electronics

MDPI

Article

Network-Cognitive Traffic Control: A Fluidity-Aware On-Chip Communication

Wen-Chung Tsai [1,*], Sao-Jie Chen [2], Yu-Hen Hu [3] and Mao-Lun Chiang [1,*]

[1] Department of Information and Communication Engineering, Chaoyang University of Technology, Taichung 413310, Taiwan
[2] Department of Electrical Engineering, National Taiwan University, Taipei 106216, Taiwan; csj@ntu.edu.tw
[3] Department of Electrical and Computer Engineering, University of Wisconsin-Madison, Madison, WI 53706, USA; yuhen.hu@wisc.edu
* Correspondence: azongtsai@cyut.edu.tw (W.-C.T.); mlchiang@cyut.edu.tw (M.-L.C.); Tel.: +886-4-2332-3000-7843 (W.-C.T.); +886-4-2332-3000-7241 (M.-L.C.)

Received: 5 September 2020; Accepted: 9 October 2020; Published: 13 October 2020

Abstract: A novel network-on-chip (NoC) integrated congestion control and flow control scheme, called Network-Cognitive Traffic Control (NCogn.TC), is proposed. This scheme is cognizant of the fluidity levels in on-chip router buffers and it uses this measurement to prioritize the forwarding of flits in the buffers. This preferential forwarding policy is based on the observation that flits with higher levels of fluidity are likely to arrive at their destinations faster, because they may require fewer routing steps. By giving higher priority to forward flits in high-fluidity buffers, scarce buffer resources may be freed-up sooner in order to relieve on-going traffic congestion. In this work, a buffer cognition monitor is developed to rapidly estimate the buffer fluidity level. An integrated congestion control and flow control algorithm is proposed based on the estimated buffer fluidity level. Tested with both synthetic traffic patterns as well as industry benchmark traffic patterns, significant performance enhancement has been observed when the proposed Network-Cognitive Traffic Control is compared against conventional traffic control algorithms that only monitor the buffer fill level.

Keywords: buffer fluidity; cognitive communication; traffic control; multi processors; networks on chip

1. Introduction

Network-on-Chip (NoC) is an emerging on-chip communication infrastructure [1] for on-chip communications in Multi-Processor System on Chip (MPSoC) [1] and Chip Multi-Processor (CMP) [2], platforms. With the increasing number of processor cores, the communication complexity over the NoC network increases accordingly. A cognitive communication system [3], which is cognizant of the rapid changing in-network traffic requirements promises great performance enhancement for on-chip communication [4], over conventional designs. For example, Melo et al. [5] analyzed the router behavior for detecting signal upset in order to minimize error propagations, Chang et al. [6] provided a contention prediction scheme for better adaptive routing path decisions, and Tang et al. [7] implemented a congestion avoidance method for NoC, based on the speculative reservation of network resource that was proposed by Jiang et al. [8]. Recently, Mehranzadeh et al. [9], designed a congestion-aware output selection strategy based on calculations of congestion levels of neighboring nodes, Giroudot et al. [10], realized a buffer-aware worst-case timing analysis of wormhole routers with different buffer sizes when consecutive-packet queuing occurs.

However, these cognitive network traffic control schemes address isolated issues in NoC communication. There is yet an integrated approach to consider routing, congestion control, and flow control jointly to enhance performance of NoC communication. In this work, we leverage a novel

fluidity concept proposed earlier [11,12], in order to develop a cognitive network traffic control scheme that features preferential data forwarding and corresponding integrated congestion control and flow control algorithms.

The key performance metrics of an NoC architecture include low packet delivery latency and high throughput rate. These metrics are critically impacted by network congestions due to resource contentions [13]. Therefore, contention resolution is key to avoid network congestions [7,9]. Assuming the routing scheme is both deadlock-free and livelock-free, contention resolution relies on the efficient utilization of available network resources to enhance overall performance [13].

Contentions can be alleviated by both congestion control and flow control because "Congestion control keeps the router network free of traffic jams; while flow control ensures that no transmission master overwhelms any of its counterpart slaves" [13]. Congestion control and flow control were both extensively used in NoCs. For congestion control, adaptive routing schemes [14,15], can be used to forward packets around the contentious regions. In NoCs, most congestion control schemes select profitable routing paths by monitoring some forms of buffer fill level [7,9,10,16–20]. NoC link-layer (node-to-node) flow control schemes [6,8,20–22], are developed to prevent buffers overfill between neighboring nodes. Moreover, transmission-layer (end-to-end) flow control schemes mitigate network traffic congestions by exchanging buffer fill messages (*credits*) between the transmission pairs [23–26]. However, credit traffic will consume a substantial amount of bandwidth [13]. Accordingly, most NoCs, such as [27], only apply the link-layer flow control to result in a chain of queued flits. Then the transmission flow control can be automatically obtained by the chain of queued flits between transmission ends [13]. Meanwhile, the network becomes congested. Some hybrid-layer flow control schemes [8,28–31] were proposed to achieve a better tradeoff between the link and transmission layers. These aforementioned traffic control schemes refer to the static buffer fill information to make decisions for flit and packet movements.

In this paper, we propose to leverage a buffer fluidity level to estimate dynamic network traffic loads and buffer utilization information. We observe that both the congestion control and flow control share a common objective of promoting efficient utilization of scarce network resources. However, in current NoC traffic control schemes, congestion control and flow control are managed independently and evaluated separately. Based on the buffer fluidity information, we propose a novel Network-Cognitive Traffic Control (NCogn.TC) scheme that facilitate both congestion control and flow control simultaneously. With the NCogn.TC, a buffer cognition monitor that constantly estimates the density of buffer operation types. Through an online learning process, real time traffic loads as well as accurate traffic predictions may be used to facilitate dynamic on-chip network traffic control. With this integrated approach, NCogn.TC achieves the objective of meeting the network traffic demands while optimizing network resource utilization.

In the remaining of this paper, related works are reviewed and motivations are presented in Section 2. The NoC performance metrics, including latency and throughput, are discussed in Section 3 with special focus on congested networks. In Section 4, the proposed NCogn.TC scheme is developed. Subsequently, experimental results are reported and discussed in Section 5. Further discussions are provided in Section 6. This is followed by the conclusion in Section 7.

2. Background and Related Works

Transmissions experiencing contentions along its routing paths result in a congestion tree [32], as illustrated in Figure 1, where lots of packets and flits blocked in the network lead to bad performance in both latency and throughput. Contentions can be alleviated by congestion and flow control schemes that have been extensively used in NoCs. In this section, we first present a review of different traffic control schemes. Next, we introduce an integrated scheme to satisfy the demanding NoC traffic control requirements.

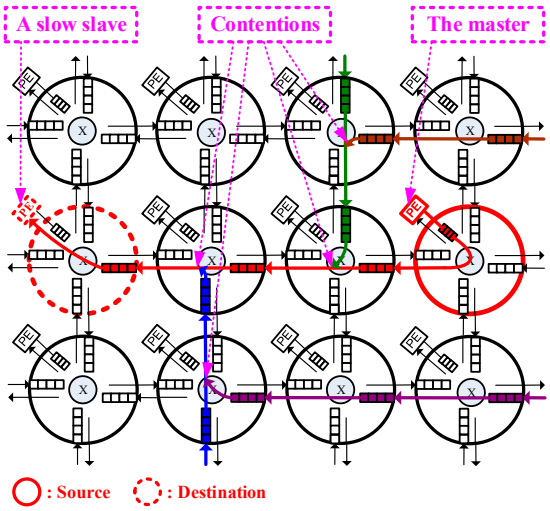

Figure 1. Congestion tree caused by contentions.

2.1. Congestion Controls in NoCs

2.1.1. Congestion Control, Using Resource Reservations

The congestion is caused by packets contending for the same resource (e.g., buffer, link). For this reason, in most schemes, congestion can be entirely controlled by scheduling packets globally, namely resource reservations. In NoCs, Virtual-Circuit Switching and Time Division Multiplexing used in Æthereal's contention-free routing [27], is a typical example. In such a way, network users must negotiate their requirements of resources, which requires insight into their communication behavior, and that takes time in the set-up and tear-down phases of reserving networking resources. Basically, in CMP NoCs with dynamic applications where flows are short-lived, changing frequently, or very variable in their demands; resource reservations are difficult to achieve [13]. Therefore, congestion control without resource reservation could be an alternative solution. Accordingly, adaptive routing schemes are commonly used in NoCs, as introduced in the next section.

2.1.2. Congestion Control, Without Resource Reservation

Adaptive routing algorithms can be decomposed into two functions: routing and selection. The routing function (e.g., Turn model [14] and Odd-Even [15]) supplies a set of admissible output paths. An output selected from this set is made by the selection function, such as [18], based on the status of output paths. Accordingly, adaptive routing algorithms are able to use alternative paths instead of waiting for busy channels. In adaptive routing algorithms, most selection functions are based on monitoring the buffer fill level in its neighboring nodes, such as Contention-look-ahead on-chip routing scheme [16], Dynamic Adaptive Deterministic (DyAD) switching [17], Neighbors-on-Path (NoP) selection [18], Local Congestion Avoidance [7], Mixed-criticality NoC [19], and Destination Intensity and Congestion-Aware (DICA) [9], all of which allow for each router to monitor the buffers of its nearest or two-hop neighboring routers in order to detect potential congestion earlier.

2.2. Flow Controls in NoCs

2.2.1. Flow Control, in Link Layer

A flow control scheme regulates the data transmission rates to prevent a sender overwhelming its receiver's data buffer. The transmission can be seen in the link layer (i.e., node-to-node) and in transmission layer (i.e., end-to-end). On one hand, in the link-layer flow control, a node transfers its buffer availability to the upstream nodes in order to prevent loss of packets. STALL/GO [21] and ACK/NACK [22] are typical examples that regulate the traffic flow locally by exchanging control

information between neighboring nodes. However, the node-to-node flow control relies on congestion backpressure that propagates the unavailability of buffers in the downstream nodes to the transmission sources. Consequently, when the congestion information reaches the transmission sources, packets have seriously congested the network. This phenomenon is distinct in wormhole switching and so-called a macro-pipeline of flits in [32] and consecutive-packet queuing in [10].

2.2.2. Flow Control, in Transmission Layer

On the other hand, transmission-layer (i.e., end-to-end) flow control conserves buffer spaces in the network by regulating the packet injection rate right at the transmission source [23–26]. However, the conventional credit-based (or token-based) end-to-end flow controls that are applied in macroscopic networks indeed require computation and communication overheads in both transmission master and slave. Besides, the end-to-end communication delay makes the credit received at the receiving end not able to timely reflect the traffic condition at the transmitting end. These overheads become more critical in an on-chip environment and lead to considerable bandwidth loading (31% as quoted in [33]) and unpredictable communication delay in exchanging traffic conditions between transmission pairs, as described in [13]. In our opinion, the implementation of overheads should be discussed globally from a system viewpoint to validate the practice.

