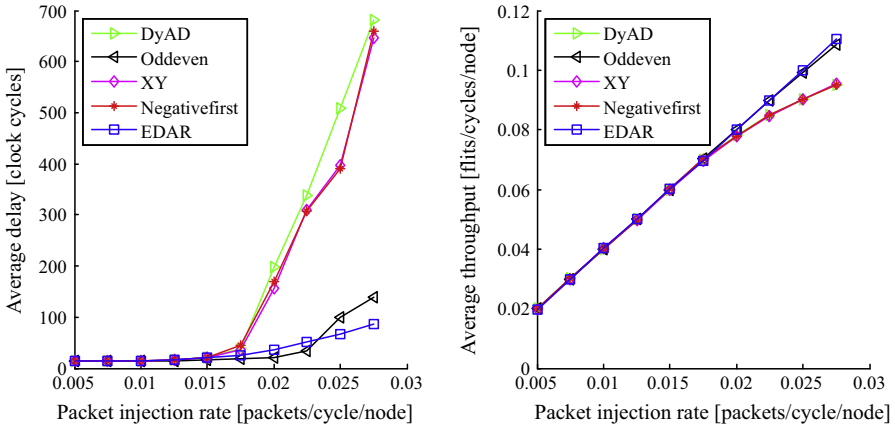


Fig. 9. Performance under transpose traffic load.

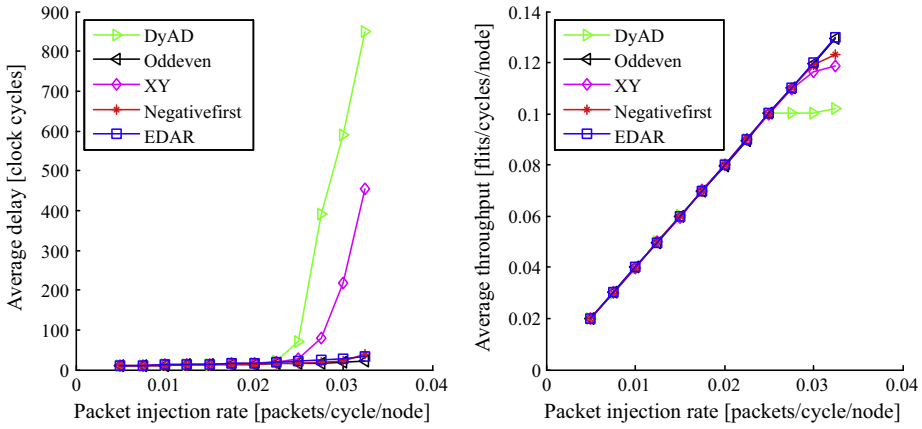


Fig. 10. Performance under shuffle traffic load.

faults, the RX router cannot recognise the packets as the channel has been stuck. Under this assumption, the 8 × 8 NoC is simulated with 5%, 10%, 15% and 20% of faulty links. Based on these simulations the EDAR throughput performance can be evaluated and compared.

Fig. 11 presents the throughput of the NoC system for the four benchmark routing algorithms and EDAR, under the four different traffic patterns. The *PIR* rate below the saturation point is chosen for the routing algorithms of uniform, transpose and shuffle. The benefit of using this *PIR* rate as the baseline value is that the throughputs of all the routing algorithms are the same for the fault-free NoC system (i.e. not in saturation); this allows a fair evaluation of system performance under various faulty link percentages. When the system has faulty links, the throughput performance of the four benchmark algorithms are degraded as the faulty links do not transmit packets any longer. The throughput value of all the routing algorithms for different fault rates and various traffic patterns are presented in detailed in Fig. 11. The throughput degradations (shown by percentage) for all these scenarios are outlined in Table 3. It can be seen that for the system with 5–20% faulty links, all the routing algorithms have different levels of performance degradation. Across all four traffic patterns, DyAD experienced between 26.19–58.07% degradation for 5–20% fault rates; Odd–Even 25.72–61.48% degradation; Negative-first and XY had degradation of 31.01–68.19% and 31.71–68.3%, respectively. However, EDAR had the lowest degradation with between 0.07% and 20.3%. When the fault rate is 5% and 10%, the throughput degradation of EDAR is very low, 0.04% and 4.3%, respectively. When the fault rate increases, the throughput of EDAR has a small degradation, i.e. 13.6% for a fault rate of 15%, and 20.3% degradation for a 20% fault rate. These decreases in throughput performances are significantly lower than the benchmarks, for example, in comparison EDAR achieves between 26% and 48% improvement on throughput. Therefore, the results in Fig. 11 and Table 3 illustrate that the EDAR routing algorithm maintains system performance under a low fault rate and only has a marginal performance degradation when the system has a significant fault rate.

