International Journal of Innovative Computing, Information and Control
Volume **15**, Number **5**, October **2019**

ICIC International ©2019 ISSN 1349-4198
pp. **1821–1834**

NETWORK-ON-CHIP WITH EQUALITY-OF-SERVICE: A LOCAL FAIR RUNTIME ARBITRATION METHOD FOR GLOBAL FAIR BANDWIDTH SHARE

Faizal Arya Samman[1,*] and Thomas Hollstein[2,3]

[1]Department of Electrical Engineering
Universitas Hasanuddin
Jl. Poros Malino Km. 6, Bontomarannu, South Sulawesi 92171, Indonesia
[*]Corresponding author: faizalas@unhas.ac.id

[2]Faculty of Informatics and Engineering Sciences
Frankfurt University of Applied Sciences
Nibelungenplatz 1, D-60318 Frankfurt am Main, Germany

[3]Department of Computer Systems
Tallinn University of Technology
Akadeemia tee 15A, Tallinn 12618, Estonia

Received December 2018; revised April 2019

Abstract. *This paper presents a network-on-chip (NoC) with equality-of-service (EoS), having local fair runtime arbitration schemes to perform global fair bandwidth shares between packets flowing in the NoC. The proposed throughput-balance-aware (TBA) arbitration scheme selects competing packets to access the same router output port based on flows. The TBA method just requires one pipeline arbitration stage, resulting in a low routing cycle. The flow detection is made locally at runtime and is enabled by the attachment of a local identity-tag (ID-tag) on each header flit of a packet. The scheme enables the flit-level packet flow overlapping, which is the main advantageous characteristic of our NoC. A hotspot traffic scenario is used to test the NoC performance. The simulation test presents that the proposed arbitration method performs a global fair bandwidth sharing effectively.*

Keywords: Network-on-chip, Equality-of-service (EoS), Multicore processor, Fair arbitration method, Global fair bandwidth share

1. **Introduction and Motivation.** The number of processing element (PE) cores on a single chip will be significantly increased in the future. The main reason to increase the number is to improve processor system performance and system reliability. Chip-level many core processor (CMP) systems have been nowadays a challenging solution for high performance computing systems.

Since the number of PE cores increases, then the need for better communication infrastructure is a must. The interconnection between processor cores can be implemented by using bus systems. However, the use of the traditional bus interconnection will lead to bottleneck performance problem, especially when the number of processor cores is high, or even reaches more than hundred cores. Direct point-to-point interconnection indeed can solve the problem, but it can extremely increase the interconnect complexity.

Another useful and natural solution can be suggested by implementing on-chip network together with the processor on a single chipset. The bandwidth of a mesh-based network-on-chip (NoC) is scalable, because the increase of PE cores will be followed by the increase

DOI: 10.24507/ijicic.15.05.1821

total bandwidth constructed from many interconnected local bandwidths. Because of that interesting bandwidth figure, multi core or many core processing system integrated with the mesh-based network-on-chip (NoC) will be a powerful and sophisticated platform in the future.

Many core processor system interconnected with the NoC is presented in Figure 1. Using the NoC infrastructure, many core processor systems can also maintain the processor power at an accepted level. The systems can be operated at optimum working frequency, resulting in low chip temperature level and low power dissipation, without systems performance degradation.

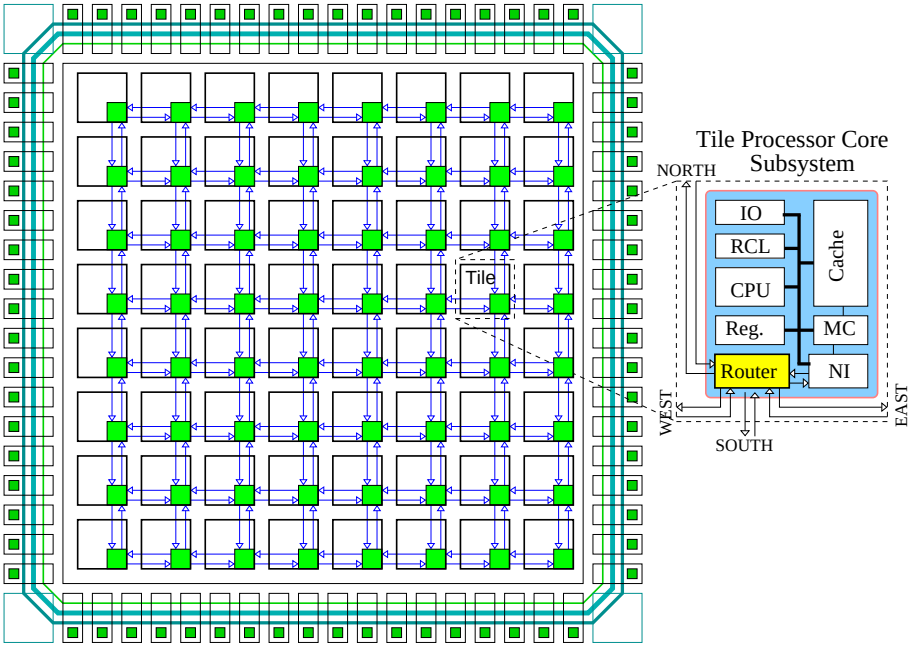


FIGURE 1. An NoC-based many core processor system

Packet routing and arbitration in NoC can potentially introduce a bottleneck performance problem, if they are not well organized. The solutions for the problem can be generally classified into two main approaches, i.e., adaptive output selection and input selection approaches. The adaptive output selection approach is related to adaptive routing algorithms [1]. Adaptive routing algorithms are proposed to solve the hotspot problem as well as to solve the deadlock problem due to faulty network links or nodes. However, adaptive routing, without adaptive input selection, cannot drain packets well from hotspot condition in the backward (previous) routers.

Beside the adaptive output selection, the hotspot problem can also be solved by using input selection approach [2]. The method used in this approach is also called *arbitration methods*. Arbitration method is a methodology to select one of many possible requests from input ports, which will be selected to access an output port at certain cycle period. The above input selection technique makes arbitration using packet-based switching, while our approach presented in this paper makes it using flit-based switching. The flit-based switching with flit injection control gives interesting performance profiles, such as packet delay that changes linearly as the data injection rate is increased.