2.2.3. Hybrid-Layer Flow Control Schemes

As discussed above, neither node-to-node nor end-to-end flow control schemes can satisfy the requirements of NoCs. For this reason, some hybrid-layer schemes were proposed. Global-Feedback-based Flow Control (GFFC) is a typical example [30], in which extra buffers and additional channels are required to buffer congestion-causing packets and feedback congestion information to the transmission masters. Besides, Lu et al. introduced a layered switching for NoC in [29], Shin and Daniel proposed a hybrid switching scheme in [28], Kim et al. [31] provided a credit network in parallel with data network to support hybrid flow control, and Jiang et al. [8] introduced a channel pressure concept, which refers to the largest amount of packets imposed on a channel of the NoC. All of the above-implemented routers are to reduce the number of packets or flits in a congested network. When compared with the above schemes, this paper implemented a buffer cognition monitor to reduce the length of macro-pipeline of flits caused by wormhole switching in the congestion tree [10,32] (i.e., tree saturation in [8]), as shown in Figure 2.

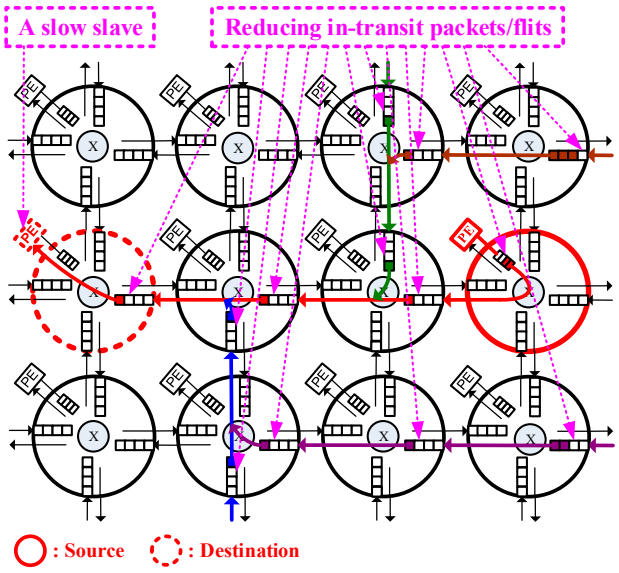

Figure 2. Congestion tree with less in-transit packets/flits.

2.3. Motivation of Integrating Congestion Control and Flow Control

Conventional implementations of congestion control and flow control take advantages of the well-studied results from the traditional macroscopic network and on-chip bus. However, for on-chip micro-networks (i.e., NoCs), some inherited architecture constraints and related software or hardware implementation overheads were not fully considered. For example:

2.3.1. Software-Based Implementation Occupying Processor Computation Resource

In common computer network implementations, the datalink layer and physical layer are designed by hardware circuits (e.g., Ethernet) operating in parallel. In contrast, the transport layer and network layer are software implemented (e.g., TCP/IP) upon processors. Therefore, congestion and flow controls usually require software efforts and pass control messages between communication pairs. Consequently, additional computation and communication overheads of processors cannot be avoided. However, in contrast to the macroscopic networks, NoCs are always equipped with a large number of micro-processors working in parallel and they require communication latency in nanoseconds. Thus, general hardware implementations for routine procedures (e.g., routing) are commonly used in NoCs to offload extra computation overheads of processors. In our opinion, the additional overheads of congestion and flow control schemes that are mainly implemented by software approaches should be carefully evaluated.

2.3.2. Hardware-Based Implementation Requiring Additional Sideband Wire Cost

As control signals used in on-chip bus [34], it is feasible to add sideband control signals among routers in an NoC to provide design flexibilities in congestion and flow control schemes. However, the following two major impacts were always overlooked in related studies. The first is the wiring issue: adding one extra control signal to each unidirectional link of a five-port router in an 8×8 mesh requires a total of 576 long wires to be used as interconnections among all of the routers. Moreover, timing and area of metal wires are getting much critical and costly than silicon gates in a chip, and this trend will continue with the shrinking process technology. The second is the reusability impact: in functionality, adding a new control signal to the router interface probably causes incompatible problems with the legacy design (i.e., processing element). Physically, the new control wires crossing the whole chip needs a new layout where more timing closure efforts are required.

2.3.3. Proposed Integrating Traffic Control

Accordingly, because the conventional congestion and flow controls were realized individually and based on distinct schemes. Therefore, it is difficult to jointly consider the required implementation overheads for software and hardware for NoCs. However, and obviously, congestion and flow controls have the same target, which is enhancing network performance by controlling traffic flows. Therefore, it is possible to implement a traffic control mechanism that can not only take the advantages of congestion and flow controls, but also achieve a better tradeoff between system performance and implementation overhead. Therefore, this paper proposes an integrated scheme: Network-Cognitive Traffic Control (NCogn.TC). For which, neither software computation overhead nor hardware sideband control signal is required. This means that means a significant advantage that NCogn.TC can be easily integrated into most existent NoC architectures, as introduced in Section 4 of design methodology and implementation.

3. Traffic Controls and Analyses

In this section, we analyze the NoC operation principles from the viewpoint of buffer fluidity and such realization will help to refine the implementation of the proposed NoC traffic control. For example, we assume that an NoC fabric consists of buffers in routers. Each buffer is a ring type of four flits long and operates in a FIFO (First In, First Out) fashion, as shown in Figure 3. Additionally, the NoC

uses common settings of variable packet length, wormhole switching, and a link bandwidth of one flit per cycle.

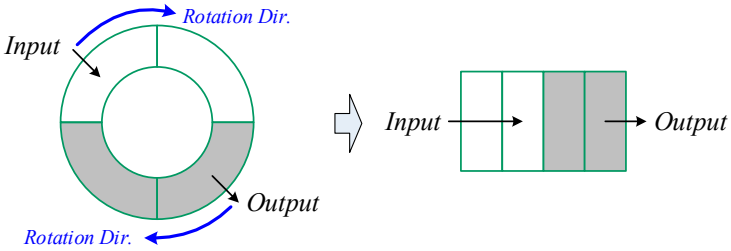

Figure 3. Ring buffer operated in a FIFO fashion.

3.1. Performance Analysis for Latency

Because one of the main NoC objectives is lowering the average packet latency, we study the sources of latency that hinder buffers from being fluid. Latency (l) can be separated into three different factors, as:

$$l = p_f + h + s \tag{1}$$

where the length of a packet in a number of flits (p_f) causes latency from receiving the head flit to the tail flit at the destination node. The hops (h) occur when flits are moving toward the next nodes (i.e., the buffer is in a fluid state). The length of packets and total number of hops are fixed and cannot be reduced to optimize the latency performance. On the contrary, stalls (s) are states when flits are waiting for grants to move ahead (i.e., the buffer is in an un-fluid state). Flit stalls are caused by resource contentions and should be considered as a key factor for incurring excess latency. Accordingly, Figure 4 shows a simplistic view of streams in six buffers; each arc represents a transition that the flit will take on next cycle. A flit, not being able to move out of the buffer, may be caused by a stall of buffer full (s_b), a stall of link unavailable (s_l), or a stall of waiting in the queue (s_w), where the flit is not at the front of a buffer. Flit stalls can be represented as:

$$s = s_b + s_l + s_w \tag{2}$$

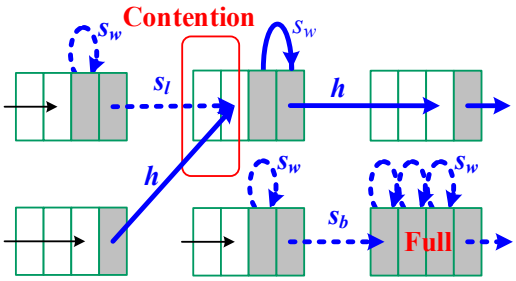

Figure 4. Latency components.

Stalls of s_b and s_l can be reduced by adaptive routing coupling with congestion control. It is notable that s_w is artificially created by buffer size and it seems to be unavoidable before. In this paper, we will demonstrate that s_w can be reduced by our proposed traffic control scheme in the following sections.

3.2. Performance Analysis for Throughput

The maximum throughput of an NoC can be analogous to the maximum-flow problem with multiple sources and sinks in graph algorithms in order to compute the greatest data transmission rate among every communication pairs without violating any capacity constraints. Given a flow network graph $G = (V, E)$, each edge $(u, v) \in E$, and vertices $u, v \in V$. A cut c (S, T) of flows f (S, T) in a network G

is a partition of V into S and T ($V = S + T$), such that $s \in S$ and $t \in T$. A minimum cut of a flow network is a cut whose capacity is minimum overall cuts of the network. The max-flow-min-cut theorem tells that a flow f (s, t) is maximum if and only if its residual network contains no augmenting path (i.e., c (s, t) is minimal). The throughput of a flow network G can be represented as:

$$t(G) = f(S,T) = \sum_{s \in S} \sum_{t \in T} f(s,t) \leq \sum_{s \in S} \sum_{t \in T} c(s,t) = c(S,T) \tag{3}$$

Accordingly, the maximal throughput of a flow network G equals the minimum cut:

$$Max\ t(G) = Max\ f(S,T) = Min\ c(S,T) \tag{4}$$

It is well known that an adaptive routing algorithm (e.g., Odd–Even) can increase the maximal throughput than a static routing algorithm (e.g., XY). The reason is that due to the additional routing paths provided by the adaptive routing ability, the minimum cut of the flow network G with Odd–Even (G_{OE}) is probably greater than the minimum cut of the flow network G with XY (G_{XY}), as:

$$Max\ t(G_{OE}) = Min\ c(S,T)_{OE} \geq Min\ c(S,T)_{XY} = Max\ t(G_{XY}) \tag{5}$$

Therefore, in this paper, we propose a new Buffer-Cognition-Level Congestion Control (BCogL.CC) scheme in order to thoroughly utilize the possible transmission paths explored by using adaptive routing algorithm. Besides, in an NoC, the rate of traffic flows is probably a kind of distributions (e.g., Poisson). Accordingly, by enlarging the size of buffers to spread the burst traffic evenly, then the channel utilization can be increased, and the throughput performance can also be enhanced.

3.3. Performance Analysis for Congested Network

In a network system, referring to the queueing model in [35], utilization (ρ) of a server equals the proportion of packet arrival rate (λ) to packet service rate (μ), which is $\rho = \lambda/\mu$ and the queue length (L) of the server equals $\rho/(1-\rho)$. If ρ (server utilization) gets closer to 1 (i.e., 100% utilization), then L (queue length) increases rapidly. Accordingly, long queues always exist in a network with busy servers. To put it more concretely, in a realistic system, the queue length is actually limited by the adopted buffer size. Therefore, long queues exist in the queueing model represents buffers are usually full in an NoC. Furthermore, in order to calculate the average packet waiting time (W) in a highly-loaded buffer, we refer to the Little's Law: $L = \lambda W$ of queueing theorem in [35]. Because λ is a constant value of packet arrival rate, a relation between packet waiting time and the buffer size can be deduced as:

$$W \propto L \approx the\ buffer\ size \tag{6}$$

According to Equation (6), the packet waiting time can be reduced by using small buffers. However, the sizes of buffers are fixed in an NoC; besides, smaller buffers could lead to lower utilization of servers and decrease the maximal throughput saturation point, which we will discuss in the next section. Therefore, we propose a new Buffer-Cognition-Level Flow Control (BCogL.FC) scheme. In the section of experimental results, we will demonstrate that BCogL.FC operates with the ring buffer, as illustrated in Figure 3, can effectively decrease latency as Equation (1) without sacrificing throughput performance as Equation (5) in most traffic types and loads.