In addition to the key metric of throughput, the communication latency is also used as a metric to evaluate the routing algorithm performance. Fig. 12 presents the average delay and the number of received flits at different fault rates under various traffic patterns. When the fault rate increases, the average delay of the EDAR increases, i.e. it takes a longer time for a packet to arrive at the destination node. However, for the benchmark routing algorithms, the average delays remains either relatively unchanged (for traffic patterns of uniform and shuffle) or decreased marginally (for the rest traffic patterns). This is due to the fact that the average delay only takes account of the flits that actually arrive at the destination node successfully. For the benchmark routing algorithms, when the fault rate increases, they cannot choose fault-free paths to forward the flits, therefore many flits are lost and the numbers of received flits decreases significantly. These lost flits are not counted in the average delay calculation; therefore the average delays are lower at a high fault rate. However for EDAR, the average latencies increase but the numbers of received flits remains at effectively the same level (at fault rates of 5–10%) or have a marginal decrease (at fault rate of 15–20%), i.e. EDAR requires a longer

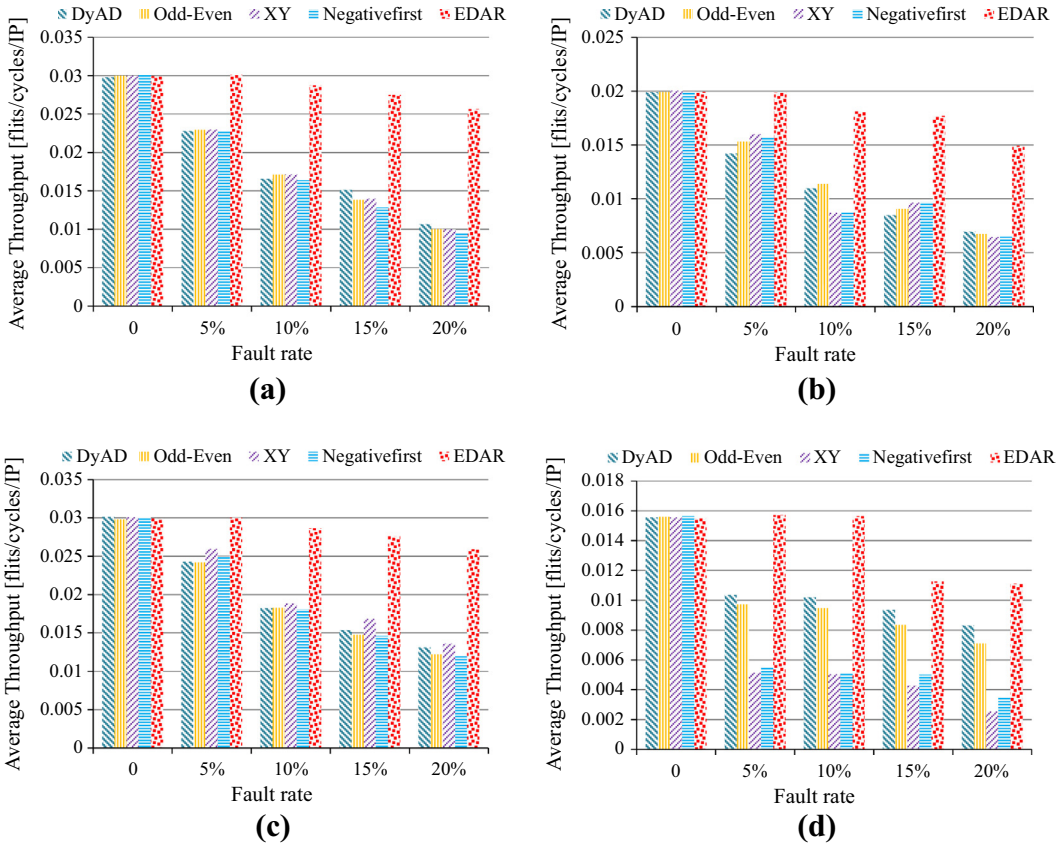


Fig. 11. Throughput at different fault rates under various traffic patterns: (a) uniform, (b) transpose, (c) shuffle and (d) MMS.

Table 3
The throughput degradation under various traffic patterns.

Routing algorithm	Fault rate (%)	Throughput degradation (%)				
		Uniform	Transpose	Shuffle	MMS	Average
DyAD	5	23.51	28.55	19.41	33.28	26.19
	10	44.37	45.53	39.31	34.43	40.91
	15	49.18	57.43	48.96	39.72	48.82
	20	64.23	65.09	56.39	46.57	58.07
Odd–Even	5	23.48	23.11	18.76	37.53	25.72
	10	42.81	43.69	38.62	39.16	41.07
	15	53.73	54.46	50.42	46.43	51.26
	20	66.40	66.19	58.95	54.38	61.48
Negative-first	5	23.42	20.11	13.71	66.81	31.01
	10	42.91	51.57	37.38	67.55	49.85
	15	53.36	51.77	44.01	72.45	55.40
	20	66.61	67.73	54.75	83.66	68.19
XY	5	24.37	20.97	16.90	64.60	31.71
	10	45.51	51.29	39.89	66.99	50.92
	15	57.34	51.36	51.36	67.87	56.98
	20	68.16	67.42	60.17	77.47	68.30
EDAR	5	√	0.30	√	√	0.07
	10	4.07	8.78	4.36	0.00	4.30
	15	8.12	11.13	7.82	27.32	13.60
	20	14.31	25.11	13.51	28.27	20.30

'√' the throughput is not degraded under the respective fault rate (%).

time to forward the packets to the destination nodes, but they do arrive.

