TS-Router: On Maximizing the Quality-of-Allocation in the On-Chip Network

Yuan-Ying Chang[†], Yoshi Shih-Chieh Huang[†], Matthew Poremba[‡],
Vijaykrishnan Narayanan[‡], Yuan Xie[‡,§], and Chung-Ta King[†]

[†]Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan, {elmo,yoshi,king}@cs.nthu.edu.tw
[‡]Department of Computer Science and Engineering, Pennsylvania State University, University Park, USA,
{mrp5060,vijay,yuanxie}@cse.psu.edu
[§]AMD Research, yuan.xie@amd.com

Abstract

Switch allocation is a critical pipeline stage in the router of an Network-on-Chip (NoC), in which flits in the input ports of the router are assigned to the output ports for forwarding. This allocation is in essence a matching between the input requests and output port resources. Efficient router designs strive to maximize the matching. Previous research considers the allocation decision at each cycle either independently or depending on prior allocations. In this paper, we demonstrate that the matching decisions made in a router along time actually form a time series, and the Quality-of-Allocation (QoA) can be maximized if the matching decision is made across the time series, from past history to future requests. Based on this observation, a novel router design, TS-Router, is proposed. TS-Router predicts future requests to arrive at a router and tries to maximize the matching across cycles. It can be extended easily from most state-of-the-art routers in a lightweight fashion. Our evaluation of TS-Router uses synthetic traffic as well as real benchmark programs in full-system simulator. The results show that TS-Router can have higher number of matchings and lower latency. In addition, a prototype of TS-Router is implemented in Verilog, so that power consumption and area overhead are also evaluated.

I. Introduction

As the number of cores keeps increasing on chip multiprocessors, the Network-on-Chip (NoC) technology is becoming essential to interconnect these cores [16], [6]. Many prior efforts on the design of efficient routers have been developed to design efficient router for NoCs. One class of works focused on the resource allocation inside a router, including *Virtual-channel Allocation* (VA) and *Switch Allocation* (SA), because these operations are in the critical path of the router pipeline [24], [18]. A key issue in resource allocation is the Quality-of-Allocation (QoA), which is the subject of serveral recent works [2], [3], [21], [23].

QoA refers to the ability of a router to match input port requests with output port resources [3], which is often measured as the number of matches that can be made in a cycle. A higher QoA means that a router can move more packets across its internal switches in a cycle, resulting in higher throughput. Most existing works aim at maximizing the number of matches but only consider the allocation within a single cycle. In fact, the matching decisions made in a router along time actually form a time series[1]. A greedy allocation algorithm which maximizes the matching in each cycle might not lead to best QoA across the whole time series.

Recent works such as *Packet Chaining* [21] and *Pseudo-Circuit* [2] have discovered that allocation decisions made in the current cycle may be affected by those made in the previous cycles. Allocation strategies are thus proposed to improve the number of matches in a router by inheriting the results of previous cycles. Experiment results show this strategy can even outperform maximal matching via wavefront allocator [26] and maximum matching via augmenting path algorithm [9]. This implies that maximal/maximum matching within a single cycle is not good enough and back-to-back allocations should be considered.

Although the idea of performing switch allocation based on past history works well, the requirement to achieve good performance is still quite rigid – the current allocation must have *suitable* similarity with the previous one. Unfortunately, this is only valid in some specific cases. If the similarity is too low, the performance of history-based strategies [21], [2] will degenerate to that of independent allocation [20]. On the other hand, if the similarity is too high, the performance will also be degraded because inheriting the previous allocation will *starve* the new requests. Therefore, a more general and less demanding strategy is needed.

In this paper, we discuss Quality-of-Allocation of switch allocation from the perspective of time series of matching be-

[1]Generally speaking, a time series can be a sequence of data, results, or decisions, which have happen-before relationships and therefore can be represented along with time.