3.4. Performance Measurement for Traffic Control

By monitoring the buffer status, the collected statistical information can be used to represent the network performance from a standpoint of resource utilization. In minimal routing, latencies that are caused by transferring a packet in the network architecture include basic transmission hops (h) and

additional stall periods (s). The total number of h and s represents the cycle time that buffers are in use, and the percentage of h of this total number reflects the efficiency of buffer operations:

$$\frac{\sum\limits_{b \in \mathbf{B}} h}{\sum\limits_{b \in \mathbf{B}} (h+s)} = B_{efficiency}, \tag{7}$$

where b is a buffer in the set of all network buffers B. Additionally, the buffer usage can be represented as the ratio of the active cycles of buffers to the total simulation time ($SimTime$) of buffers:

$$\frac{\sum\limits_{b \in \mathbf{B}} (h+s)}{|\mathbf{B}| \times SimTime} = B_{usage} \tag{8}$$

The Buffer Efficiency ($B_{efficiency}$) is relevant to the communication efficiency and the Buffer Usage (B_{usage}) reflects the resource utilization. We use these two performance parameters to analyze our proposed fluidity-aware, Network-Cognitive Traffic Control (NCogn.TC) scheme in the experiment section.

3.5. Performance Interactions between Latency and Throughput

Low latency must lead to high throughput. This could be a fallacy, even in minimal routing. As mentioned above, large-size buffer and high routing adaptivity profit the throughput performance; nevertheless, these could be disadvantageous for latency, especially in a congested network path where the packet arrival rate (λ) is close to the packet service rate (μ). For example, in Figure 5b, there are four additional buffer slots when compared to Figure 5a. In a heavy traffic-load network, a larger buffer could buffer more packets referring to Equation (6). Accordingly, when compared with Figure 5a, the additional buffers in Figure 5b cause two more buffered flits, which lead to two additional waiting time (s_w) in Equation (2) for each flit in the network. Next, in Figure 5c, there are more routing paths than in Figure 5a, but the throughput cannot be improved, since the minimum cut for the flow is the same as that in Figure 5a by Equation (4). Moreover, as shown in Figure 5c, when merging flows from different routing paths, contentions could happen and lead to additional waiting time (s_w and s_l) in Equation (2). A network might perform inconsistently in merits of throughput and latency, as mentioned above. In the section of experimental result, we will show that this phenomenon is implicit for random traffic types but distinct in regular traffic scenarios. Furthermore, we will demonstrate that the negative latency effect will be mitigated by using our proposed design methodology, as introduced in the following sections.

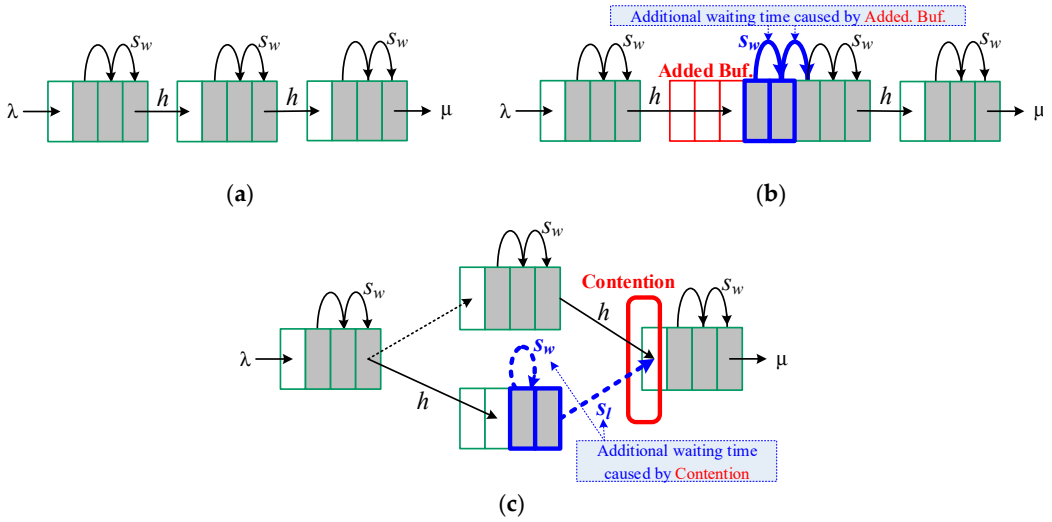

Figure 5. Latency analyses in cases of (**a**) a single path with a buffer size of twelve, (**b**) a single path with a buffer size of sixteen, and (**c**) two alternative paths with a buffer size of sixteen. The packet arrival rate is close to the packet service rate in a congested network. The throughputs in (**a**,**b**), and (**c**) are similar, but the latency in (**a**) is less than the latencies in (**b**,**c**).

4. Methodology and Implementation

In this section, we introduce the methodology foundations and implementation details that are based on our previous studies and researches [11,12,36] for the proposed NCogn.TC scheme, as follows.

4.1. Methodology Foundations

4.1.1. Congestion Avoidance and Congestion Relief

Congestion Avoidance (CA) [7,9,11,16–18] is a common congestion control method that depends on downstream nodes to report its buffer status to the upstream nodes. Accordingly, in adaptive routing, a node can select to output packets to a neighboring node that is expected to be in less congestions when compared with other alternative selections, as shown in the scenario of Figure 6a. In contrast to avoiding congestions by selecting a profitable output port, Congestion Relief (CR) [11,12,37] is another congestion control scheme used for the input port selection, which provided a contention-aware input port selection algorithm for a node to receive the packets in the upstream node that is expected to contain more traffic flows than nodes connecting to other input ports, as shown in Figure 6b.

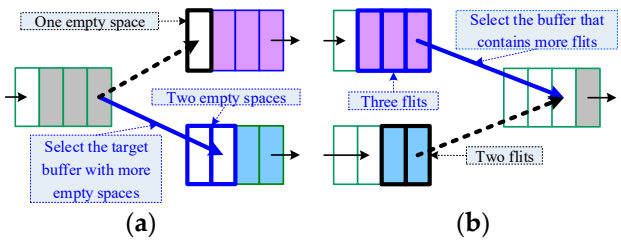

Figure 6. (**a**) Congestion avoidance selects a target buffer with more empty space and (**b**) congestion relief selects a source buffer with more flits.

4.1.2. Buffer Fluidity Concept

Conventional congestion control schemes depend on some forms of buffer fill level information to dynamically select routing paths [7,9,16–19]. Differently, we applied the proposed concept of buffer fluidity level in [11]. When compared with the buffer fill level that indicates the queue length of a

buffer, the buffer fluidity level reflects the operation status of a buffer, such as the fluid and un-fluid (or stalled) states that are shown in Figure 7.

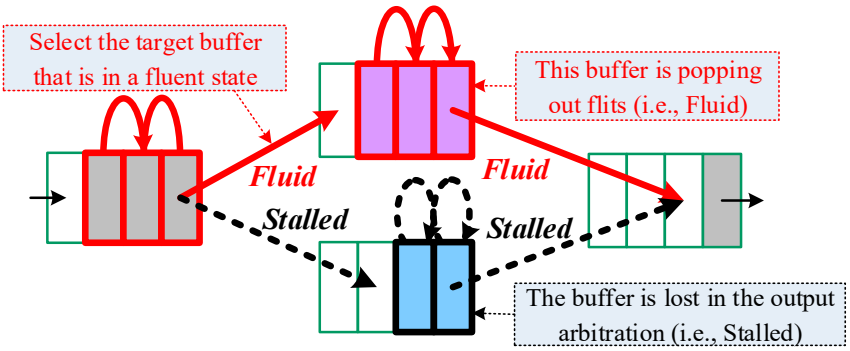

Figure 7. Path selection based on the buffer fluidity state.

To put it another way, the buffer fill level reflects the buffer's current state status and our proposed buffer fluidity level tries to predict the buffer's next state status. That is, a buffer in a fluid state is expected to be continuously fluid, because a buffer always keeps in the same state for a period of time. For example, in Figure 7, rather than selecting a routing path to the neighboring node with less buffer filled, the upstream node outputs its packets to another node that is in a fluid state. If the prediction is correct, then a better performance will be achieved. Actually, the prediction could be more precise when referring to more historical operation data rather than the current buffer state status. Based on the buffer fluidity concept, a buffer cognition monitor is implemented in order to realize the Network-Cognitive Traffic Control (NCogn.TC), as described in the following sections.

4.1.3. In-Transit Packets Reduction

Shin and Daniel proposed a hybrid switching scheme [28], which dynamically adopts wormhole or virtual-cut-through switching to obtain a better performance balance between latency and throughput. Accordingly, we observed that, in a light traffic-load network, wormhole switching performs better than virtual-cut-through in throughput performance. On the contrary, in a heavy traffic-load network, the latency can be reduced by virtual-cut-through, since it causes less in-transit packets (two packets) than wormhole does (three packets). Figure 8a,b show the scenarios.

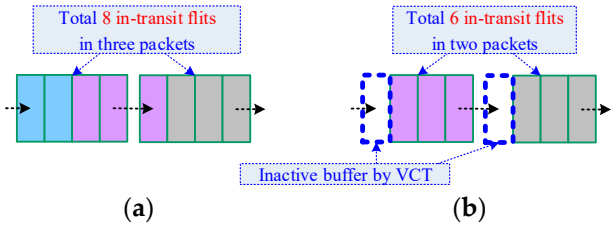

Figure 8. (**a**) Wormhole switching and (**b**) virtual-cut-through switching.

In other words, wormhole performs link-layer flow control on the flit level; virtual-cut-through does this on the packet level. Consequently, these two switching methods deliver diverse latency performances between light and heavy traffic loads. Besides, Lu et al. [29] proposed a layered switching that implements wormhole switching on top of virtual-cut-through. It realizes flow control on a new group that comprises several flits. Instead of queuing packets in unit of one flit, the buffers are allocated group by group, causing fewer packet transits in the network as well as [28], does.

4.1.4. Stall and Go Link-Layer Flow Control

In NoC link-layer flow controls, in addition to the application of ACK/NACK to handle link errors between nodes as Xpipes [22], STALL/GO (ON/OFF) is also commonly used in errorless link environments [21]. Besides, STALL/GO also exhibits the equivalent behavior as a one credit-based link-layer flow control operating in a pipelined fashion, as shown in Figure 9. Referring to the description in [13,21], STALL/GO is a low-overhead, low-cost scheme and it assumes reliable flit delivery. It uses a control wire going backward to signal that either a condition of buffers filled ("STALL") or buffers free ("GO"). STALL/GO can be implemented with few buffers and the minimal control logic, as shown in Figure 9. Therefore, we selected to implement an enhanced flow control based on the STALL/GO.