5.2. Performance comparison with fault-tolerant routing algorithms

An additional six state of the art fault-tolerant routing algorithms, namely FoN, Cost, FTDR, FTDR-H, LAFT and HLAFT [43], are also chosen as benchmarks in evaluating the EDAR performance. The throughput degradation is chosen as a benchmark metric for all of the routing algorithms to allow a fair comparison of system performance. No single testbench platform is standard however the throughput degradation can accurately reflect the fault-tolerant capability of the routing algorithms. Traffic loads of uniform, transpose and shuffle, and fault rates of 5%, 10% and 20% were chosen as the testbench baseline, as was done in [43]. In addition performance results in [26] were also presented under these baseline conditions. Table 4 presents the throughput degradation results. It can be seen that FoN, Cost, FTDR and FTDR-H have 40–70% throughput degradation under fault rates between 10% and 20%. However, EDAR has a significantly lower throughput degradation of 25% at fault rates between 10–20% compared to FoN, Cost, FTDR and FTDR-H. From the table it can also be seen that the fault tolerance of HLAFT is better than LAFT [26], where HLAFT has 0.9–54% throughput degradation under fault rate 5–20%. However, EDAR has a lower throughput degradation of between 0.3% and 25%; i.e. outperforming HLAFT. Therefore, the results show that the EDAR routing algorithm outperforms these benchmarks. This demonstrates EDAR's ability in tolerating faults in the NoC system and how it can minimise degradation of performance when faults occur. Note that the comparison is a likely to likely comparison as the router architectures are different.

6. Hardware performance

This section presents the methodology for implementing the EDAR algorithm in hardware and also presents an evaluation on area overheads. The additional circuit area required for the EDAR is highlighted and the performance benchmarked against other state of the art approaches. In order to evaluate the hardware