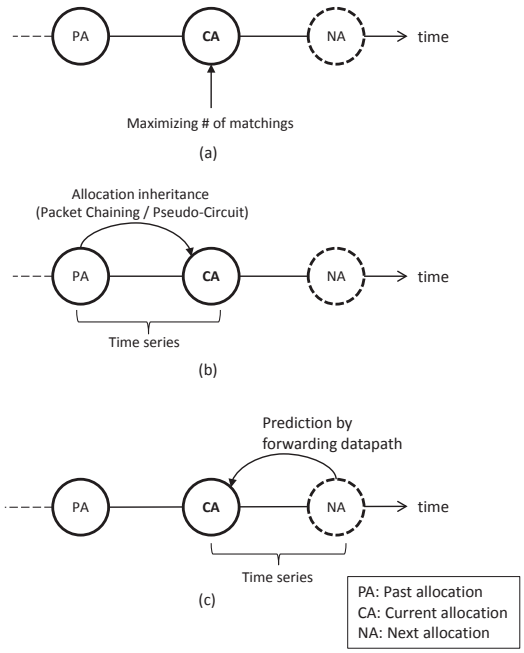


Fig. 1: An overview of different designs. (a) The initial works for maximizng the number of matchings. (b) History-based solution, which uses previous allocation(s) to improve current allocation. (c) TS-Router predicts the future requests to improve the current allocation.

tween input port requests and output port resources. With such a perspective, *the allocation decision made at each cycle should consider not only* ***past decisions*** *but also* ***future requests***. In the next section, we will show an example to illustrate that maximizing the number of allocations in a single cycle is not enough, since current allocation will affect future allocations. If the requests in the following cycles can be predicted, the allocation decision made in the current cycle can be adjusted accordingly, which results in more concurrent connections in consecutive allocations. Figure 1 illustrates an overview of previous works (maximizing matches in a single cycle and history-based) and our design.

Based on this observation, a novel router design, TS-Router, is proposed. TS-Router predicts future requests to arrive at a router and tries to maximize the matching across cycles. In this paper, a forwarding design in the router architecture for maximizing the current and future allocations is proposed, in which possible conflicts in the future are predicted and resolved, resulting in more concurrent connections along time. In contrast to naive prediction, the forwarding datapath directly forwards the requests from virtual-channel allocation stage to switch allocation stage, which results in accurate prediction.

To summarize, the main contributions of this paper are as follows:

- To the best of our knowledge, this is the first work to propose the concept of time-series matchings, which is in contrast to maximal and maximum matching.
- A novel router design, TS-Router, is proposed, which realizes the idea of time-series matching.

This paper is organized as follows: In the next section, preliminaries of this paper are introduced, including a background of allocation and matching, followed by the representation of an allocation. A motivating example is given in Section III to show that maximal and maximum matching considered within a single cycle do not lead to a global optimal allocation. Instead, time-series-based allocation outperforms these allocation algorithms. In Section IV, the design of TS-Router is introduced, including a discussion of the implementation overheads. For evaluating the performance of TS-Router, we present comprehensive experimental results in Section V to compare the TS-Router with the classic iSLIP allocator [20] and the most recent allocation algorithm [21]. Finally, related works are discussed in Section VI, and the conclusions and future work are given in Section VII.

II. Preliminaries

In this section, we give a brief introduction on the preliminaries on allocation in NoC architecture.

A. Allocation and Matching

An allocation or a matching in a router is to pair the input ports (in-ports) and output ports (out-ports). To avoid conflicts, one input port can only be paired with one output port, and vice versa. A *maximal matching* means that while there is an existing matching, it is impossible to add more pair(s) to the existing matching. On the contrary, a *maximum matching* means that it is the largest matching in terms of the number of paired input-to-output ports. Note that a maximum matching is certainly a maximal matching. However, a maximal matching is not necessarily a maximum matching. [2]

B. Representation of Allocation

We use a *request matrix* to show requesting-requested relationships between input ports and output ports in the stage of Switch Allocation (SA). Each row stands for an input port and the cells in a row are the requested output ports. Each column stands for an output port and the cells in a column are the input ports that are requesting for this output port. White circles stands for the requests. A $V \times U$ request matrix can be represented as a bipartite graph denoted as $G = (V; U; E)$, and vice versa.

Once an allocation is done, the resultant matrix has the following properties: (1) Each row can only contains one white circle, that is, each input port can request at most one output port at a time. (2) Each column can contain only one white circle, that is, each output port can only be requested by at most an input port at a time. A request matrix can also be depicted as a bipartite graph $G = (V; U; E)$ but each element in V can only have one outgoing link to U, and vice versa. The corresponding bipartite graph of a request matrix forms a *matching*.