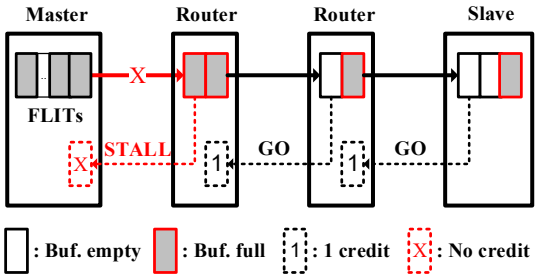

Figure 9. STALL/GO protocol implementation.

4.2. Implementation Details

4.2.1. Buffer Cognition Monitor (BCogM)

Referring to the primitive fluidity concept [11] and meter [12], in this section, we will explain how to apply both concepts of buffer fill and buffer fluidity to realize a Buffer Cognition Monitor (BCogM) in order to reflect the real-time traffic condition in multi-levels. Two six-state (L0 to L5) State Machines (SMs) as shown in Figure 10 are used to demonstrate the dynamic traffic control features. Definitions of our proposed Buffer Fluidity Level (BFluL) and the conventional Buffer Fill Level (BFilL) are respectively given in Tables 1 and 2.

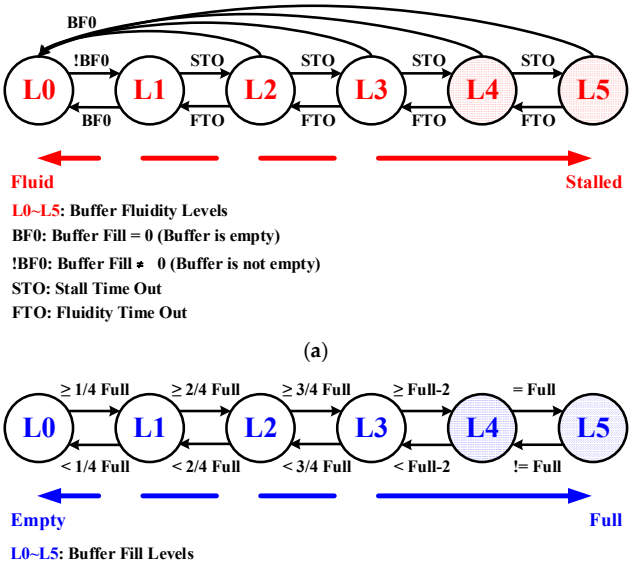

Figure 10. A Buffer Cognition Monitor (BCogM) consisted of State Machines (SMs) monitoring (**a**) the Buffer Fluidity Level (BFluL) and (**b**) the Buffer Fill Level (BFilL).

Table 1. Buffer Fluidity Level (BFluL).

Level	Definition
L0	Empty (buffer is empty)
L1	Fluid (buffer is not empty and can output flit rapidly)
L2	Fair-fluid (buffer is not empty and can output flit frequently)
L3	Less-fluid (buffer is not empty and can output flit in a short period)
L4	Un-fluid (buffer is not empty but cannot output flit in a short period)
L5	Stalled (buffer is not empty but cannot output flit in a long period)

Table 2. Buffer Fill Level (BFilL).

Level	Definition
L0	Near-empty (filled flits < 1/4 buffer size)
L1	1/4-full (filled flits ≥ 1/4 buffer size)
L2	2/4-full (filled flits ≥ 2/4 buffer size)
L3	3/4-full (filled flits ≥ 3/4 buffer size)
L4	Almost-full (filled flits ≥ buffer size−2)
L5	Full (filled flits = buffer size)

In order to gather the buffer fluidity level information, in Figure 10a, the default state of the state machine is L0, which is valid when the buffer is empty. L1 to L5 represent five buffer fluidity levels from fluid to un-fluid. The state transfer event, STO (Stall Time-Out), indicates that the buffer is not empty and cannot transfer any flit to other nodes for a pre-defined period (i.e., buffer output is stalled). The other state transfer event, FTO (Fluidity Time-Out), indicates that there is a certain number of flits being moved out from the buffer (i.e., buffer output is fluid). Different timeout conditions of STO and FTO determine the sensitivity levels of BCogM. For example, a smaller timeout value causes more state transitions due to STO or FTO being frequently valid. Therefore, the timeout values should be configurable to fit different flow conditions in different traffic types and buffer locations. As a whole, the state in Figure 10a trends moving to the un-fluid state (L5) when the buffer cannot forward any flit to the next nodes. On the contrary, the state presented in Figure 10a probably keeps in the fluid state (L1) when the FIFO buffer push and pop operate alternately. In short, by constantly calculating the density of operations in the buffer, the buffer fluidity concept can be realized to represent not only a real-time traffic condition, but also a precise traffic prediction. Additionally, Figure 10b presents another common buffer fill level state machine for comparisons with our proposed one. In the following sections, we will introduce that both levels of fluidity and fill will be applied in our proposed NCogn.TC scheme.

4.2.2. Buffer-Cognition-Level Congestion Control (BCogL.CC)

In conventional congestion controls, the adaptive routing algorithm trends to select an alternative routing path, in which the buffer is with a lower buffer fill level. Using conventional buffer fill level in congestion control is good, but not sufficient, because it reveals the static status (i.e., buffer fill level), but misses the dynamic information (i.e., buffer fluidity level) of buffers. To put the matter simply, if there were two target paths, whose buffers with an identical buffer fill level (e.g., L3 in Figure 10b), but one is fluid (e.g., L1 in Figure 10a) and the other is stalled (e.g., L5 in Figure 10a), in our selection, choosing a path with the fluid buffer could deliver a better performance.

From the above consideration, we propose a new buffer level and named it buffer cognition level (BCogL), which jointly considers the buffer fill level (BFilL) and buffer fluidity level (BFluL). The basic concept is that for two buffers with the same buffer fill level, an un-fluid buffer should have a higher buffer level than its buffer fill level. For example, as Algorithm 1 shows, the BCogL equals the BFilL if the BFluL is in a fluid state. Otherwise, the BCogL is assigned a higher level than the BFilL when the buffer is in an un-fluid state. Even though under such a simple level adjustment, the performance can be effectively increased by applying the proposed BCogL in both of Congestion Avoidance (CA) and Congestion Relief (CR) to be a hybrid BCogL Congestion Control (BCogL.CC) scheme, as demonstrated in the section of experimental results.

Algorithm 1 Buffer-Cognition-Level Congestion Control (BCogL.CC)

```
Input: BFluL //Buffer Fluidity Level
Input: BFilL //Buffer Fill Level
Output: BCogL //Buffer Cognition Level
Begin
    if (L0 ≤ BFluL ≤ L2) //buffer is in a fluid state
        BCogL = BFilL //assign the buffer cognition level as the original buffer fill level
    else //L3 ≤ BFluL ≤ L5) //buffer is in an un-fluid state
        BCogL = BFilL + 1 //assign the buffer cognition level to a higher buffer fill level
End
```

4.2.3. Buffer-Cognition-Level Flow Control (BCogL.FC)

The Buffer Cognition Monitor (BCogM) presented in Figure 10 can be used to enhance NoC performance, not only in congestion control but also in flow control. Common flow control methods regulate packets or flits injecting into the network after routing paths had been selected by the adopted congestion control schemes. Furthermore, our proposed BCogL.FC uses the in-transit packets concept to enhance latency performance when the network is under heavy traffic loads. Besides, instead of mixing both wormhole and virtual-cut-through switchings in a router as [28,29], we give the common wormhole router an additional flow control ability to suppress injecting packets or flits into the network when the BCogM reflects that the traffic flow is un-fluid, rather than that buffer is full. Therefore, in a congested network, the length of macro-pipeline of flits can be reduced; in consequence, the congestion backpressure is rapidly propagated by a shorter macro-pipeline of flits. This scenario is illustrated as a congestion tree with less filled flits in Figure 2. Additionally, in order to prevent the throughput performance loss under light and middle traffic loads, the traditional flow control scheme must operate in a smarter way, as Algorithm 2 discussed below.

Algorithm 2 discloses the conditions where the STALL signal is asserted or de-asserted (GO) by the BCogL.FC mechanism. In the conventional STALL/GO flow control, STALL is only asserted when the buffer is full to prevent loss of packets. Moreover, BCogL.FC dynamically asserts STALL according to the real traffic conditions. For example, to maintain high performance under light traffic loads, BCogL.FC asserts STALL only when the buffer is near full. This condition does rarely exist, since the input flits are probably forwarded to next nodes if there were less congestions. On the contrary, as the un-fluid level rises, BCogL.FC increases the probability of asserting STALL to disallow flits to input its buffer, even though there exists free space to reduce the number of in-transit packets. Consequently, latency performance can be enhanced.

Algorithm 2 Buffer-Cognition-Level Flow Control (BCogL.FC)

```
Input: BFluL //Buffer Fluidity Level
Input: BFilL //Buffer Fill Level
Output: STALL //legacy buffer STALL indication
Begin
    if (BFilL = L5) //buffer is full (L5)
        STALL = TRUE //assert STALL signal when buffer is full (as STALL/GO)
    else if (BFilL = L4 & BFluL = L1) //buffer is almost-full(L4) and flow is fluid(L1)
        STALL = TRUE //pre-assert STALL signal when buffer is almost-full and flow is fluid
    else if (BFilL = L3 & BFluL = L2) //buffer is 3/4-full(L3) and flow is fair-fluid(L2)
        STALL = TRUE //pre-assert STALL signal when buffer is 3/4-full and flow is fair-fluid
    else if (BFilL = L2 & BFluL = L3) //buffer is 2/4-full(L2) and flow is less-fluid(L3)
        STALL = TRUE //pre-assert STALL signal when buffer is 2/4-full and flow is less-fluid
    else if (BFilL = L1 & BFluL = L4) //buffer is 1/4-full(L1) and flow is un-fluid(L4)
        STALL = TRUE //pre-assert STALL signal when buffer is 1/4-full and flow is un-fluid
    else if (BFilL = L0 & BFluL = L5) //buffer is near-empty(L0) and flow is stalled(L5)
        STALL = TRUE //pre-assert STALL signal when buffer is near-empty and flow is stalled
    else //valid always when buffer is near-empty and flow is fluid
        STALL = FALSE //de-assert STALL signal as the GO condition in STALL/GO
End
```

4.2.4. Network-Cognitive Traffic Control (NCogn.TC)

Here, an integrated scheme, the Network-Cognitive Traffic Control (NCogn.TC), is presented. NCogn.TC is an incorporation of the Buffer Cognition Monitor (BCogM) in Figure 10, the Buffer-Cognition-Level Congestion Control (BCogL.CC) in Algorithm 1, and the Buffer-Cognition-Level Flow Control (BCogL.FC) in Algorithm 2. NCogn.TC was implemented as a hardware module depicted in Figure 11a and incorporated in a typical router, as shown in Figures 11b and 12.

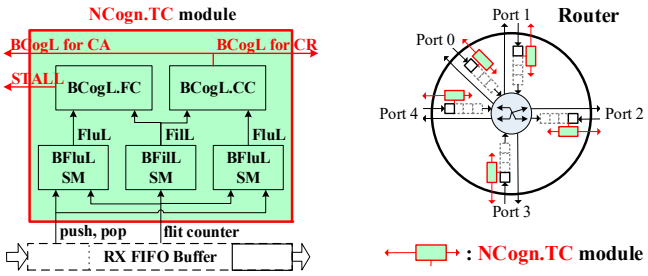

Figure 11. (**a**) NCogn.TC module (**b**) incorporated in a typical five-port router.