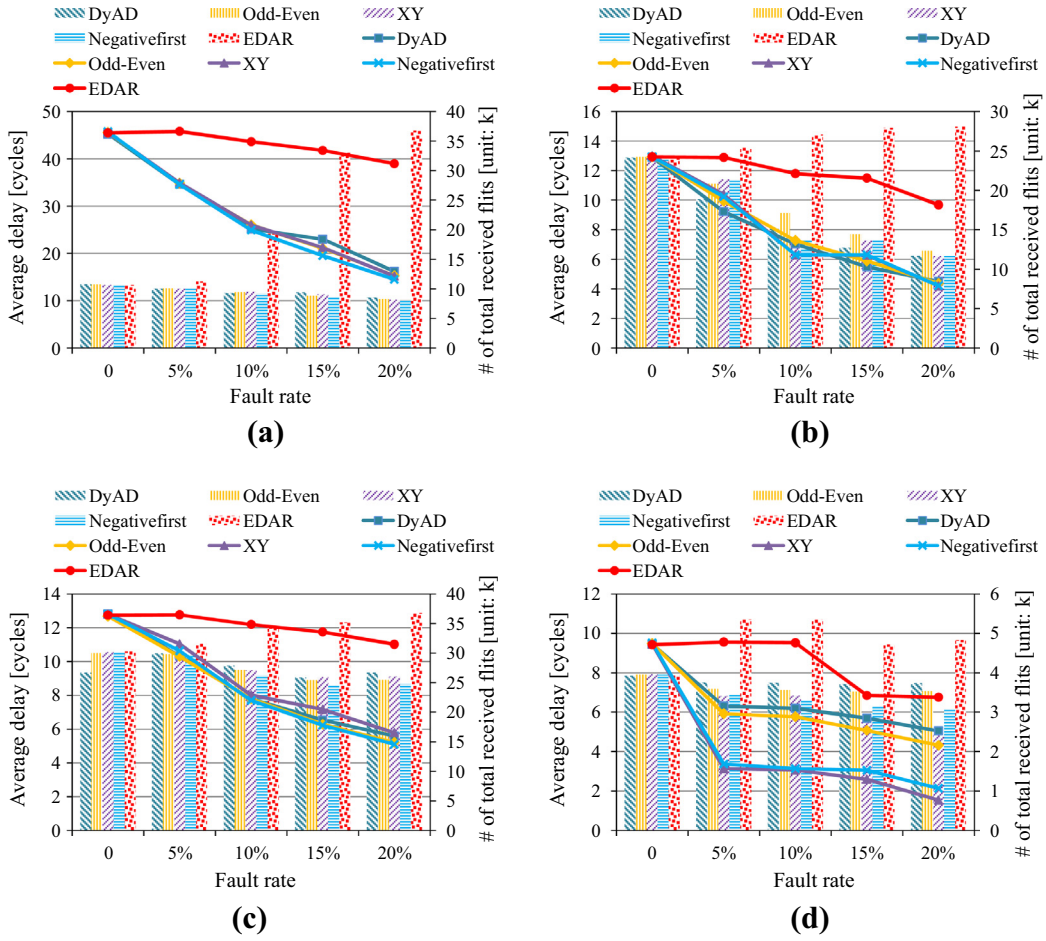


Fig. 12. Average delay (bars) and the number of received flits (lines) under various traffic patterns: (a) uniform, (b) transpose, (c) shuffle and (d) MMS.

Table 4
The throughput degradation comparison between FoN, Cost, FTDR, FTDR-H, LAFT, HLAFT and EDAR routing algorithms.

Traffic load	Throughput degradation (%)								
	Uniform			Transpose			Shuffle		
Fault rate (%)	5	10	20	5	10	20	5	10	20
FoN [43]	N/A	55.88	64.71	N/A	45.83	59.72	N/A	51.22	65.85
Cost [43]	N/A	61.11	72.22	N/A	54.76	71.43	N/A	58.14	69.77
FTDR [43]	N/A	52.94	61.76	N/A	44.44	52.78	N/A	48.78	63.41
FTDR-H [43]	N/A	48.39	58.06	N/A	28.57	46.43	N/A	46.15	61.54
LAFT [26]	11.24	42.7	98.88	7.21	27.3	99.1	N/A	N/A	N/A
HLAFT [26]	8.99	33.71	43.82	0.9	13.51	54.05	N/A	N/A	N/A
EDAR	√	4.07	14.31	0.3	8.78	25.11	√	4.36	13.51

'√' the throughput is not degraded under the respective fault rate (%).
'N/A' the results are not presented in the respected authors' paper.

performance of the EDAR, the EMBRACE NoC router has been extended to include the EDAR in each NoC direction and has been implemented using VHDL. The VHDL model of the EDAR is parameterised in terms of the widths of the channel and the packet header information, and can be easily instantiated to fit different requirements for other specific applications other than the EMBRACE router used in this evaluation. The modified EMBRACE NoC router is based on our previous work by the authors [10,27,28] and it is characterised by its area overhead. The hardware implementation parameters for the router include the following: (1) 36 bit data packets; (2) 100 MHz system frequency and (3) one MM per channel to test for faults. The area overhead of the proposed EDAR has been obtained using the Synopsys Design Compiler tool based on SAED 90 nm CMOS technology.

Fig. 13(a) shows the schematic of a single EMBRACE router implementing the adaptive routing scheme using the EDAR algorithm. The top level scheme of the router consists of an input buffer (i.e. FIFO), Monitor Module (MM), EDAR and Adaptive Arbitration Policy (AAP) components. The EDAR module is used when a routing decision has to be made, that is, when a header flit reaches the input buffer. The packet header, together with the traffic information signals coming from the neighbours, is used to make a routing decision. The output of the EDAR, that is, the optimal selected *N*, *E*, *S* or *W* direction, is then used by the AAP in order to set up the connection between the input traffic flows to the proper output directions. When a router attempts to send or forward a packet, EDAR checks the traffic status of all the channels and decides which port is the best way to forward the packet. This routing decision is made