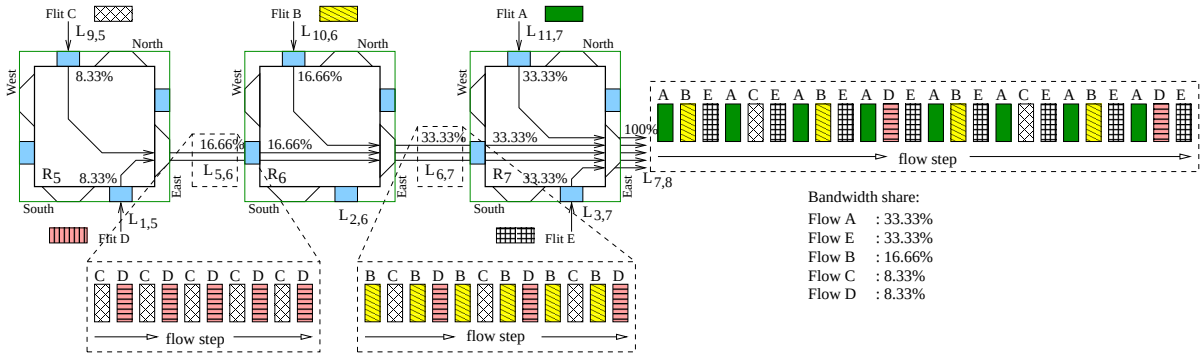
The most famous arbitration method, which is not only familiar in NoC community but also in Internet community, is the round-robin arbitration method. The use of the traditional round-robin arbitration method will lead to the unfair bandwidth allocation,

which accordingly results in a bottleneck performance problem. Other methods are ordered arbitration method [3] and programmable priority arbitration [4]. However, both methods give good quality-of-service (QoS), but they cannot provide equality-of-service (EoS). The above works should design extra logical module to provide the EoS.

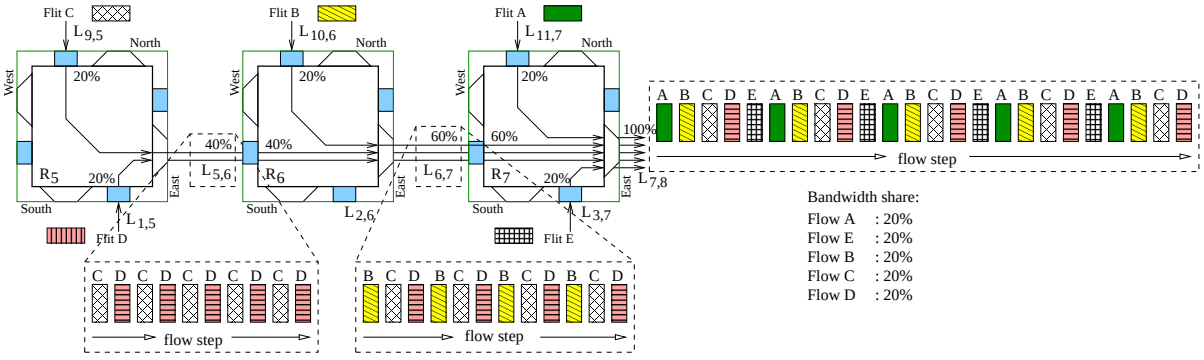
This paper presents a novel arbitration method that run locally at each router but it performs a global equality-of-service (EoS). To present the idea clearly, the paper is organized as follows. Section 2 shows some related works and justifies the contribution of this paper. Section 3 presents the formulation of the problem. Section 4 presents the proposed concept to solve the problem and gives the router architecture, in which the arbitration method is implemented. Section 5 presents the simulation results. Finally, Section 6 concludes the work.

2. **Related Works and Contribution.** Without considering the number of stream flows from different input ports, a simple arbitration method such as round-robin arbitration [5] cannot balance bandwidth assignments fairly between the data/stream flows as presented in Figure 2(a). Each input port could have more than one flows. Hence, making selection per active input port will result in an unfair arbitration.

We are motivated to design a fair global bandwidth share, which is implemented through a runtime local arbitration without pre-scheduling mechanism to give potential lower logic area and time-delay overhead. A slack-based global and dynamic prioritization of data streams is proposed in [6]. However, the work requires an overlay network. It combines a



(a) The problem



(b) The proposed solution

FIGURE 2. (a) The problem of the unfair throughput by using the flit-by-flit round arbitration (FRA) method; (b) the balanced traffic flow after using the solution of the throughput-balance-aware (TBA) arbitration algorithm

scheduling unit at local arbitration machine with a global scheduling unit of entire logical transmissions to guarantee end-to-end throughput. Our method does not use that extra global scheduler.

An NoC that provides performance isolation for both safety-critical traffic and low latency for best-effort traffic at the same time is proposed in [7]. The work gives contrast method compared to existing methods. It prioritizes best-effort over guaranteed throughput traffic. The switching priority is only made when necessary. The concept provides sufficient performance isolation among different critical levels.

Critical packet services are becoming important issue in NoC-based MPSoC. Critical packets should be delivered within predefined limits of time [8]. Time-division multiplexing (TDM) method is one of the proposed techniques [9]. However, TDM method usually uses pre-scheduling algorithm to configure virtual TDM switching circuits.

The unfair arbitration can be solved using a distance-based probabilistic (DBP) method [10]. In the DBP method, the output selection is probabilistically determined based on the weight of the input requests. The arbitration weight is determined by using some metrics such as hop count or the distance between the current position of a packet to the packet source node. The two extra weight computations, i.e., weight calculation and weight summation are made before the arbitration takes place. This method can potentially increase delay time to make the arbitration.

There are also many arbitration methods proposed for networks-on-chip such as priority-based arbitration scheme [11, 12, 13]. The same scheme with dynamic approach can also be used to improve the performance figure [14, 15, 16]. The priority-based arbitration methods can still potentially perform an unfair arbitration between packets, because certain packets will be given the first priority to flow than other packets. This arbitration scheme is commonly used when the NoC routes packet requesting bandwidth guaranty for specific applications, such as audio and video processing applications.

Another method to overcome the unfair arbitration is by using adaptive weighted round-robin (AWRR) arbitration [17]. The AWRR method uses also arbitration weight at each input port, which is reconfigurable. The arbitration weight, the same as the previous approach, is determined during reconfiguration-time (after application mapping), or at pre-application runtime (before application running), by computing the contenting packet requesting the same output.

We contribute to a new runtime local arbitration scheme called *Throughput-Balance-Aware arbitration* (TBA) strategy that provides equality-of-service (EoS) for global fair bandwidth share. The TBA method does not need the extra weight computations or probabilistic analysis, which is made before application runtime. This arbitration strategy is accordingly simple to implement locally in the NoC router at runtime, because of the use of ID-based routing organization. Therefore, the unique condition in our TBA arbitration method, compared to other methods in the literature, is that the NoC router switches packets flit-by-flit with organized local ID labels. Different packets can be interleaved at flit level in the same input buffer. Each flit is assigned with a local ID label from a routing table at each input port. Flits belonging to the same packet group will be allocated to the same local ID slot. The arbiter will only check the number of used ID tags from the routing table to identify the number of flows from the input ports. The modification on our current arbiter unit (modelled in VHDL) is therefore simply applicable.