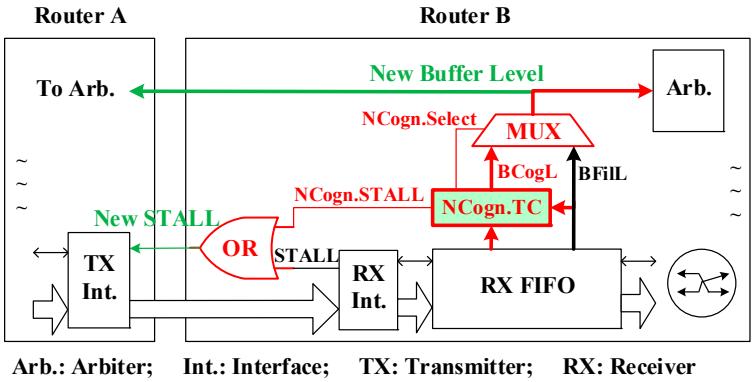

Figure 12. Router incorporated with the Network-Cognitive Traffic Control (NCogn.TC) module using simple OR and MUX logic gates.

4.3. Distinctions and Contributions

The major differences among this paper and our previous works of CARRS (Congestion Avoidance and Relief Routing Scheme) [11], BiNoC-FM (Bi-directional NoC Fluidity Meter) [12], and other researches are discussed as follows and are listed in Table 3 (denoted as footers 1-8).

1. First, CARRS [11] defines three levels ("Inactive", "Fluid", and "Non-fluid") of coarse-grained buffer fluidity. Comparably, this paper provides six levels ("Empty", "Fluid", "Fair-fluid", "Less-fluid", "Un-fluid", "Stalled") of fine-grained buffer fluidity (cf. Section 4.2.1). The three-level design of CARRS [11] causes a problem that when a buffer (with size of n flits) contains only one flit and cannot forward the flit to the downstream nodes, the buffer changes its state from "Fluid" to "Non-fluid"; consequently, the buffer cannot accommodate any flit for upstream node, even if there are still n-1 flits buffer space, the possibly impermanent virtual congestion condition will propagating to other upstream nodes leading to actual congestion conditions in the network. Accordingly, this paper proposed a novel buffer cognition level (BCogL; cf. Section 4.2.2 and Algorithm 1), which jointly considers the buffer fill level (BFilL) and buffer fluidity level (BFluL) for properly handling the congestion control.
2. Second, CARRS [11] applies the primitive wormhole switching as its flow control method (at link-layer, between neighboring nodes). However, as previously discussed, CARRS [11] could waste useable buffer space due to the simple three-level fluidity state, and lead to a reducing length of macro-pipeline of flits [32]. By contrast, this paper implemented the Buffer Cognition Monitor (BCogM; cf. Section 4.2.1), which is applied by the Buffer-Cognition-Level Congestion Control (BCogL.CC; cf. Section 4.2.2) to reduce the length of macro-pipeline of flits when detecting possible congestions that existed between transmission-layer source and destination (i.e., end-to-end). Especially, the proposed Buffer-Cognition-Level Flow Control (BCogL.FC; cf. Section 4.2.3) can further select to buffer more flits to enhance the global throughput performance when congestions are encountered locally between neighboring nodes (i.e., node-to-node).
3. Third, CARRS [11] evaluates the network performance using packet latency, but without throughput metrics, for the three synthetic traffic patterns (uniform, hotspot, and transpose). Large-size buffer and high routing adaptivity can profit the throughput performance; nevertheless, these could be disadvantageous for latency, as discussed in Section 3.5. Similarly, some studies could enhance the latency performance by such packet entry control at the transmission source end and just calculate the transmission time (i.e., latency) of packets travelling in the network. In our opinion, a general purpose network shall be balanced in both performance metrics of latency and throughput for different traffic types. Accordingly, this paper provides experimental performance analyses of both latency and throughput, for either synthetic or real traffic patterns, as demonstrated in Section 5, in order to validate the outstanding performance merits by utilized the proposed Network-Cognitive Traffic Control (NCogn.TC) scheme.
4. Besides, BiNoC-FM [12] implements a Fluidity Meter (FM) to provide real-time estimates of bandwidth utilization of Virtual Channel (VC) buffers in a special Bi-directional NoC (BiNoC) router in order to provide a better Quality-of-Service (QoS) service. The provided fluidity meter is similar to CARRS [11] and relatively simpler than the buffer monitor implemented in this paper.
5. PCAR [6] provides a path congestion-aware adaptive routing (PCAR) method, which uses a rate of change in the buffer level to predict possible contentions for further performance improvement. The fluidity concept resembles that of CARRS [11], thus achieving similar performance. Especially, PCAR [6] provides implementation cost analyses for its router area and power dissipation.
6. TMPL [7] suggests to avoid the global congestion in every local region by keeping routing pressure of every local region minimum, and conducts that when local region size is 5×5, the optimal routing performance of large size network could be achieved. Where, the routing pressure refers to the amount of packets imposed on a channel as the conventional buffer fill level (i.e., BFilL) defined in this paper.

7. DICA [9] is a Destination Intensity and Congestion-Aware (DICA) output selection strategy that is based on the proposed congestion level. It is defined as the number of occupying input buffer slots of adjacent nodes, especially the calculation can comprise buffers of two hops away neighboring nodes. The experiments were evaluated compressively including both synthetic and real traffic scenarios with three performance metrics of latency, throughput, and energy consumption. However, the applied congestion level remains as the conventional buffer fill level (i.e., BFilL).
8. BATA [10] addresses the problem of worst-case timing analysis of wormhole routers with different buffer sizes when consecutive-packet queuing occurs. The provided buffer-aware method predicts that delay bound increases when the buffer size increases, which is similar to the issue of macro-pipeline of flits [32] and the corresponding flow control scheme we discussed and realized in the paper. Accordingly, BATA [10] can be a proof that can validate the efficacy of our proposed Buffer-Cognition-Level Flow Control (BCogL.FC) scheme.

Table 3. Comparisons among the proposed NCogn.TC and other researches.

Scheme/Research		NCogn.TC	CARRS [11]	BiNoC-FM [12]	PCAR [6]	TMPL [7]	DICA [9]	BATA [10]
Routing Algorithm	XY (or DOR)							•
	OE	•	•	•	•	•	•	
Buffer Monitor	BFilL					• [6]	• [7]	• [8]
	BFluL		• [1,2]	• [4]	• [5]			
	BCogL (Fil + Flu)	• [1,2]						
Congestion Control	CA				•	•	•	•
	CR			•				
	CC (CA + CR)	• [1]	• [1]					
Flow Control	LL		• [2]	•	•	•	•	•
	TL							
	CL (LL + TL)	• [2]						• [8]
Performance Analyses	Latency	• [3]	• [3]	•	•	•	• [7]	•
	Throughput	• [3]			•		• [7]	
	QoS			• [4]				
	Cost				• [5]	•	• [7]	
Simulation Traffic	Synthetic	• [3]	• [3]	•	•	•	• [7]	•
	Real	• [3]					• [7]	

Note 1: Table footers 1-8 are denoted as the paragraph numbers in this section. Note 2: Full words of the abbreviations in the table are denoted as follows: DOR (Dimension Order Routing), OE (Odd-Even), BFilL (Buffer-Fill-Level), BFluL (Buffer-Fluidity-Level), BCogL (Buffer-Cognition-Level; a combination of BFilL and BFluL), CA (Congestion-Avoidance), CR (Congestion-Relief), CC (Congestion-Control; a combination of CA and CR), LL (Link-Layer; i.e., Node-to-Node), TL (Transmission-Layer; i.e., End-to-End), CL (Cognition-Layer; a combination of Link and Transmission Layers), and QoS (Quality-of-Service), Cost (including area and/or power).

5. Experimental Results

In the experiments, we used the state-of-the-art Odd-Even turn-model [15] as the fundamental adaptive routing algorithm (OE-Baseline). Additionally, a basic XY routing algorithm (XY-Baseline) is also included for comparisons. As Figure 13 shows, in order to validate that additional performance benefits could come from applying the fluidity concept in the NoC traffic controls, first we provided the performance for XY and Odd-Even (OE). Next, OE equipped with the Buffer Fill Level and the hybrid Congestion Control (OE-BFilL.CC) are compared with OE equipped with the Buffer Cognition level and the hybrid Congestion Control (OE-BCogL.CC), in order to validate the performance differences between applying the conventional BFilL and our proposed BCogL. Finally, an integrated traffic control scheme, coupling OE routing algorithm, BCogL Congestion Control (BCogL.CC), and BCogL Flow Control (BCogL.FC), namely OE based, Network-Cognitive Traffic Control (OE-NCogn.TC) is displayed.

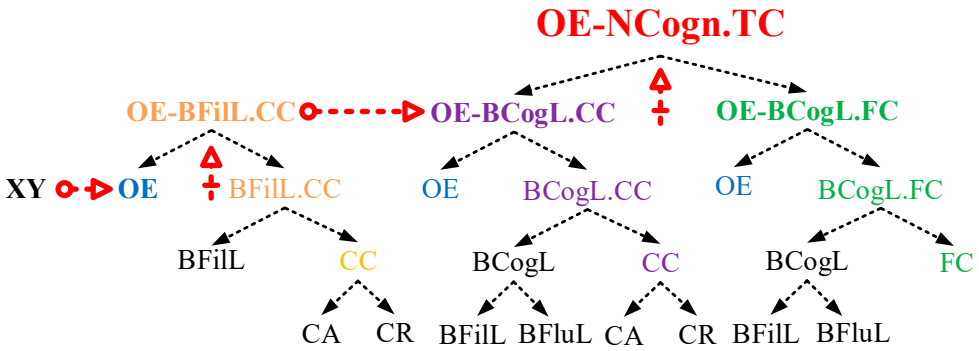

Figure 13. Taxonomy of the couplings of routing algorithms and traffic control schemes evaluated in experiments.

5.1. Cycle Accurate Evaluation

Comprehensive simulations were run in Register Transfer Level (RTL) while using Cadence NC-Verilog. Performance metrics were carried out on an 8×8 mesh network. Each link's bandwidth is set to one flit (32 bits) per cycle. Four cycles are required for switching a header flit to an output port in a pipelining fashion. In order to use a practical buffer size considering both cost and performance, we assigned each input-port buffer in common 32 flits (i.e., 1024 bits) to accommodate a feasible number of multiple packets of burst traffic [38]; besides, a packet may be queued in different nodes due to wormhole switching.