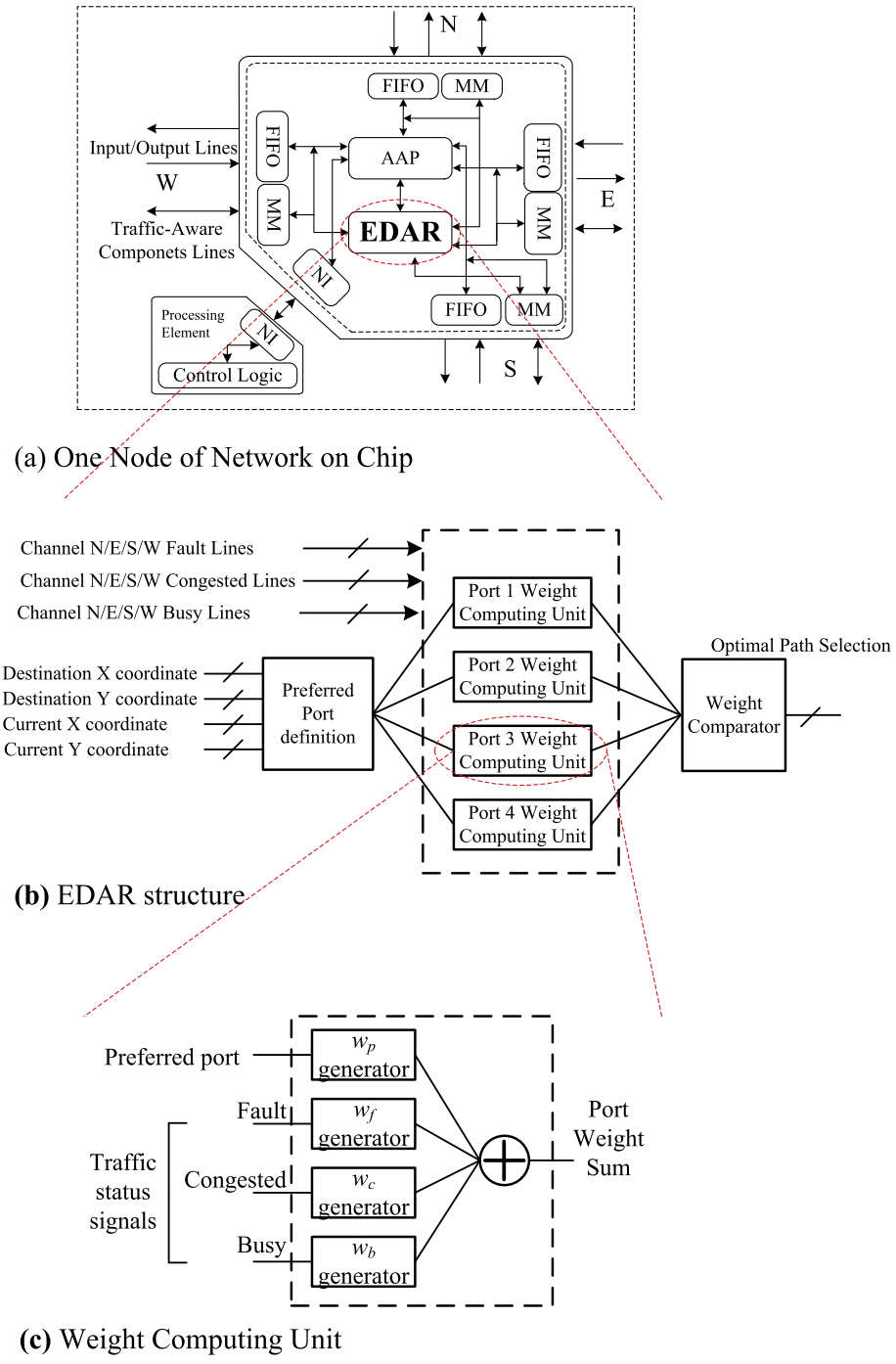


Fig. 13. Adaptive routing scheme structure: (a) one node of the NoC system; (b) the adaptive routing scheme structure and (c) the weight computing unit structure.

by choosing the lowest weight of all the ports and follows 3 steps: (1) defines the port priority definition, (2) computes the port weight and (3) compares all the weights to choose the port with the lowest weight. An overview of EDAR hardware structure is depicted in Fig. 13(b). Each EDAR module consists of three components: preferred port definition module, weight computing unit and weight comparator. The preferred port definition module defines the port priority weight w_p according to the relative position of the current and destination nodes. The preferred port weight value for the N/E/S/W direction is a default weight determined by the current (x_c, y_c) and destination (x_d, y_d) coordinates. The output value from a preferred port definition module is combined with the traffic status information (B/C/F), and forwarded to the weight computing unit to calculate the port weights; calculation based on (2). All port weights are input to weight comparator and the port with the lowest weight is the selected output port for transmission of the packet. Therefore, this port selection can provide the optimal output port selection result based on the destination node position and the current traffic status (i.e. under faulty or traffic-load conditions). Fig. 13(c) illustrates the structure of the weight computing unit for one direction; it is identical for all four directions. The inputs are preferred port level and traffic status signals (i.e. B/C/F). Four weight generators generate corresponding $w_p/w_f/w_b/w_c$ weights based on the input

Table 5
Router hardware overhead and power consumption summary.

The approach	Congestion aware	Fault detection	Fault-tolerant	Device technology	Router (mm^2)	Power Consumption (mW)
[10]	√	×	×	90 nm CMOS	0.056	1.716
[27]	√	√	×	SAED 90 nm	0.182	2.175
EDAR	√	√	√	SAED 90 nm	0.241	2.291

signals. All the weights are accumulated to provide the port weight sum. The weight generators are implemented efficiently using multiplexers; therefore low area overhead is achieved.