[2]In the following paragraphs, if a matching is maximal and maximum simultaneously, we denote such matchings as *max matchings* for short.

III. Motivating Examples

In the following examples, we discuss the influences of max matching and show the cases that max matchings do not lead to the best results. We assume the pipelined router architecture as in [21]. For simplifying the discussion, we assume a 5-port router and the number of virtual channel is set to 2.

In this example, we use *request matrix* to represent the relationship between input ports and output ports. We use white circles to state the requests in the stage of Switch Allocation (SA), and black dot is specially for representing the future requests of SA stage. The gray cells stands for the previous grants in last allocation, which is used by *history-based strategy*, such as in [21], [2].

Figure 2a shows two possible solutions denoted as i and i'. Both of the two solutions achieve max matchings simultaneously (3 in this case). Then, Figure 2b shows the following allocation. Previously unallocated requests are left in $(i+1)$ and $(i+1)'$, respectively. Moreover, the future request, depicted in Figure 2a with the black dot, becomes a current request in Figure 2b. Note that both of the allocation $(i+1)$ and $(i+1)'$ are max matchings (2 and 1, respectively). However, allocation $(i+1)'$ results in an input port conflict and therefore the latter max matching is worse than the former.

This example shows that in two back-to-back allocations, the current allocation may affect the next allocation. If we only consider each single allocation, both of the two allocations achieve max matchings and the drawback cannot be found. Note that this example only shows the case of column conflicts. The same idea can be applied to row conflicts.

Next, with the gray cells, we are able to observe the allocations made by history-based strategies [21], [2]. It shows that history-based strategies provide higher priorities to the previous grants. However, in this example, it would select the allocation $(i+1)'$ based on the previous grants, and it eliminates the possibility of concurrent transmission since it does not take the future requests into consideration. To conclude, history-based strategies are more conservative and tend to reserve the previous connections to lower the risks of ports being idle. In contrast, TS-Router takes the current allocation and incoming allocation into consideration and explores more possibilities of parallel connections. In other words, history-based strategies are *look-behind* strategies which uses history of grants to improve the QoA. In contrast, TS-Router not only considers the history, but also contains a *look-ahead* strategy which uses the forwarding messages to predict the incoming matchings.

IV. Design Concept of TS-Router

In the following discussion, we first give the router architecture to use. Second, we give the formulation of matching maximization for current-next allocations. Third, we give the detailed operations of an allocation with the future requests. An illustration is given for demonstrating the operations step-by-step. Finally, we discuss the modifications in the datapath to support the prediction of future requests.

A. The General Router Architecture

We use a general router architecture mentioned in [1], [24] as our baseline router architecture. The router has five pipeline stages: *Routing Computation* (RC), *Virtual-channel Allocation* (VA), *Switch Allocation* (SA), *Switch Traversal* (ST), and *Link Traversal* (LT). Each router has multiple virtual channels per input ports and VC flow control is used [5]. Note that the reason that using a very general router architecture rather than optimized ones is that our idea of predicting the future requests can be realized by adding a few additional links and simple logics, which is orthogonal to the most modern designs. Therefore, using a general router architecture helps understand the spirit of our idea. However, the idea can be integrated into most modern router architectures as long as the router architecture involves VA and SA stages.

B. Time-Series Switching

To design a time-series-considered router, while doing a decision of allocation, the impact of the decision needs to be calculated by estimating the profit brought by this allocation. In a single router, for the Current-Next allocations, the Matching Quality (MQ) is defined as the resulting Numbers of Matchings (NoM) with a prospecting vision $v > 0$. The goal of MQ is as follows with a given v:

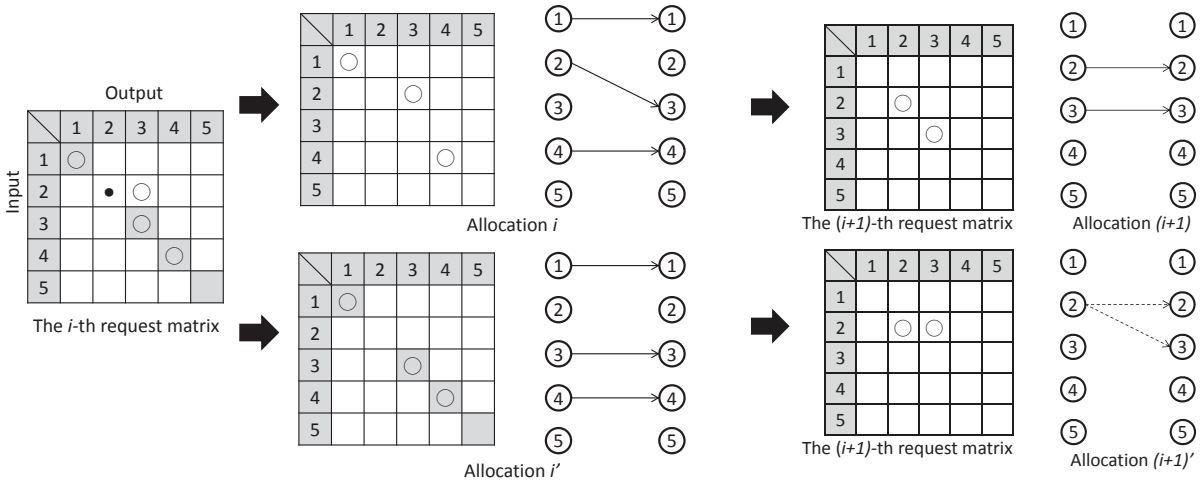
$$\max MQ(allocation_i) = \max \sum_{i<i+v} NoM_i$$

A larger v can be set for more aggressive design. Theoretically $v \to \infty$ gives the upper bound of MQ. However, setting $v \to \infty$ is not practical in real cases since the incoming traffic patterns are always unknown until the runtime. In TS-Router, v is set to 1, which means that when doing the i-th allocation, it also takes the next $(i+1)$-th allocation into consideration. This prediction is performed inside the router by forwarding, so the accuracy is relatively high. However, it is still possible that when doing i-th allocation, more than one following allocations are considered, i.e., $(i+v)$-th allocations, where $v > 1$. To do the more aggressive prediction, the information required by the prediction may not only come from the local router, but also the other neighbor routers. Nevertheless, the accuracy will be relative low since the prediction is made across the routers. Therefore, we focus on conservative prediction in this paper rather than aggressively prediction across the routers.

Figure 3 shows the datapath of TS-Router. A general separable arbitrators design can be applied. A forwarding link from VA to SA is for predicting the participants in the next allocation.

C. Priority Propagation in Priority Matrix

To represent the priority when conflict occurs in inport arbitration or outport arbitration, we use *priority matrix* which is similar to the request matrix mentioned in Section II-B. Differently, request matrix is a binary matrix, i.e., each entry in a cell is either 0 or 1, but an entry in a priority matrix is a value indicating the priority when conflict occurs.



(a) The i-th request matrix

(b) The $(i+1)$-th request matrix

Fig. 2: The white circles are the current requests and the black dot is the future request in the stage of Switch Allocation (SA). The gray cells stand for the prioritized cells, which is used by *history-based strategy*, such as in [21], [2]. In this case, the gray cells are previously granted requests. Note that: (a) both allocations i and i' achieve max matchings (3 in this case). (b) In the next allocation, although allocation $(i+1)$ and $(i+1)'$ also are both max matchings (2 and 1, respectively), the latter has input port conflicts, so the max matching is worse than the former.

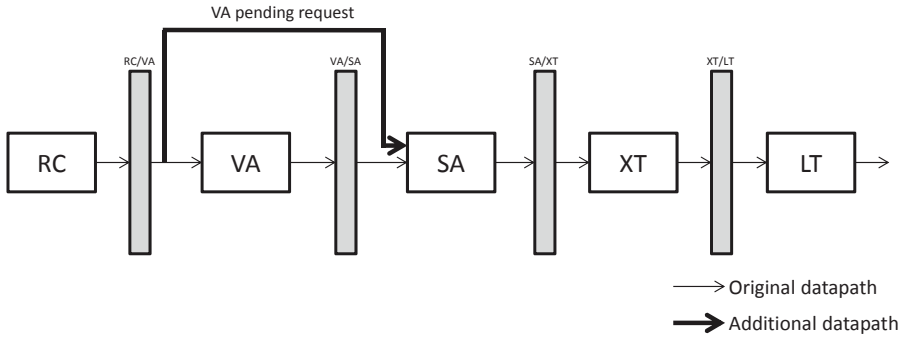


Fig. 3: Datapath of TS-Router. Dedicated links from the downstream routers provides the feedback information and stored in the VA/SA latch. One forwarding link is from the pending request in the VA stage for predicting the participants in the next allocation.