To adapt to various traffic flow conditions at different buffer locations of the NoC router (cf. Figure 11b), the timeout value of STO of the buffer Cognition Monitor (BCogM) was set to a quarter (×1/4) or a quadruple (×4) number of the buffer size (32 flits), depending on the buffer connected to a neighboring router (at port 1/2/3/4) or a processing element (at port 0). Accordingly, the STO value was set to 8 ($32 \times 1/4$) for buffers of port 1/2/3/4, which is much smaller than that of 128 (32×4) for the buffer of port 0. The reason is that, in common simulations, as compared with flits in the buffer of port 0 that can always be flushed by a processing element; however, flits in buffers of port 1/2/3/4 probably cannot be fetched by neighboring routers that are congested in a busy network. Therefore, assigning a smaller value of STO could cause a faster state transition (cf. Figure 10a) to react on the state of buffer fluidity level from fluid to stall. Besides, the other timeout value of FTO was assigned as one to rapidly transfer its state to the fluid when detecting any flit moved out. In general, to validate that, the proposed buffer fluidity level can be applied for performance enhancement, thus the values of STO and FTO were just selected for evaluations and can be optimized for further studies.

5.2. Experiments with Synthetic Traffics

In NoCs, traffic conditions are irregular and unpredictable. Accordingly, we tried to analyze the traffic control effects for such traffic scenarios. Three common traffic patterns, namely uniform, hotspot, and transpose used in [15], were considered in our experiments. In uniform traffic, a node transmits a packet to any other node with equal probability. In hotspot traffic, uniform traffic is applied, but 10% of packets change their destination to one of the following four selected nodes ((7, 2), (7, 3), (7, 4), and (7, 5)) with equal probability. In transpose traffic, a node at (i, j) always sends packets to a node at (j, i). Especially, the size of packets was randomly distributed between four and 32 flits in order to confirm that the packet size should be adjustable for really communication needs. Furthermore, all of the performance metrics are averaged over 60,000 packets after a warm-up session of 30,000 arrival packets at each of the forty packet injection rates.

5.2.1. Uniform Traffic

Uniform traffic is a common pattern used in most NoC studies. In Figure 14a, we observed that the XY-Baseline performs the best when compared to other algorithms. Identical results are shown

in [15]. The reason is that, under uniform traffic, XY-Baseline happens to spread traffic much evenly across the paths in a mesh. However, except for XY-Baseline, OE-NCogn.TC performs better than other traffic control schemes in all aspects. Referring to Table 4 and Figure 15, OE-NCogn.TC averagely improves OE-Baseline 51.49% in latency, 13.14% in throughput, 16.79% in buffer efficiency, and 5.67% in buffer usage.

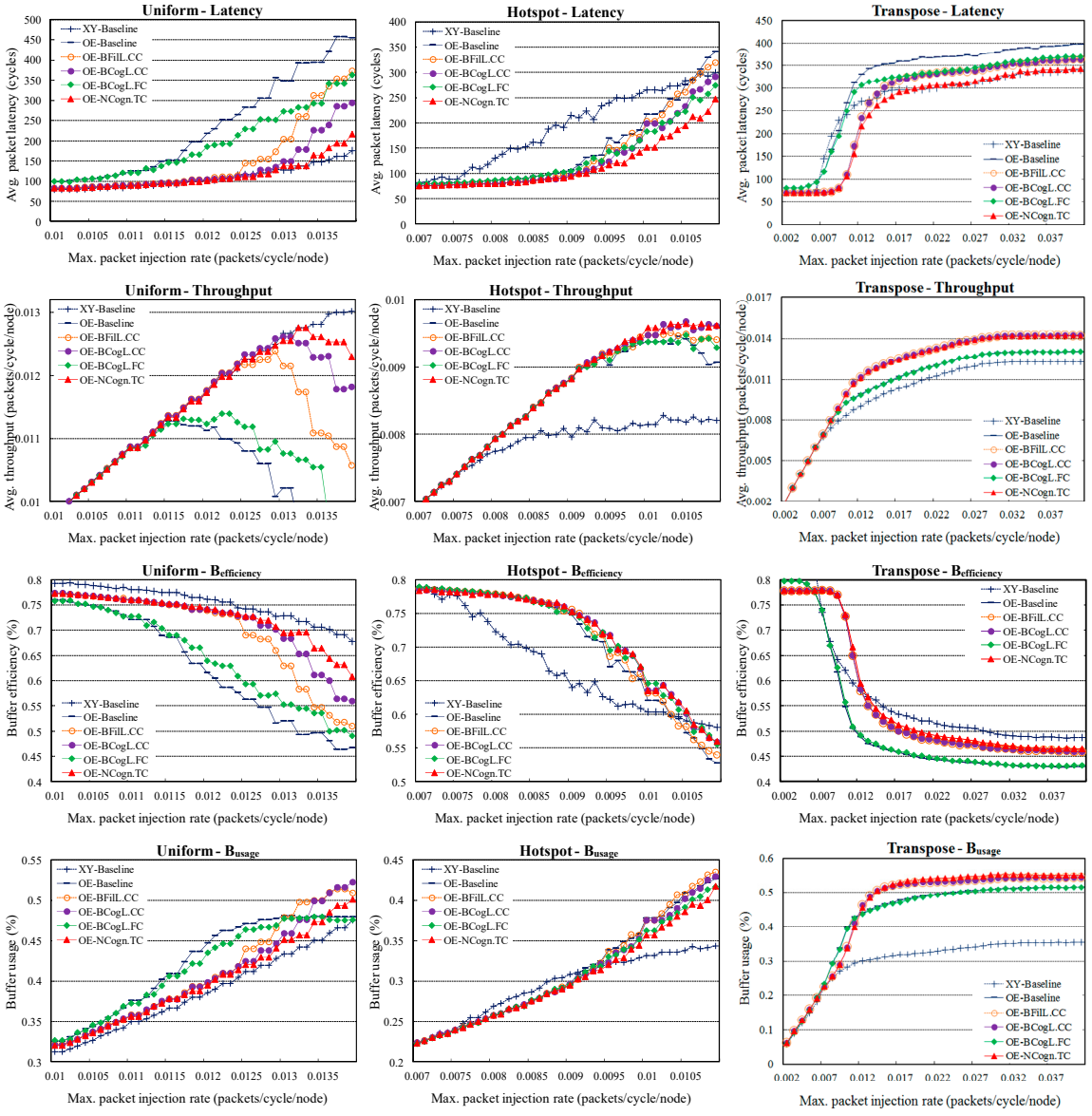

Figure 14. Performance variations in latency (top), throughput (top-middle), buffer efficiency (bottom-middle), and buffer usage (bottom) under (**a**) uniform, (**b**) hotspot, and (**c**) transpose traffics.

Table 4. Performance comparisons in average latency, maximal throughput, average buffer efficiency, and average buffer usage among different control schemes and synthetic traffic types.

Performance	Average Latency			Maximal Throughput			Average Buffer Efficiency			Average Buffer Usage		
Traffic Type	**Unif.**	**Hots.**	**Tran.**	**Unif.**	**Hots.**	**Tran.**	**Unif.**	**Hots.**	**Tran.**	**Unif.**	**Hots.**	**Tran.**
XY-Baseline	108.88	192.83	270.45	0.0130	0.0083	0.0123	0.7539	0.6695	0.4666	0.3870	0.2960	0.3012
OE-Baseline	234.9	148.3	317.45	0.0113	0.0095	0.0130	0.6234	0.7066	0.5084	0.4232	0.3106	0.4299
OE-BFilL.CA	167.75	141.4	271.63	0.0121	0.0095	0.0142	0.6792	0.7173	0.5496	0.4100	0.3109	0.4460
OE-BFilL.CR	215.65	147.13	320	0.0116	0.0094	0.0131	0.6385	0.7088	0.5084	0.4258	0.3112	0.4341
OE-BFilL.CC	152.43	139.45	272.38	0.0124	0.0095	0.0143	0.6965	0.7103	0.5520	0.4094	0.3111	0.4514
OE-BCogL.CA	160.78	141.65	271.28	0.0123	0.0094	0.0142	0.6277	0.7084	0.5499	0.4091	0.3108	0.4458
OE-BCogL.CR	183.08	137.35	319.78	0.0119	0.0097	0.0131	0.6874	0.7165	0.5088	0.4236	0.3083	0.4337
OE-BCogL.CC	129	130.58	272.2	0.0126	0.0097	0.0142	0.7155	0.7182	0.5521	0.4059	0.3081	0.4513
OE-BCogL.FC	193.7	133.1	294.45	0.0114	0.0095	0.0131	0.6439	0.7164	0.5104	0.4182	0.3058	0.4282
OE-NCogn.TC	113.95	117.35	253.95	0.0128	0.0096	0.0142	0.7281	0.7185	0.5590	0.3992	0.3031	0.4566

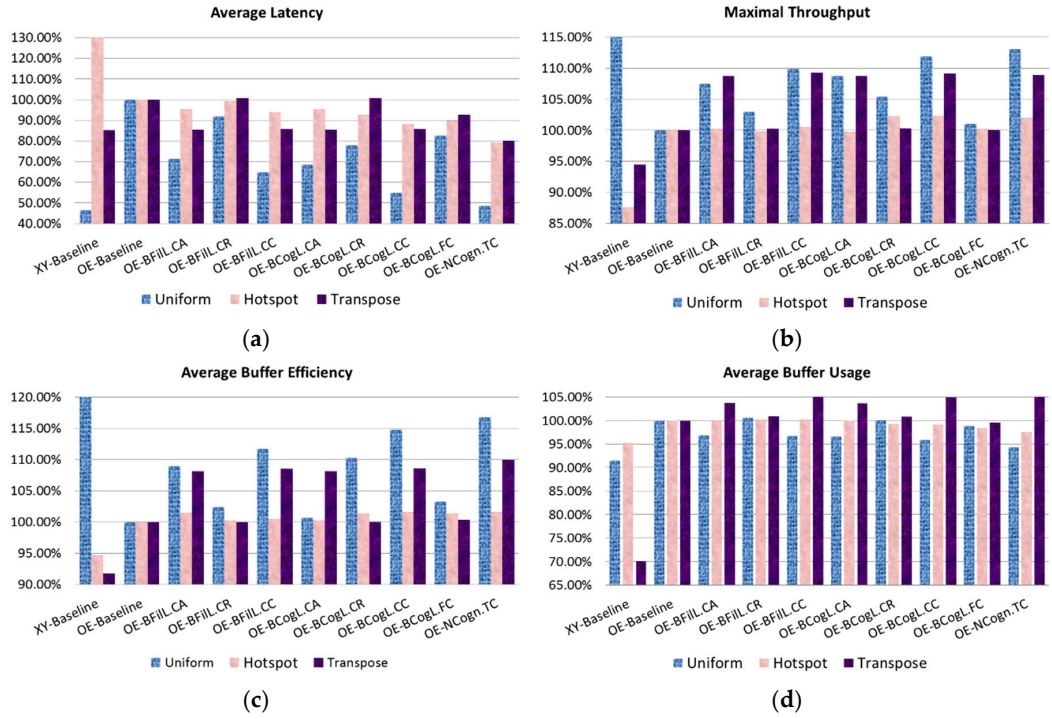

Figure 15. Comparisons in (**a**) average latency, (**b**) maximal throughput, (**c**) average buffer efficiency, and (**d**) average buffer usage among different traffic control schemes and synthetic traffic types.