The implementation approach followed the standard ASIC cell design flow, synthesis and verification based on a SAED 90 nm CMOS technology. The implementation was evaluated for efficiency with respect to hardware area and power consumption of the router. The authors use their previous EMBRACE NoC router [10,27] to incorporate EDAR, which has the four-port connectivity. Table 5 provides a comparison of the EDAR algorithm against previous work and illustrates the router capabilities, hardware area and power consumption. The approach in [10] provides congestion-aware adaptive routing however it does not provide a fault-tolerant capability, therefore the area overhead and power consumption is relative low (0.056 mm^2 and 1.716 mW). Based on [10], another router was developed which was equipped with an online fault detection mechanism [27]. Its routing algorithm is not fault-tolerant and its area overhead and power consumption are 0.182 mm^2 and 2.175 mW, respectively. The proposed EDAR router has the capabilities of congestion-aware, fault detection and fault-tolerant. The area overhead and power consumption is 0.241 mm^2 and 2.291 mW. From Table 5, it can be seen that EDAR has the largest area overhead and power consumption. Compared to the approach of [27], the increment is not significant (i.e. ∼5%). In addition, the EDAR routing requires extra control signals (i.e. B/C/F lines) to aid in gathering traffic and fault information. In this approach, each NoC channel is 36 bits wide; given this width the extra wire overhead is 3/36 = 8.33%. Generally the extra wire cost of ∼10% can be accepted, such as in the approach of [9]. Therefore, the additional wires are not a hardware constraining factor.

7. Conclusion

In this paper an EDAR routing algorithm is proposed to improve the NoC throughput performance for complex traffic conditions. The aim of EDAR is to exploit the situations of indecisions that can occur in the adaptive routing algorithm. This approach employs the traffic status information from a monitor module to aid in making routing decisions. A weighted path selection strategy was proposed for the NoC to identify a minimal latency output port direction. The performance evaluation results showed that in the majority of test traffic patterns, EDAR achieves an improvement in average delay and throughput; in particular, it does so under the traffic conditions where faulty links exist. The hardware overhead is also shown to be very low enabling system scalability to be maintained.

Future work will explore issues such as how to re-allocate the underlying resources to reconfigure the system after identifying temporary faults, and how to provide system functionality in a reduced capacity if the underlying resources are no longer available from physical, permanent faults. In summary, future work will explore how to make an optimal repair decision and aims to build on the presented fault detection and EDAR routing strategy.

Acknowledgements

Junxiu Liu is supported by the University of Ulster's Vice-Chancellor's Research Scholarship (VCRS) and the Vice-Chancellor's Student Fund.