3. **The Problem Formulation.**

Before we start defining the problem encountered in this paper, let us overview a few definitions as follows.

Definition 3.1. *NoC consisting of N_{node} number of nodes will have a set of NoC* ***Router*** $\Re = \{R_1, R_2, \ldots, R_{N_{node}}\}$ *or* $R_c \in \Re$ *where* $c = \{1, 2, \ldots, N_{node}\}$.

Definition 3.2. ***Communication link*** $L_{i,j} \in \Lambda$ *is a communication media connecting router node* R_i *and* R_j *where* $R_i, R_j \in \Re$*, and* $i, j = \{1, 2, \ldots, N_{node}\}$.

Each link will have local bandwidth, which is defined as follows.

Definition 3.3. *The* ***Rest Available Local Bandwidth*** *of each NoC link* $L_{i,j}$ *as defined in Definition 3.2, is defined as* $B_L^{(i,j)} \leq B_{\max}$*, which measured in* fpc *unit, or percentage of* $B_{\max}$*, where* $B_{\max}$ *is the maximum bandwidth of each link.*

The available local bandwidth depends on two factors, i.e., the number of flows routed to the link $L_{i,j}$ connected an output port m of the router R_i, and the remaining local bandwidth assigned to the link $L_{i,k}$ due to a contention to share an output port in the Router R_j. Now let us see the following definition.

Definition 3.4. *A* ***Set of Requests to an Output Port*** m *from input ports of the router* R_j *is defined as* $\Phi_{m(j)} \leq N_p$*, where* N_p *is the number of input/output ports in the router* R_j.

For a mesh router case, which has five I/O ports, then $m(j) \in \{E, N, W, S, L\}$, where E = EAST, N = NORTH, W = WEST, S = SOUTH, and L = LOCAL. According to Definition 3.2 and Definition 3.4, then the available local bandwidth of a link $L_{i,j}$ connecting router R_i to router R_j, when there is no contention at the next router R_j and so on, is defined as

$$B_L^{(i,j)} = \Phi_{m(j)}^{-1} B_{\max}. \tag{1}$$

When a packet flowing through the link $L_{i,j}$ will use an output port m in router R_j, and there is contention to access the output port m, then the remaining local bandwidth of link $L_{i,j}$ cannot be set to the maximum value as presented in Equation (1). It will be reduced to any lower value depending on the number of contention to the contenting output port m in router R_j. If there is again contention at an output in the next router R_k used by the packet, then local bandwidth of link $L_{j,k}$ is

$$B_L^{(j,k)} = \Phi_{m(k)}^{-1} B_{\max}. \tag{2}$$

Therefore, the remaining local bandwidth of the link $L_{j,k}$ used by the packet will be

$$B_L^{(i,j)} = \Phi_{m(j)}^{-1} B_L^{(j,k)}. \tag{3}$$

By substituting Equation (2) to Equation (3), then we have

$$B_L^{(i,j)} = \Phi_{m(j)}^{-1} \Phi_{m(k)}^{-1} B_{\max}. \tag{4}$$

Figure 2(a) illustrates the problem of the unfair bandwidth consumptions because of the different distances of each packet flow from an output port, where the packets compete to use it. As presented in the figure, packets A, B, C, D, E and F compete to use the same EAST output port of the router node R_7. Assuming the packets are expected to flow with maximum throughput, then by using a round arbitration between input ports NORTH, WEST and SOUTH at the router node R_1, packets A and F will use 33.33% link bandwidth, respectively. The remaining 33.33% bandwidth is shared between packets B, C, D, and E.

From router node R_6, based on the same situation, i.e., three flows from 3 different input ports, i.e., NORTH, WEST and SOUTH access the same EAST output port, which connected to the link $L_{6,7}$. The remaining available local bandwidth of the link $L_{6,7}$ is $B_L^{(6,7)} = 33.33\%$, which is then assigned to 16.66% to packet from NORTH input port, and also 16.66% to packets C and D from WEST input port.

Link $L_{5,6}$, which is connected to the EAST output port of the router R_5 has available bandwidth $B_L^{(5,6)} = 16.66\%$. At the EAST output port in the router R_5 packets C and D will share fifty-fifty the remaining local bandwidth of 16.66%, i.e., respectively 8.33% for each of them. We can see that packets farther away from an output port, in which packets sequentially and collectively compete to flow from the previous router nodes, will use the least bandwidth.

When a packet flows from a source router node, eventually assigned as R_1, to a target router node via K number of intermediate nodes, such that the target router node is eventually assigned as R_K, where the set of links used by the packet is $\{1, 2, 3, \ldots, K, K-1\}$ then according to Definition 3.2, Definition 3.3, and Definition 3.4, Equation (2) and Equation (3), the local bandwidth of the first output port to which the packet is routed is formally modeled as follows.

$$B_L^{(1,2)} = \Phi_{m(2)}^{-1}\Phi_{m(3)}^{-1}\Phi_{m(4)}^{-1}\cdots\Phi_{m(K-1)}^{-1}\Phi_{m(K)}^{-1}B_{\max}. \tag{5}$$

If we see the problem illustrated in Figure 2(a), then according to Equation (5) we can model the problem as follows. The local bandwidths of link $L_{11,7}$ (used by packet A), $L_{6,7}$ (used by packets B, C and D) and $L_{3,7}$ (used by packet E) connected to router R_7 are the same, i.e., $B_L^{(11,7)} = B_L^{(6,7)} = B_L^{(3,7)} = \Phi_{E(7)}^{-1}B_{\max} = 3^{-1}B_{\max}$. Or in percentage, we can write $B_L^{(11,7)} = B_L^{(6,7)} = B_L^{(3,7)} = 33.3333\%B_{\max}$.

The local bandwidths of link $L_{10,6}$ (used by packet B) and $L_{5,6}$ (used by packets C and D) connected to router R_6 are the same, i.e., $B_L^{(10,6)} = B_L^{(5,6)} = \Phi_{E(6)}^{-1}\Phi_{E(7)}^{-1}B_{\max} = 2^{-1}3^{-1}B_{\max} = 6^{-1}B_{\max}$. Or in percentage, we can write $B_L^{(10,6)} = B_L^{(5,6)} = 16.6667\%B_{\max}$.