5.2.2. Hotspot Traffic

In contrast to uniform traffic, Figure 14b shows that the XY-Baseline is much inferior in performance under hotspot traffic. Hotspot is a more realistic traffic scenario [15], since, in a real system, some nodes such memories probably collect more traffic flows from other nodes. Instead of waiting in front of traffic jams that are caused by hotspot nodes, adaptive routing algorithms, such as Odd-Even, support a certain degree of routing freedom for packets to route around traffic busy blocks through other available routing paths. Referring to Table 4 and Figure 15, OE-NCogn.TC averagely improves OE-Baseline 20.87% in latency, 2.03% in throughput, 1.67% in buffer efficiency, and 2.44% in buffer usage.

5.2.3. Transpose Traffic

The transpose traffic pattern is a kind of specific operations that is similar to the Matrix-Transpose pattern. The communication graph can be visualized as a number of concentric squares on which

the source and destination pairs lie between northwest and southeast areas of the mesh. In such a communication scenario, half of all uni-directional links are unused for XY routing, consequently leading to the extra low buffer usage, as shown in Figure 14c (bottom). On the contrary, Odd-Even can utilize all routing paths, thus it can perform much better. When compared with the only latency data in Odd-Even [15], we analyzed both latency and throughput. We found that the throughput is saturated much slowly, and the latency is finally saturated due to the regular traffic behavior compared to Uniform and Hotspot. It is notable that OE-NCogn.TC achieved the lowest latency and the highest throughput when compared with other traffic control schemes. Referring to Table 4 and Figure 15, OE-NCogn.TC averagely improves OE-Baseline 20.00% in latency, 8.94% in throughput, 9.96% in buffer efficiency, and −6.22% in buffer usage.

5.2.4. Overall Analyses

In accordance with Table 4, we normalized each data to OE-Baseline (100%) and then showed them in Figure 15, in which we observed some meaningful performance phenomena among most traffic control schemes, as follows: (1) XY-Baseline behaves diversely among different traffic types due to its non-adaptive routing property. (2) In congestion controls, CC performs better than CA and CA is better than CR. (3) Adaptive routing selections in BCogL can achieve more performance gains than in BFilL. (4) Except for XY-Baseline under uniform traffic, NCogn.TC performs the best among all traffic types. (5) The average buffer usage of Hotspot in Figure 15d is higher than that of Uniform, but the average buffer efficiency of Hotspot presented in Figure 15c is much smaller than that of Uniform, the reason is that Hotspot causes more waiting packets due to traffic jams than Uniform. (6) XY-Baseline has an ultra-low buffer usage in transpose traffic due to the imbalanced link utilization. (7) Basically, buffer usage and efficiency are in direct proportion; however, it is notable that NCogn.TC has the lowest buffer usage but has the highest buffer efficiency among all traffic control schemes based on Odd-Even. These statistics can validate the outstanding performance metrics of both low latency and high throughput by utilized our proposed NCogn.TC scheme.

5.3. Experiments with Real Traffics

As we know, real data transmissions of MPSoCs are much regular than synthetic patterns. Therefore, we used E3S benchmarks from the Embedded Microprocessor Benchmark Consortium (EEMBC) [39], to realize the variations. The experiments used the same setting for synthetic traffics, but each of the three adopted real traffic patterns: consumer, auto-indust, and telecom, was repeated 100 times and run on a 4×4, 5×5, and 6×6 mesh, respectively. In our experiences, the task mapping of a communication graph on the mesh can greatly influence the final performance result in different manual assignments. Task mapping shall put tasks with heavy communications closer to optimize the total communication cost. For a fair comparison, here we adopt a public simulated annealing tool [40] and show its task mapping results in Figure 16 (top). The cost function for task mapping is:

$$\sum_{\forall W_{ij} \in task\ graph} w_{ij} \times \left(\left| x_i - x_j \right| \right) + \left(\left| y_i - y_j \right| \right) \tag{9}$$

The simulation results showed that there is no distinct throughput performance difference among router traffic control schemes presented in Figure 16 (bottom). The reason is that most communications of well-mapped tasks in the mesh, as shown in Figure 16 (top), just took one hop between neighbor nodes. According to the throughput analysis presented in Section 3.2, either BFilL.CC or BCogL.CC scheme caused little effect in throughput for such local and regular traffic flows. Nevertheless, we found that the OE-BCogL.FC has a significant latency performance enhancement, which is even better than OE-NCogn.TC, as shown in Figure 16 (middle). The reason is that, as Section 3.3 discussed, BCogL.FC (part of OE-NCogn.TC in Figure 13) can efficiently decrease latency without sacrificing throughput performance. However, BCogL.CC (part of OE-NCogn.TC in Figure 13) trends to accommodate more flits in order to enhance throughput performance, which could be unfavorable for

latency performance, as discussed in Section 3.5. Referring to Table 5 and compared with OE-Baseline, OE-BCogL.FC, respectively, improves latency performance 3.39%, 17.85%, and 15.18% in consumer, auto-indust, and telecom real traffics, without sacrificing the throughput performance.

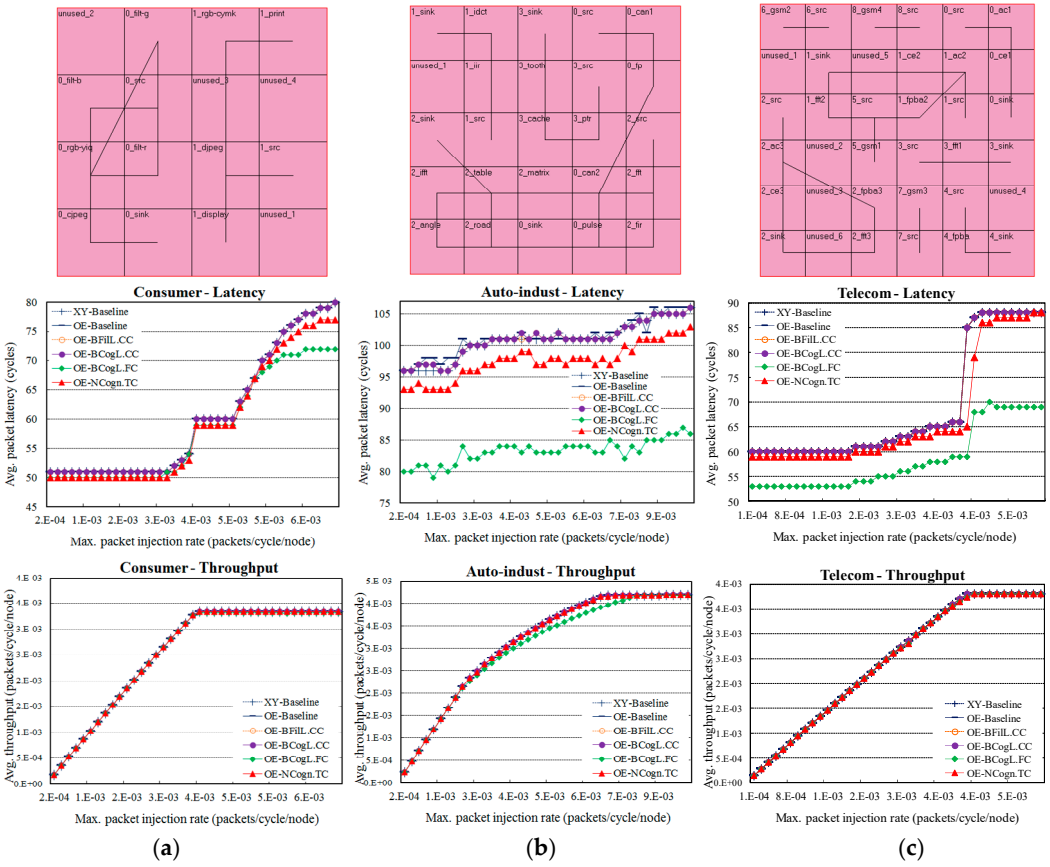

Figure 16. Task mapping results (top) and performance variations in latency (middle) and throughput (bottom) under (**a**) consumer, (**b**) auto-indust, and (**c**) telecom traffics.

Table 5. Performance comparisons in average latency among different traffic control schemes and real traffic types.

Performance	Average Latency		
Traffic Type	**Consumer**	**Auto-Indust**	**Telecom**
XY-Baseline	60.43	100.83	68.83
OE-Baseline	60.43	101.28	68.83
OE-BFilL.CA	60.43	100.9	68.83
OE-BFilL.CR	60.43	101.3	68.83
OE-BFilL.CC	60.43	100.93	68.83
OE-BCogL.CA	60.43	100.9	68.83
OE-BCogL.CR	60.43	101.3	68.83
OE-BCogL.CC	60.43	100.95	68.83
OE-BCogL.FC	58.38	83.2	58.38
OE-NCogn.TC	59.23	97.58	67.1

6. Discussion

In this paper, we intended to realize the fluidity concept in an integrated, effective, practical, and economic traffic control scheme that achieves a good tradeoff among demanding on-chip requirements for NoCs. As the description in [13]: "in the most extreme scheme, contention can be eliminated entirely by scheduling the time and location of all packets globally". Thus, a better performance can be achieved in another scheme, but the implementation overhead is probably higher than our scheme. From what has been demonstrated before, we believe that the buffer cognition level information can help to enhance the performance of most traffic control schemes that are originally based on the common buffer fill level information. The differences of this paper from the previous works and other researches are described, as follows:

1. First, using the proposed novel buffer cognition level to analyze the performance root causes, we realized the integrated traffic control scheme that jointly considers both congestion and flow controls, which were designed in distinct schemes and evaluated separately.
2. Next, performances were comprehensively analyzed in not only network latency and throughput, but also buffer efficiency and usage. Furthermore, the real traffic patterns were well mapped in the multi-processing elements of the implemented NoC by the applied public tool in order to evaluate the performance in both conditions of real traffic and real task-mapping tool.
3. Last, and especially, our scheme was implemented as a pure hardware module, which uses neither extra network bandwidth nor additional sideband signal. With these practical properties of reusability, feasibility, and scalability, NCogn.TC can be easily integrated with most NoC architectures to gain extra performance.

From the buffer fluidity point of view, we looked into the latency and throughput factors of our proposed Network-Cognitive Traffic Control (NCogn.TC) scheme and concluded with three distinct principles:

1. First, instead of increasing the buffer usage as in Equation (8) by buffering as many as flits that a network can afford, NCogn.TC improved the buffer efficiency, as in Equation (7), by making as many as flits kept in fluidity to deliver better performance.
2. Second, low latency does not indeed lead to high throughput, and vice versa. For example, on one hand, a bigger buffer is good for throughput, because it can spread the burst traffic evenly; on the other hand, the bigger buffer causes worse packet latency due to the longer distance (or waiting time) from input to output when the buffer is near full, as in Equation (6). Additionally, the maximal throughput limitation is relevant to the minimum cut of the flow network based on the max-flow-min-cut theorem, as described in Equations (3)–(5); as experiments with real traffics justified and explained in Section 5.3, this phenomenon is distinct in regular and local traffic flows, as shown in Figure 16.
3. Last, based on the above analyses, by monitoring the link-layer flow volume, our proposed NCogn.TC can dynamically regulate the buffering of in-transit flits, thereby balancing the performance requirements of packet delivery latency and overall network throughput.