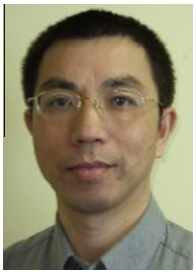
References

[1] A. Agarwal, B. Raton, C. Iskander, H. Multisystems, R. Shankar, Survey of Network on Chip (NoC) architectures & contributions, J. Eng. Comput. Arch. 3 (1) (2009) 1–15.
[2] E. Salminen, A. Kulmala, D.H. Timo, Survey of Network-on-Chip Proposals, White Paper OCP-IP, 2008, pp. 1–13.
[3] R.A. Shafik, J. Mathew, D.K. Pradhan, Introduction to energy-efficient fault-tolerant systems, in: Energy-Efficient Fault-Tolerant Systems, 2014, pp. 1–10.
[4] ITRS <http://www.itrs.net/>.
[5] K. Aisopos, A. DeOrio, L.-S. Peh, V. Bertacco, ARIADNE: agnostic reconfiguration in a disconnected network environment, in: International Conference on Parallel Architectures and Compilation Techniques, 2011, pp. 298–309.
[6] J. Hu, R. Marculescu, DyAD – smart routing for networks-on-chip, in: 41st Design Automation Conference, 2004, pp. 260–263.
[7] M. Li, Q. Zeng, W. Jone, DyXY – a proximity congestion-aware deadlock-free dynamic routing method for network on chip, in: 43rd ACM/IEEE Design Automation Conference, 2006, pp. 849–852.
[8] B. Niazmand, M. Reshadi, A. Reza, PathAware: a contention-aware selection function for application-specific Network-on-Chips, in: NORCHIP, 2012, pp. 1–6.
[9] G. Ascia, V. Catania, M. Palesi, D. Patti, Implementation and analysis of a new selection strategy for adaptive routing in networks-on-chip, IEEE Trans. Comput. 57 (6) (2008) 809–820.
[10] S. Carrillo, J. Harkin, L. McDaid, S. Pande, S. Cawley, B. McGinley, F. Morgan, Advancing interconnect density for spiking neural network hardware implementations using traffic-aware adaptive Network-on-Chip routers, Neural Networks 33 (9) (2012) 42–57.
[11] P. Lotfi-Kamran, A.M. Rahmani, M. Daneshtalab, A. Afzali-Kusha, Z. Navabi, EDXY – a low cost congestion-aware routing algorithm for Network-on-Chips, J. Syst. Arch. 56 (7) (2010) 256–264.
[12] X. Chang, M. Ebrahimi, M. Daneshtalab, T. Westerlund, J. Plosila, PARS – an efficient congestion-aware routing method for networks-on-chip, in: 16th CSI International Symposium on Computer Architecture and Digital Systems, 2012, pp. 166–171.
[13] Z. Lu, M. Zhong, A. Jantsch, Evaluation of on-chip networks using deflection routing, in: 16th ACM Great Lakes Symposium on VLSI, 2006, pp. 296–301.
[14] N. Jiang, J. Kim, W.J. Dally, Indirect adaptive routing on large scale interconnection networks, in: 36th Annual International Symposium on Computer Architecture, 2009, pp. 220–231.
[15] R. Manevich, I. Cidon, A. Kolodny, I. Walter, Centralized adaptive routing for NoCs, IEEE Comput. Arch. Lett. 9 (2) (2010) 57–60.
[16] R. Manevich, I. Cidon, A. Kolodny, I. Walter, S. Wimer, A cost effective centralized adaptive routing for networks-on-chip, in: 14th Euromicro Conference on Digital System Design, vol. 9(2), 2011, pp. 39–46.
[17] P. Gratz, B. Grot, S.W. Keckler, Regional congestion awareness for load balance in networks-on-chip, in: IEEE 14th International Symposium on High Performance Computer Architecture, 2008, pp. 203–214.
[18] M. Ramakrishna, P.V. Gratz, A. Sprintson, GCA: global congestion awareness for load balance in networks-on-chip, in Seventh IEEE/ACM International Symposium on Networks-on-Chip (NoCS), 2013, pp. 1–8.
[19] Z. Zhang, A. Greiner, 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.
[20] J. Wang, F. Fu, T. Zhang, Y. Chen, A small-granularity solution on fault-tolerant in 2D-mesh Network-on-Chip, in: 10th IEEE Conference on Solid-State and Integrated Circuit Technology, 2010, pp. 382–384.
[21] R. Parikh, V. Bertacco, uDIREC: unified diagnosis and reconfiguration for frugal bypass of NoC faults, in: 46th Annual IEEE/ACM International Symposium on Microarchitecture, 2013, pp. 148–159.
[22] I. Pratomo, S. Pillement, Gradient – an adaptive fault-tolerant routing algorithm for 2D mesh Network-on-Chips, in: Design and Architectures for Signal and Image Processing (DASIP), 2012, pp. 1–8.
[23] Y. Wang, M. Zhang, C. Xiao, A fault tolerant adaptive routing algorithm in 2D mesh network on chip, in: IEEE 3rd International Conference on Communication Software and Networks, 2011, pp. 140–144.
[24] A. Vitkovskiy, V. Soteriou, C. Nicopoulos, Dynamic fault-tolerant routing algorithm for networks-on-chip based on localised detouring paths, IET Comput. Dig. Tech. 7 (2) (2013) 93–103.
[25] C. Feng, Z. Lu, A. Jantsch, J. Li, M. Zhang, A reconfigurable fault-tolerant deflection routing algorithm based on reinforcement learning for Network-on-Chip, in: Third International Workshop on Network on Chip Architectures, 2010, pp. 11–16.
[26] A. Ben Ahmed, A. Ben Abdallah, Graceful deadlock-free fault-tolerant routing algorithm for 3D Network-on-Chip architectures, J. Parall. Distrib. Comput. 74 (4) (2014) 2229–2240.