The local bandwidths of link $L_{9,5}$ (used by packet C) and $L_{1,5}$ (used by packet D) connected to router R_5 are the same, i.e., $B_L^{(9,5)} = B_L^{(1,5)} = \Phi_{E(5)}^{-1}\Phi_{E(6)}^{-1}\Phi_{E(7)}^{-1}B_{\max}$. Since there are two competing packets to access the E output port at the router R_5, then $\Phi_{E(5)} = 2$. Substituting the values of $\Phi_{E(6)}$ and $\Phi_{E(7)}$ obtained before into the previous equation, then we have $B_L^{(9,5)} = B_L^{(1,5)} = 2^{-1}2^{-1}3^{-1}B_{\max} = 12^{-1}B_{\max}$. Or in percentage, we can write $B_L^{(9,5)} = B_L^{(1,5)} = 8.3333\%B_{\max}$.

This section presents the formal model of the problem. In particular, Equation (5) shows the example mathematical model, in which a packet flow line entering more competing outgoing ports will experience more bandwidth reduction. The following section proposes solution of the problem, which is described conceptually and is illustrated with its corresponding router architecture.

4. **Problem Solution and NoC Router Architecture.**

This paper presents a simple and feasible solution to overcome the unfair bandwidth allocation, which is realized during application runtime. All packet flows will be globally assigned to the same bandwidth space, regardless the distance of a source node to the last or target node, whose output port competitively accessed by some flows. The new arbitration strategy is proposed based on the number of packet flows, where at each on-chip router, the flows can be easily identified based on ID-tag utilization in our NoC.

4.1. **The conceptual solution.**

Our proposed arbitration scheme is applied by simply rotating its selection flit-by-flit between active requests from any input ports and concurrently considering the number of flows from the input ports. In other words, the input selection is made per flow, not (only) per active input port. The number of flows is obtained directly from the runtime reconfigurable routing engine table in our NoC. So, extra computing in the arbiter hardware, and extra weight computations for reconfiguration before application run are not required in our proposed TBA arbitration method.

Figure 2(b) illustrates the solution to solve the unfair flow problem by using the TBA arbitration strategy as presented in Figure 2(a). By using the proposed TBA arbitration strategy, all competing packets use fairly the same bandwidth. We can see that in the router R_7, the arbiter unit at the EAST output port detects 5 flows from 3 input ports, i.e., 1 from NORTH, 1 from SOUTH and 3 from WEST input port. In this condition, the arbitration algorithm in the EAST arbiter unit adapts to give more routing selection to the flows from the WEST input port. Thus, the arbitration algorithm allocates 20% bandwidth to the NORTH input port, also 20% bandwidth to the SOUTH input port, and 60% bandwidth to the WEST input port.

The proposed TBA arbitration algorithm implemented on the arbiter units at each output port of the NoC routers can solve effectively the unfair routing selection due to the sequential and collective contentions on the routing paths.

4.2. **Router and arbiter architecture.** The microarchitecture of a mesh router that supports our proposed arbitration scheme is illustrated in Figure 3. In general, the router consists of five input/output ports and four main modules. They are implemented on each input and output port. The main modules at input port include a routing engine (RE), a FIFO buffer, and output port includes a multiplexor (Mux) and an arbiter unit (AU). The FIFO is used to buffer data. The RE routes packet flit-by-flit. In our current NoC version, a static XY routing is used. The arbiter selects a flit to flow through the multiplexer. The proposed TBA arbiter algorithm is implemented on each arbiter unit.

The architecture of the on-chip router consists of five input/output ports, labeled as port number 1, 2, 3, 4 and 5. In our mesh router design, each aforementioned port number is assigned respectively as East, North, West, South and Local port. The local port (port number 5) is connected to a processing element via an on-chip network interface. The rest ports are connected to other routers.

The arbitration function at an output port is used to select a flit from a certain input port to flow via the multiplexor at the output port. The arbiter unit at the output port will then send a select signal and concurrently send a read signal to enable data flit reading from the data slot/register of the selected routing engine. This arbitration scheme is the main topic of this paper and will be further discussed in the following subsections.

4.3. **Control paths for the number of flows.** There are two main paths in the router, i.e., data paths D_k, $k \in \{1, 2, 3, 4, 5\}$ and control paths. The control paths consist of 3 paths, i.e., routing request paths $r_{j,k}$, $j, k \in \{1, 2, 3, 4, 5\}$, arbitrations paths $a_{j,k}$, $j, k \in \{1, 2, 3, 4, 5\}$, and the number of flows paths $F_{j,k}$, $j, k \in \{1, 2, 3, 4, 5\}$. All paths are interconnected in a crossbar structure among the routing engines and the arbiter units.

Figure 3 shows an example interconnected from four input ports, i.e., port number 2, 3, 4 and 5 to the output port number 1. As illustrated in the figure, the arbiter unit at the output port 1 is connected to routing request paths and signal paths indicating the number of flows ($F_{j,k}$) from the four input ports. The arbiter unit connects also its four arbitration paths to every four input port. The number of flows indicates actually the number of different packets interleaved in the input ports (FIFO buffers). The number can be easily identified in the NoC from the number of the used ID-tags in a dynamically programmable routing table at each routing engine. The signal paths of the number of flows ($F_{j,k}$) are the main unique feature of our NoC router, which we call it as XHiNoC 2.0.

The $F_{j,k}$ and $r_{j,k}$ signals are then used by the arbiter units to make an arbitration scheme called the throughput-balance-aware (TBA) arbitration method. We designed two types of NoC routers for comparative study: one router having arbiter units, which do not use the number of flows paths and signals, while the other one having arbiter units that use them. Both are further explained in the following subsection.