7. Conclusions

When compared with previous works and other researches separately dealing with NoC congestion avoidance, congestion relief, link-layer flow control, and transmission-layer flow control, the proposed NCogn.TC method jointly considers avoiding congestions in downstream nodes, releasing congestions in upstream nodes, dynamically adjusting filled flit number in local nodes, and rapidly regulating flit injection rate at each transmission end. Besides, instead of exchanging message between communication pairs, collecting neighboring nodes' information to do traffic predictions, while using additional buffers and channels to relieve local congestions, or adding a large number of control wires to look ahead distant signals, NCogn.TC reacts the traffic conditions based on monitoring its own buffer status.

That is, neither extra network bandwidth nor additional sideband control signal are required for applying NCogn.TC, which means that a significant advantage that NCogn.TC can be easily integrated into most existent NoC architectures to gain extra performance. In NoCs, a considerable number of researches have been conducted on congestion and flow controls, which were designed in distinct schemes and evaluated separately. However, there seems to be no established integrated method as NCogn.TC to discuss the global NoC traffic control requirements. This is intended to be the major contribution of this paper.

Author Contributions: Methodology, implementation, and writing, W.-C.T.; advisement and review, S.-J.C.; supervision and writing, Y.-H.H.; editing and corresponding, M.-L.C. All authors have read and agreed to the published version of the manuscript.

Funding: This work is partially supported by the MOST, ROC, under grant numbers of MOST 108-2221-E-324-011 and MOST 109-2221-E-324-018.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Benini, L.; Micheli, G.D. Networks on chips: A new SoC paradigm. *IEEE Comput.* **2002**, *35*, 70–78. [CrossRef]
2. Owens, J.D.; Dally, W.J.; Ho, R.; Jayasimha, D.N.; Keckler, S.W.; Peh, L.S. Research challenges for on-chip interconnection networks. *IEEE Micro* **2007**, *27*, 96–108.
3. Wu, Q.; Chen, L.; Wang, Z.; Ding, G.; Zhang, L.; Zhang, X. Information measurement of cognitive communication systems: The introduction of negative cognitive information. *IEEE Access* **2018**, *6*, 34288–34295. [CrossRef]
4. Wu, W.T.; Louri, A. A methodology for cognitive NoC design. *IEEE Comput. Archit. Lett.* **2016**, *15*, 1–4. [CrossRef]
5. Melo, D.R.; Zeferino, C.A.; Dilillo, L.; Bezerra, E.A. Maximizing the inner resilience of a network-on-chip through router controllers design. *Sensors* **2019**, *19*, 5416. [CrossRef]
6. Chang, E.J.; Hsin, H.K.; Lin, S.Y.; Wu, A.Y. Path-congestion-aware adaptive routing with a contention prediction scheme for network-on-chip systems. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* **2014**, *33*, 113–126. [CrossRef]
7. Tang, M.; Lin, X.; Palesi, M. Local congestion avoidance in network-on-chip. *IEEE Trans. Parallel Distrib. Syst.* **2016**, *27*, 2062–2073. [CrossRef]
8. Jiang, N.; Becker, D.U.; Michelogiannakis, G.; Dally, W.J. Network congestion avoidance through speculative reservation. In Proceedings of the IEEE International Symposium on High Performance Computer Architecture, New Orleans, LA, USA, 25–29 February 2012; pp. 1–12.
9. Mehranzadeh, A.; Khademzadeh, A.; Bagherzadeh, N.; Reshadi, M. DICA: Destination intensity and congestion-aware output selection strategy for network-on-chip systems. *IET Comput. Digit. Tech.* **2019**, *13*, 335–347. [CrossRef]
10. Giroudot, F.; Mifdaoui, A. Graph-based approach for buffer-aware timing analysis of heterogeneous wormhole NoCs under bursty traffic. *IEEE Access* **2020**, *8*, 2169–3536. [CrossRef]
11. Lan, Y.C.; Tsai, M.C.; Su, A.P.; Hu, Y.H.; Chen, S.J. Fluidity concept for NoC: A congestion avoidance and relief routing scheme. In Proceedings of the IEEE International System-on-Chip Conference, Newport Beach, CA, USA, 17–20 September 2008; pp. 65–70.
12. Tsai, W.C.; Lin, H.E.; Lan, Y.C.; Chen, S.J.; Hu, Y.H. A novel flow fluidity meter for BiNoC bandwidth resource allocation. In Proceedings of the IEEE International System-on-Chip Conference, Beijing, China, 8–11 September 2015; pp. 281–286.
13. Benini, L.; Micheli, G.D. *Networks on Chips: Technology and Tools*; Morgan Kauffmann: San Mateo, CA, USA, 2006.
14. Glass, C.J.; Ni, L.M. The turn model for adaptive routing. *J. ACM* **1994**, *41*, 874–902. [CrossRef]
15. Chiu, G.M. The odd-even turn model for adaptive routing. *IEEE Trans. Parallel Distrib. Syst.* **2000**, *11*, 729–738. [CrossRef]

16. Ye, T.T.; Benini, L.; Micheli, G.D. Packetization and routing analysis of on-chip multiprocessor networks. *J. Syst. Archit.* **2004**, *50*, 81–104. [CrossRef]
17. Hu, J.; Marculescu, R. DyAD—Smart routing for networks-on-chip. In Proceedings of the Design Automation Conference, San Diego, CA, USA, 7–11 July 2004; pp. 260–263.
18. Ascia, G.; Catania, V.; Palesi, M.; Patti, D. Implementation and analysis of a new selection strategy for adaptive routing in networks-on-chip. *IEEE Trans. Comput.* **2008**, *57*, 809–820. [CrossRef]
19. Tobuschat, S.; Ernst, R. Providing throughput guarantees in mixed-criticality networks-on-chip. In Proceedings of the IEEE International System-on-Chip Conference, Munich, Germany, 5–8 September 2017; pp. 292–297.
20. Nain, Z.; Ali, R.; Anjum, S.; Afzal, M.K.; Kim, S.W. A network adaptive fault-tolerant routing algorithm for demanding latency and throughput applications of network-on-a-chip designs. *Electronics* **2020**, *9*, 1076. [CrossRef]
21. Pullini, A.; Angiolini, F.; Bertozzi, D.; Benini, L. Fault tolerance overhead in network-on-chip flow control schemes. In Proceedings of the Symposium on Integrated Circuits and System Design, Florianopolis, Brazil, 4–7 September 2005; pp. 224–229.
22. Bertozzi, D.; Benini, L. Xpipes: A network-on-chip architecture for gigascale systems-on-chip. *IEEE Circuits Syst. Mag.* **2004**, *4*, 18–31. [CrossRef]
23. Concer, N.; Bononi, L.; Soulie, M.; Locatelli, R. CTC: An end-to-end flow control protocol for multi-core systems-on-chip. In Proceedings of the ACM/IEEE International Symposium on Networks-on-Chip, San Diego, CA, USA, 10–13 May 2009; pp. 193–202.
24. Nychis, G.P.; Fallin, C.; Moscibroda, T.; Mutlu, O.; Seshan, S. On-chip networks from a networking perspective: Congestion and scalability in many-core interconnects. In Proceedings of the ACM SIGCOMM conference on Applications, technologies, architectures, and protocols for computer communication, Helsinki, Finland, 13–17 August 2012; pp. 407–418.
25. Postman, J.; Krishna, T.; Edmonds, C.; Peh, L.S.; Chiang, P. SWIFT: A low-power network-on-chip implementing the token flow control router architecture with swing-reduced interconnects. *IEEE Trans. Very Large Scale Integr. Syst.* **2013**, *21*, 1432–1446. [CrossRef]
26. Yousefzadeh, A.; Jablonski, M.; Iakymchuk, T.; Linares-Barranco, A.; Rosado, A.; Plana, L.A.; Temple, S.; Serrano-Gotarredona, T.; Furber, S.B.; Linares-Barranco, B. On multiple AER handshaking channels over high-speed bit-serial bidirectional LVDS links with flow-control and clock-correction on commercial FPGAs for scalable neuromorphic systems. *IEEE Trans. Biomed. Circuits Syst.* **2017**, *11*, 1133–1147. [CrossRef]
27. Goossens, K.; Dielissen, J.; Radulescu, A. Æthereal network on chip: Concepts, architectures, and implementations. *IEEE Des. Test Comput.* **2005**, *22*, 21–31. [CrossRef]
28. Shin, K.G.; Daniel, S.W. Analysis and implementation of hybrid switching. *IEEE Trans. Comput.* **1996**, *45*, 684–692. [CrossRef]
29. Lu, Z.; Liu, M.; Jantsch, A. Layered switching for networks on chip. In Proceedings of the Design Automation Conference, San Diego, CA, USA, 4–8 June 2007; pp. 122–127.
30. Wang, X.; Gan, G.; Fan, D.; Guo, S. GFFC: The global feedback based flow control in the NoC design for many-core processor. In Proceedings of the IFIP International Conference on Network and Parallel Computing, Gold Coast, Queensland, Australia, 19–21 October 2009; pp. 227–232.
31. Kim, H.; Kim, G.; Yeo, H.; Kim, J.; Maeng, S. Design and analysis of hybrid flow control for hierarchical ring network-on-chip. *IEEE Trans. Comput.* **2016**, *65*, 480–494. [CrossRef]
32. Qian, Y.; Lu, Z.; Dou, W. Analysis of worst-case delay bounds for best-effort communication in wormhole networks on chip. In Proceedings of the ACM/IEEE International Symposium on Network-on-Chip, San Diego, CA, USA, 10–13 May 2009; pp. 44–53.
33. Guerrier, P.; Greiner, A. A generic architecture for on-chip packet-switched interconnections. In Proceedings of the Design, Automation and Test in Europe Conference and Exhibition, Paris, France, 27–30 March 2000; pp. 250–256.
34. AMBA Specification. ARM Limited. Available online: https://www.arm.com/products/silicon-ip-system/embedded-system-design/amba-specifications (accessed on 3 September 2020).
35. Gross, D.; Shortle, J.F.; Thompson, J.M.; Harris, C.M. *Fundamentals of Queueing Theory*, 4th ed.; John Wiley & Sons, Inc.: Hoboken, NJ, USA, 2013.

36. Tsai, W.C. Network-on-Chip Router Design. Ph.D. Thesis, Graduate Institute of Electronics Engineering, Taiwan University, Taipei, Taiwan, 2011.
37. Wu, D.; Al-Hashimi, B.M.; Schmitz, M.T. Improving routing efficiency for network-on-chip through contention-aware input selection. In Proceedings of the Asia and South Pacific Design Automation Conference, Yokohama, Japan, 24–27 January 2006; pp. 36–41.
38. Hu, J.; Ogras, U.Y.; Marculescu, R. System-level buffer allocation for application-specific networks-on-chip router design. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* **2006**, *25*, 2919–2933. [CrossRef]
39. Dick, R. Embedded System Synthesis Benchmark Suites (E3S). Available online: http://ziyang.eecs.umich.edu/~{}dickrp/e3s/ (accessed on 3 September 2020).
40. B*-Tree Based Placement/Floorplanning. NTU EDA Lab. Available online: http://eda.ee.ntu.edu.tw/research.htm (accessed on 3 September 2020).