[27] J. Liu, J. Harkin, Y. Li, L. Maguire, Online traffic-aware fault detection for networks-on-chip, J. Parall. Distrib. Comput. 74 (1) (2014) 1984–1993.
[28] S. Carrillo, J. Harkin, L.J. McDaid, F. Morgan, S. Pande, S. Cawley, B. McGinley, Scalable hierarchical Network-on-Chip architecture for spiking neural network hardware implementations, IEEE Trans. Parall. Distrib. Syst. 24 (12) (2013) 2451–2461.
[29] S. Pande, F. Morgan, G. Smit, T. Bruintjes, J. Rutgers, B. McGinley, S. Cawley, J. Harkin, L. McDaid, Fixed latency on-chip interconnect for hardware spiking neural network architectures, Parall. Comput. 39 (9) (2013) 357–371.
[30] N.E. Jerger, L.-S. Peh, On-Chip Networks, vol. 4(1), Morgan & Claypool, 2009. pp. 1–141.
[31] W.J. Dally, H. Aoki, Deadlock-free adaptive routing in multicomputer networks using virtual channels, IEEE Trans. Parall. Distrib. Syst. 4 (4) (1993) 466–475.
[32] J. Kim, D. Park, T. Theocharides, N. Vijaykrishnan, C.R. Das, A low latency router supporting adaptivity for on-chip interconnects, in: 42nd annual Design Automation Conference, 2005, pp. 559–564.
[33] A. Alhussien, C. Wang, N. Bagherzadeh, Design and evaluation of a high throughput robust router for Network-on-Chip, IET Comput. Dig. Tech. 6 (3) (2012) 173–179.
[34] A.A. Chien, J.H. Kim, Planar-adaptive routing: low-cost adaptive networks for multiprocessors, J. ACM 42 (1) (1995) 91–123.
[35] M. Palesi, D. Patti, F. Fazzino, Noxim: Network-on-Chip Simulator, 2010 <http://www.noxim.org>.
[36] A.T. Tran, B.M. Baas, NoCTweak: a Highly Parameterizable Simulator for Early Exploration of Performance and Energy of Networks On-Chip, 2012.
[37] J. Hu, R. Marculescu, Energy- and performance-aware mapping for regular NoC architectures, IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst. 24 (4) (2005) 551–562.
[38] C. Feng, Z. Lu, A. Jantsch, L. Jinwen, M. Zhang, FON: fault-on-neighbor aware routing algorithm for networks-on-chip, in: 23rd IEEE International SoC Conference, 2010, pp. 441–446.
[39] Y.-H. Kuo, P.-A. Tsai, H.-P. Ho, E.-J. Chang, H.-K. Hsin, A.-Y. (Andy) Wu, Path-diversity-aware adaptive routing in Network-on-Chip systems, in: IEEE 6th International Symposium on Embedded Multicore SoCs, 2012, pp. 175–182.
[40] P.-A. Tsai, Y.-H. Kuo, E.-J. Chang, H.-K. Hsin, A.-Y. Wu, Hybrid path-diversity-aware adaptive routing with latency prediction model in Network-on-Chip systems, in: International Symposium on VLSI Design, Automation, and Test (VLSI-DAT), 2013, pp. 1–4.
[41] G. Chiu, The odd–even turn model for adaptive routing, IEEE Trans. Parall. Distrib. Syst. 11 (7) (2000) 729–738.
[42] T. Mak, P.Y.K. Cheung, K. Lam, W. Luk, Adaptive routing in Network-on-Chips using a dynamic-programming network, IEEE Trans. Ind. Electron. 58 (8) (2011) 3701–3716.
[43] C. Feng, Z. Lu, A. Jantsch, M. Zhang, Z. Xing, Addressing transient and permanent faults in NoC with efficient fault-tolerant deflection router, IEEE Trans. VLSI Syst. 21 (6) (2013) 1053–1066.
[44] C.J. Glass, L.M. Ni, The turn model for adaptive routing, in: 19th Annual International Symposium on Computer Architecture, 1992, pp. 278–287.
[45] C. Michael, D. Sujit, B. Xiaoliang, 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.
[46] C. Grecu, A. Ivanov, Testing Network-on-Chip communication fabrics, IEEE Trans. CAD Integr. Circ. Syst. 26 (12) (2007) 2201–2214.
[47] M. Botelho, F.L. Kastensmidt, M. Lubaszewski, E. Cota, 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.
[48] C. Concatto, P. Almeida, F. Kastensmidt, E. Cota, M. Lubaszewski, 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.
[49] M.R. Kakoee, V. Bertacco, 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.

Junxiu Liu received a BEng in Electronic Information Engineering from Hunan Institute of Science and Technology in 2007 and an MPhil in Signal and Information Processing from Guangdong University of Technology in 2010. He worked at Guangxi Normal University and then received a Vice-Chancellor's Research Scholarship (VCRS) to start attending the University of Ulster, UK to pursue his Ph.D. in Computer Science from 2012. He is developing his research project in the School of Computing and Intelligent Systems to work on self-correction strategy for Networks-on-Chip interconnect.

Jim Harkin received the BTech degree in Electronic Engineering, the M.Sc. in electronics and signal processing, and the Ph.D. degree from the University of Ulster, UK, in 1996, 1997, and 2001, respectively. He is a Senior Lecturer with the School of Computing and Intelligent Systems, University of Ulster. He is a member of the UK Neuroinformatics Node Special Interest Group on Neutrally-inspired Engineering and was co-guest editor of a Special Topic in Frontiers of Neuroscience. He has published over 70 articles in peer-reviewed journals and conferences. His research focuses on the design of intelligent embedded systems to support self-repairing capabilities.



Yuhua Li received the Ph.D. degree in General Engineering from the University of Leicester, UK. He was with Manchester Metropolitan University and then the University of Manchester from 2000 to 2005 as a Senior Research Fellow and a Research Associate, respectively. He was a Lecturer with the School of Computing and Intelligent Systems, University of Ulster, UK from 2005 to 2014. He is currently a lecturer with the School of Computing, Science and Engineering, University of Salford, UK. His current research interests include pattern recognition, machine learning, knowledge-based systems, data science, and condition monitoring and fault diagnosis.

Liam P. Maguire received M.Eng. and Ph.D. degrees in Electrical and Electronic Engineering from the Queen's University of Belfast, UK, in 1988 and 1991, respectively. He is currently Dean of the Faculty of Computing and Engineering at the University of Ulster and also Director of the Intelligent Systems Research Centre. His research interests are in two primary areas: fundamental research in bio-inspired intelligent systems (e.g. spiking neural networks) and the application of existing intelligent techniques in different domains. He is the author of over 200 research papers and has an established track record of securing research funding.