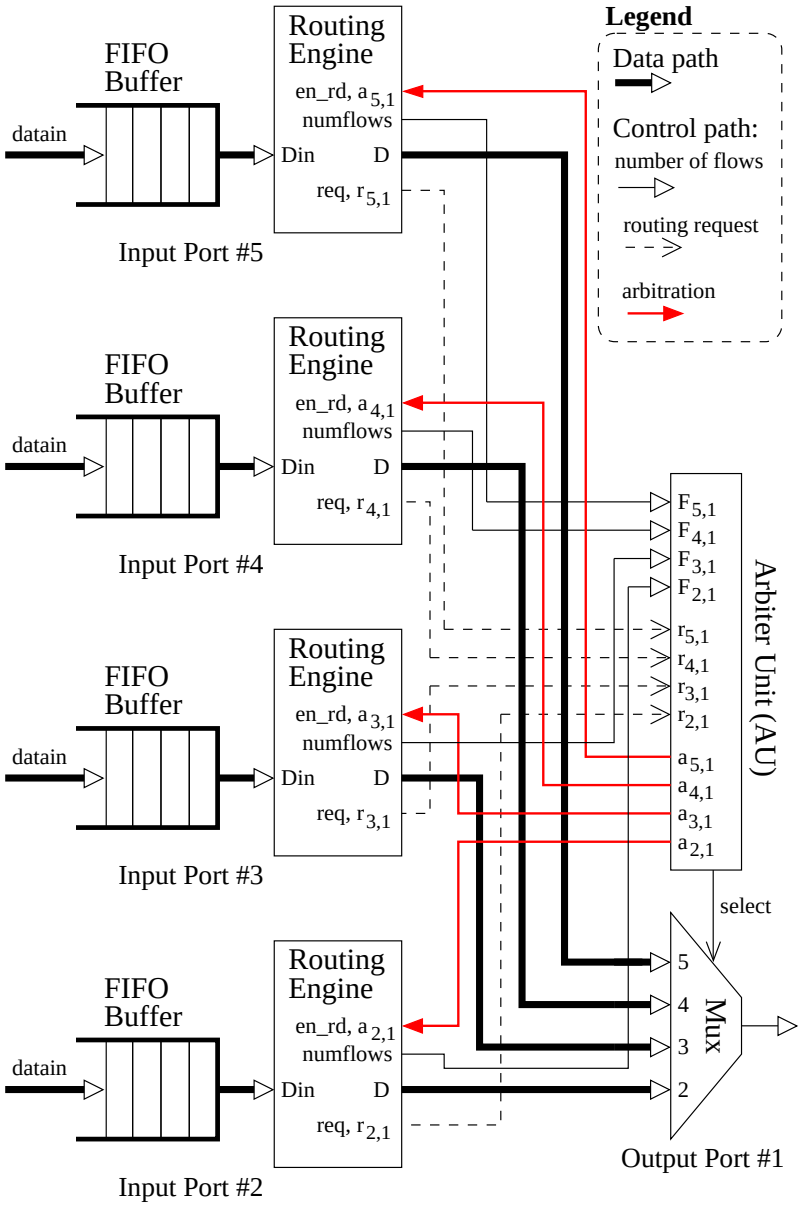


FIGURE 3. The example control and data paths in the on-chip router with detail port labels at the arbiter unit

4.4. **Arbitration unit.** As mentioned before, the proposed arbitration scheme is implemented directly into the arbiter units. We have designed two NoC routers. One of them has its arbiter units implementing the flit-by-flit round arbitration (FRA) algorithm, and the other one having arbiter units which is implemented using the throughput-balance-aware (TBA) arbitration algorithm.

4.4.1. *Flit-by-flit round arbitration algorithm.* In the FRA arbitration method, the selection to forward a flit to an output port m is rotated among the active input ports in the router with flit by flit basis. Active input port is an input port having a flit requesting a routing to the output port m. Thus, this arbitration scheme does not use the number of flows paths and signals.

4.4.2. *Throughput-balance-aware arbitration algorithm.* The TBA arbitration scheme is principally implemented nearly similar to the FRA arbitration method. The only difference is to use the number of flows paths and signals. The TBA arbitration scheme works based on the following rules.

1) The TBA arbiter unit at output port k rotates its selection among a set of active input ports flit by flit.
2) While rotating its selection and the selection ends at an active input port j, the arbite unit proceeds concurrently the number of flows signal $F_{j,k}$ from the selected active input port j.
3) As $F_{j,k}$ is greater than one, namely H number, then the selection is maintained at this active input port j for H number of selection cycle times.

5. **The Performance Test.** To see the impact of the proposed bandwidth-balance-aware arbitration method on the NoC's performance, experimental simulations are presented in this paper by using two scenarios, i.e., a hotspot and a transpose traffic scenario. Static or deterministic XY routing algorithm in the simulations is used in this simulation, because it makes the analysis on the traffic easier. Therefore, the positive impacts of the proposed *Throughput-Balance-Aware arbitration* (TBA) method over the *Flit-by-flit round arbitration* (FRA) method can be analytically concluded.

There are two measurement metrics that we used in the simulation. The first is the transient response of data rate for each individual communication pair. The data rates are measured in order to see how the arbitration methods allocate their selections to drain data stream input ports of the NoC routers. The data rate is measured in flit per cycle. *Flit per cycle* (fpc) is a unit to measure a data flow in the NoC links and nodes, which is independent from three aspects, i.e., silicon technology used to implement the NoC, switch architecture and the bit-width of data words used in the NoC packet format.

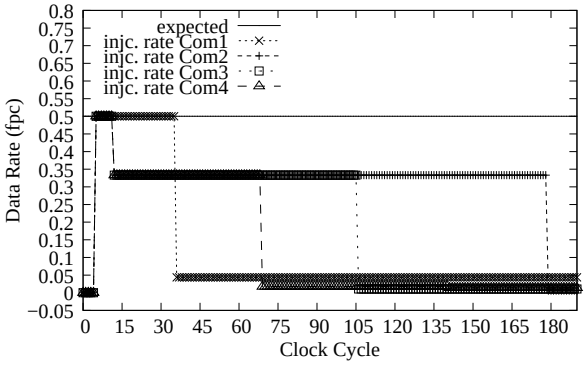
The second metric is the tail flit latency. We measured this metric in order to see the effect of the individual data rates on the latency of each communication with different workload sizes. In our NoC router, we implement a *link-to-link flit flow control* mechanism. This mechanism is used to avoid flit loses, or flits will not be dropped during congestion.

5.1. **Hotspot scenario.** The hotspot traffic scenario is used in this simulation because the traffic is very interesting to show the larger number of contentions in a hotspot node. In this scenario, 15 nodes in a 4×4 mesh NoC will send data to a single target node at the north-east network edge. We use static or deterministic XY routing algorithm in the experiments, because it will ease the analysis of the traffic.

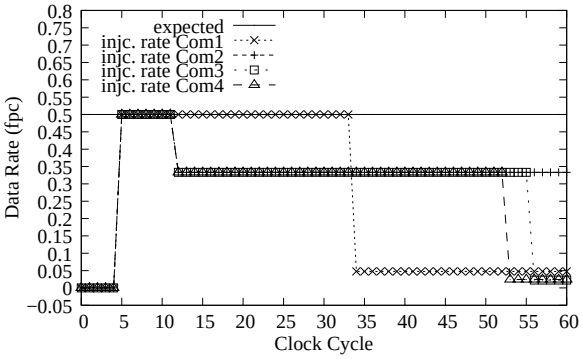
5.2. **Data rates measurements under saturated condition.** In this section, we will present the data rate measurement under saturated condition. When data are injected to the NoC with relative higher injection rates, then the NoC could be saturated. Saturating condition is condition in which the injection rates of packets reach the maximum bandwidth capacity of an NoC. Saturating condition can also happen because of contentions of packets to utilize the same output link in the NoC.

In this case, the data are injected at each source node with rate of 0.5 fpc (*flit per cycle*). In the NoC router, routing and arbitration stages are made in two clock cycles. Therefore, by injecting flits with the rate of 0.5 fpc, the data flow rate will approach the maximum bandwidth of each communication link. Hence, using that injection rate, the NoC becomes saturated. Saturating condition is a very important case to measure and analyze the NoC performance effectively.

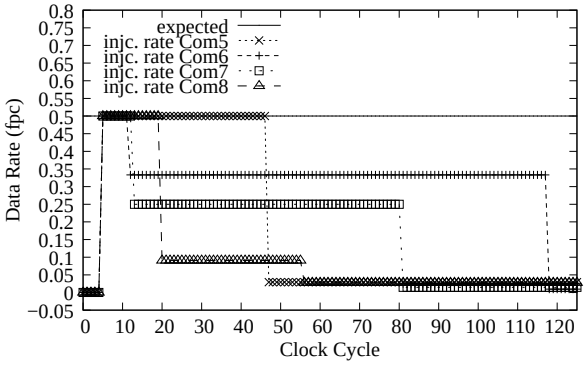
Figure 4 presents the data injection rate of the communication labels 1-15. Figure 4(a), Figure 4(c), Figure 4(e) and Figure 4(g) present the transient response curves of the data



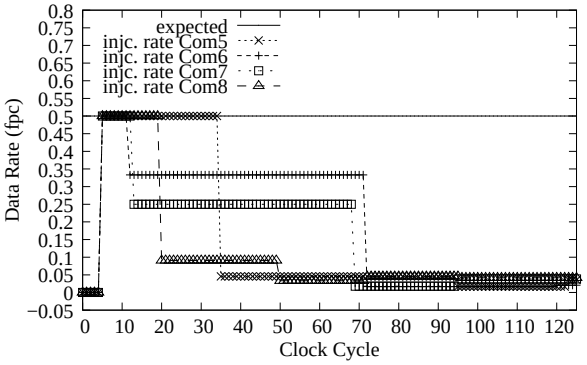
(a) Com. 1-4 with FRA



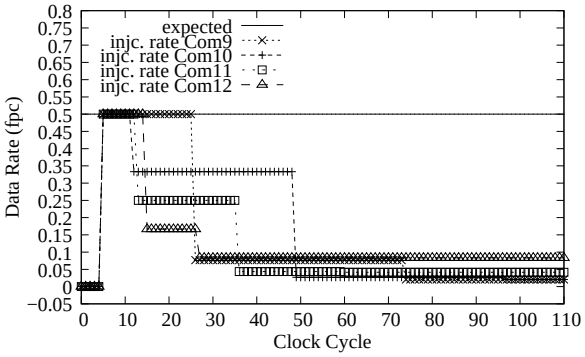
(b) Com. 1-4 with TBA



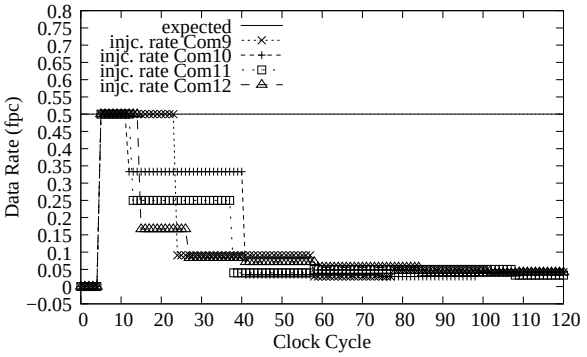
(c) Com. 5-8 with FRA



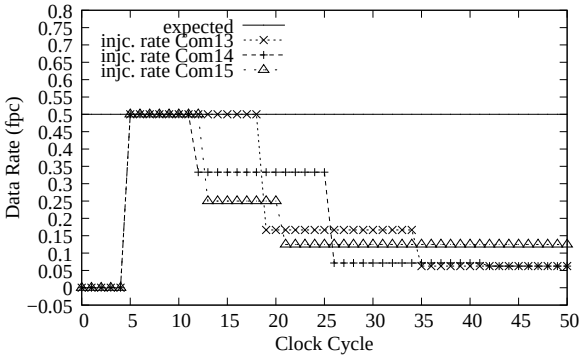
(d) Com. 5-8 with TBA



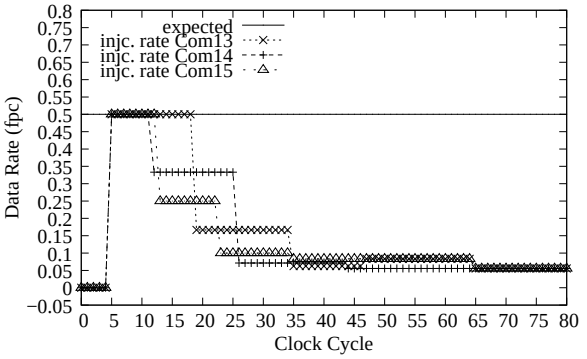
(e) Com. 9-12 with FRA



(f) Com. 9-12 with TBA



(g) Com. 13-15 with FRA



(h) Com. 13-15 with TBA

FIGURE 4. The transient response of data injection/receiving rate of Com. 1-15 in the hotspot scenario

injection rate using the FRA method. It seems that by using the FRA method, the data injection rates of some communication labels are not equal. At the first time, the injection rates start with 0.5 fpc (*flit per cycle*). After the packets start competing to access the same output port, each of them decreases to a certain data injection rate. Competing communication pairs further from the hotspot point will experience more injection rate reductions.

Figure 4(b), Figure 4(d), Figure 4(f) and Figure 4(h) present the transient response curves of the data injection rate using the TBA method. We can see that by using the TBA method, the data injection rates of all communication pairs are the same. At the first time, the injection rates start with 0.5 fpc (*flit per cycle*). After the packets start competing to access the same output port, they move to the same steady-state injection rate point to 0.0333 fpc.

5.3. **Latency with different workload sizes.** In this simulation, a range of 500 until 10000 flits are injected to the NoC through the local input ports 15 source nodes in the hotspot scenario. The number of cycle periods required to accept a tail flit at each target node is measured. The cycle count is started from the time at which the first flit is injected at the target node. The latency curves for all communication labels are shown in Figure 5.

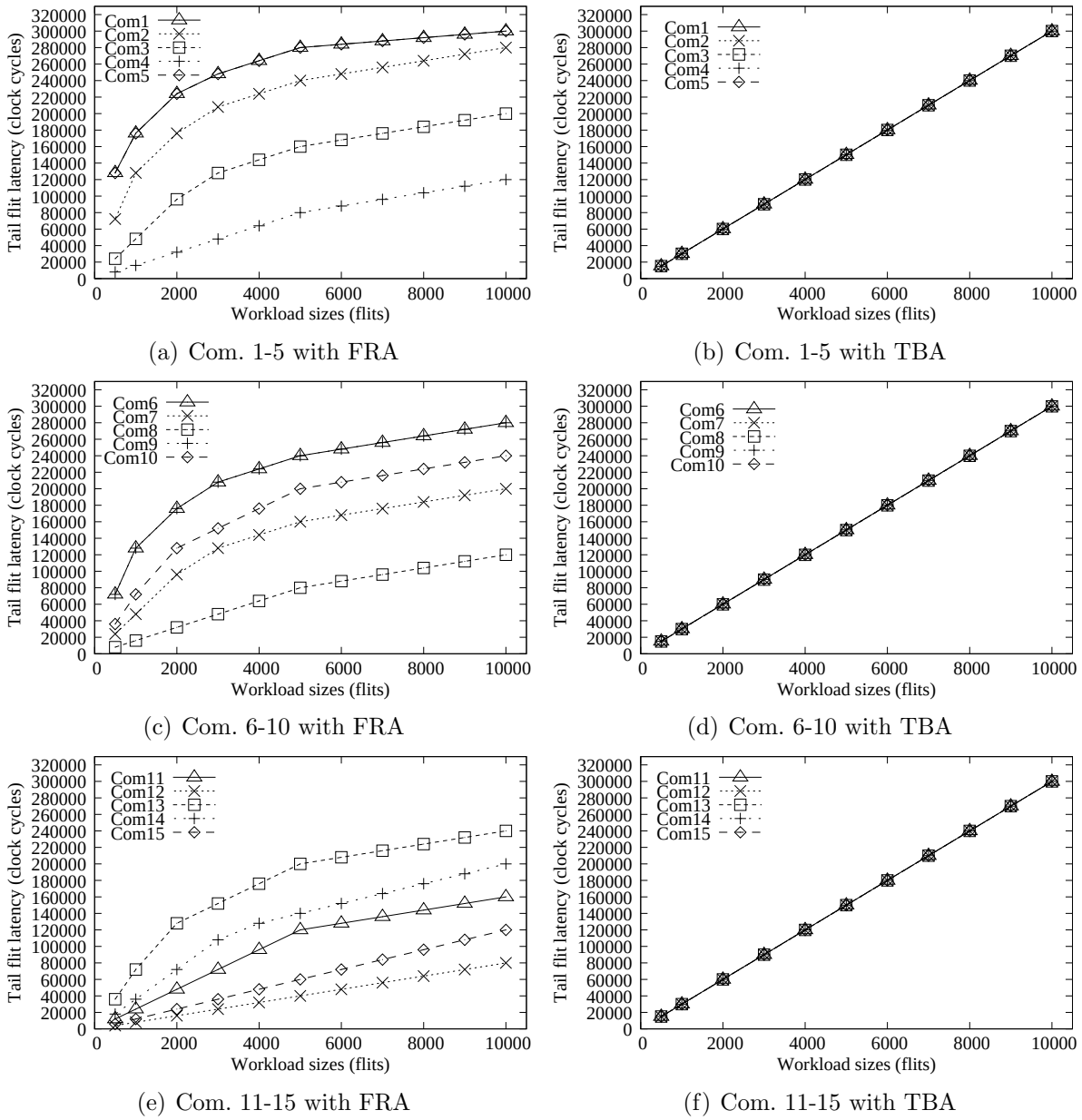
Figure 5(a), Figure 5(c) and Figure 5(e) present respectively the latency curves for all communication labels in the hotspot traffic scenario using the flit-by-flit round arbitration (FRA) method. The results with the FRA method meanwhile present different latency curves for several communication labels. The measured latency values depend not only on the data rate of the respected communication label but also on the distance between the source and target node of the communication points. Because the latency curves for each communication pairs are not uniform, then the data injection rate during the steady-state time domain is not equal for each packet.

Figure 5(b), Figure 5(d) and Figure 5(f) present respectively the latency curves for all communication labels in the hotspot traffic scenario using the TBA arbitration method. The latency curves obtained from the TBA method present a linearly increase tendency. These performance figures are very exciting to show the role of the TBA method to fairly share the bandwidth. As shown the figure, all communication labels have the same latency curves. This result is in line with the data rate performance figures presented before using this TBA method, where the injection rates of all communication labels are the same.

5.4. **Traffic analysis on the hotspot scenario.** Figure 6(a) and Figure 6(b) present the bandwidth distributions in the mesh 4×4 NoC by using the flit-by-flit round arbitration and the throughput-balance-aware arbitration, respectively. The packets are routed using deterministic XY routing. Hence, the traffic can be easily analyzed. From Figure 6(a) we can see that the bandwidth share is not uniform, when we use the flit-by-flit round arbitration. This traffic rate condition is due to the aforementioned problem described in Section 3 and illustrated in Figure 2(a).

By using the TBA arbitration algorithm, the 15 flows share fairly the bandwidth of the link globally with uniform data rate as shown in Figure 6(b). The paths of the packet flow in the figure as well as the data rates are measured from the simulation.

6. **Conclusions.** This paper has presented a novel arbitration scheme called throughput-balance-aware (TBA) arbitration, which works based on flow selection rather than input selection. The TBA arbitration scheme is implementable on NoC routers, which route packets using wormhole cut-through switching method, i.e., NoC routers which switch their packet flit-per-flit and can interleave their packets at flit level.



(a) Com. 1-5 with FRA

(b) Com. 1-5 with TBA

(c) Com. 6-10 with FRA

(d) Com. 6-10 with TBA

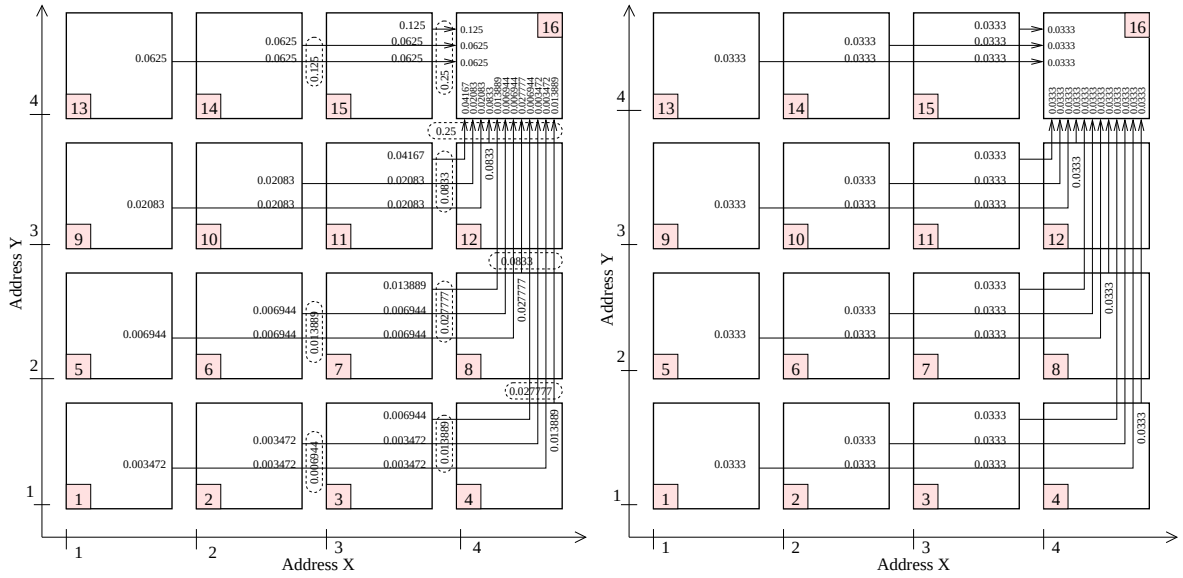
(e) Com. 11-15 with FRA

(f) Com. 11-15 with TBA

FIGURE 5. Tail flit latency measurement for different workloads in hotspot scenario

By using the aforementioned switching method, the TBA arbitration scheme is applied by rotating its selection flit-by-flit between active requests from input ports and concurrently considering the number of flows from the input ports. Hence, input ports having more than one flow will receive a fair access to an output port, targeted by other flows from different input ports. As those arbitration schemes simultaneously work in all NoC routers, the fair bandwidth assignments between competing packets in the NoC can be realized well.

The performances of the proposed TBA and the FRA arbitration schemes have been simulated using a VHDL simulator. A hotspot testbench scenario, as the best traffic condition to test the NoC performance in this case, is performed to analyze the bandwidth assignments between competing packets in the NoC routers. The simulation results have presented that the previous FRA arbitration method cannot perform the fair bandwidth



(a) Flit-by-flit round arbitration (b) Throughput-balance-aware arbitration

FIGURE 6. The bandwidth share in the hotspot traffic scenario

allocation, where in general, packets further from the contenting output router node consume less bandwidth as theoretically explained in the introductory sections. Meanwhile, the TBA arbitration scheme can allocate NoC bandwdths between competing packets fairly.

Acknowledgment. We gratefully acknowledge the Ministry for Research, Technology and Higher Education of the Republic of Indonesia (by *Direktorat Riset dan Pengabdian Masyarakat – DRPM*) for funding and supporting the research work under the scheme of "The National Strategic Outstanding Research Grant" (*Hibah Penelitian Unggulan Strategis Nasional or PUSNAS*) in the years 2017-2019.

REFERENCES

[1] C. Wang, W.-H. Hu and N. Bagherzadeh, Congestion-aware network-on-chip router architecture, *Proc. of the 15th CSI International Symp. on Computer Architecture and Digital Systems (CADS)*, pp.137-144, 2010.

[2] D. Wu, B. M. Al-Hashimi and M. T. Schmitz, Improving routing efficiency for network-on-chip through contention-aware input selection, *Proc. of the 2006 Asia and South Pacific Design Automation Conference (ASP-DAC'06)*, pp.36-41, 2006.

[3] Y. Liu, X. Guan, Y. Yang and Y. Yang, An asynchronous low latency ordered arbiter for network on chips, *Proc. of the 6th International Conference on Natural Computation (ICNC)*, pp.962-966, 2010.

[4] G. Dimitrakopoulos, N. Chrysos and K. Galanopoulos, Fast arbiters for on-chip network switches, *Proc. of the IEEE International Conference on Computer Design (ICCD)*, pp.664-670, 2008.

[5] E. Shin, V. M. III and G. Riley, Round-robin arbiter design and generation, *Proc. of the 15th International Symp. on System Synthesis*, pp.243-248, 2002.

[6] A. Kostrzewa, S. Saidi and R. Ernst, Slack-based resource arbitration for real-time networks-on-chip, *Design, Automation Test in Europe Conference Exhibition (DATE)*, pp.1012-1017, 2016.

[7] S. Tobuschat and R. Ernst, Providing throughput guarantees in mixed-criticality networks-on-chip, *Proc. of the 30th IEEE International System-on-Chip Conference (SOCC)*, pp.292-297, 2017.

[8] P. Petrakis, M. Abuteir, M. D. Grammatikakis, K. Papadimitriou, R. Obermaisser, Z. Owda, A. Papagrigoriou, M. Soulie and M. Coppola, On-chip networks for mixed-criticality systems, *Proc. of the 27th IEEE International Conference on Application-Specific Systems, Architectures and Processors (ASAP)*, pp.164-169, 2016.

[9] A. Psarras, J. Lee, I. Seitanidis, C. Nicopoulos and G. Dimitrakopoulos, PhaseNoC: Versatile network traffic isolation through TDM-scheduled virtual channels, *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol.35, no.5, pp.844-857, 2016.

[10] M. M. Lee, J. Kim, D. Abts, M. R. Marty and J. W. Lee, Probabilistic distance-based arbitration: Providing equality of service for many-core CMPs, *Proc. of the 43rd Annual IEEE/ACM International Symp. on Microarchitecture (MICRO)*, pp.509-519, 2010.

[11] C.-H. Lu, K.-C. Chiang and P.-A. Hsiung, Round-based priority arbitration for predictable and reconfigurable network-on-chip, *Proc. of the International Conference on Field-Programmable Technology (FPT)*, pp.403-406, 2009.

[12] C.-H. Chan, K.-L. Tsai, F. Lai and S.-H. Tsai, A priority based output arbiter for NoC router, *Proc. of the IEEE International Symp. on Circuits and Systems (ISCAS)*, pp.1928-1931, 2011.

[13] C. Wissem, B. Attia, A. Noureddine, A. Zitouni and R. Tourki, A quality of service network on chip based on a new priority arbitration mechanism, *Proc. of the International Conference on Microelectronics*, pp.1-6, 2011.

[14] D. Park, R. Das, C. Nicopoulos, K. Jongman, N. Vijaykrishnan, R. Iyer and C. Das, Design of a dynamic priority-based fast path architecture for on-chip interconnects, *Proc. of the 15th Annual IEEE Symp. on High-Performance Interconnects*, pp.15-20, 2007.

[15] J. Wang, Y. Li, Q. Peng and T. Tan, A dynamic priority arbiter for network-on-chip, *Proc. of the IEEE International Symp. on Industrial Embedded Systems (SIES)*, pp.253-256, 2009.

[16] G. Dimitrakopoulos and E. Kalligeros, Dynamic-priority arbiter and multiplexer soft macros for on-chip networks switches, *Proc. of Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp.542-545, 2012.

[17] H. Park and K. Choi, Adaptively weighted round-robin arbitration for equality of service in a many-core network-on-chip, *IET Computers & Digital Techniques*, vol.10, no.1, pp.37-44, 2016.