An predicted request, represented by a black dot in the priority matrix, means that the occupied cell (standing for an inport and an outport) has a request in the next allocation, so it propagates the priorities to the requests in the same column and the same row except itself based on the following observations:

1) Clean the competitor(s): to avoid them occupying the inport or the outport in the next allocation.
2) Improve parallelism: Once the competitor(s) is cleared, the request(s) which is in the same row or the same column has higher probability to be transmitted with the predicted one in the next allocation.

Specifically, assuming that the predicted request is located at (m, n), the cells in the m-th row and in the n-th column except (m, n) itself are prioritized. For simplifying the explanation, we use an illustration to help understand the process. As Figure 4a shows, an predicted request is located at $(2, 2)$. According to our algorithm, the second row and the second column are prioritized except $(2, 2)$ itself. Therefore, the inport conflict between $(4, 1)$ and $(4, 2)$ is resolved by letting $(4, 2)$ win. Similarly, $(2, 3)$ wins the outport arbitration when conflicting with $(3, 3)$. Figure 4b shows the remaining requests, i.e., the new request (2,2) and the losers in the previous arbitration. Apparently, the remaining three requests do not have conflict because time-series switching has resolved the possible conflicts in the previous allocation, and therefore they can be transmitted in parallel in the next allocation.

Note that the values in the priority matrix can be accumulated in one switch arbitration. Figure 5 shows an sample priority matrix which has two predicted requests, in which the overlapped prioritized cells have higher priorities and the values are accumulated to 2. Note also that the accumulations only occur in one switch allocation rather than accumulated along time. That is, the priority matrix is refreshed to 0s every switch arbitration.

The request matrix can be implemented with several existing methods, such as *Tree Arbiter*, *Matrix Arbiter*, and so on [23],

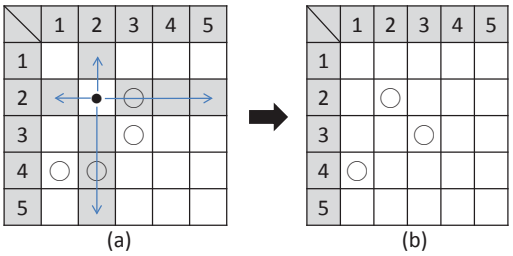


Fig. 4: (a) Gray cells are prioritized by the predicted request (black dot). As a result, (2,3) and (4,2) have higher priorities when conflicting. (b) Remaining requests can be transmitted in parallel in the next allocation.

	1	2	3	4	5
1		***1***		***1***	
2	***1***	•	***1***	***2***	***1***
3		***1***		***1***	
4	***1***	***2***	***1***	•	***1***
5		***1***		***1***	

Fig. 5: A sample of priority matrix with multiple predicted requests.

[7]. In the following subsection, we implement the TS-Router with Verilog to investigate the hardware considerations.

D. Router Implementation

To investigate the overheads in terms of several considerations, we perform estimations on the power, area, and critical path latency of our router architecture using Synopsys Design Compiler. Behavioral RTL is synthesized using ST micron's 65nm design libraries. We choose the nominal library with 1.20V core voltage. The design of our baseline is a simplistic design involving X-Y routing and only 4 stages: route computation, switch allocation, crossbar transversal, and link transversal. The stage of virtual-channel allocation is removed for simplifying the following analysis. Note that this simple baseline router may result in larger area overhead percentage when applying the logics for realizing time-series switching. However, it will be relatively small when applying the logics to other modern routers [18], [19].

a) Area overhead of priority matrix.: Before entering the RTL-based analysis, we calculate the area overhead of priority matrix to give a formal estimation. Assuming that the size of the crossbar is $N \times N$, then each cell in the priority matrix has $N-1$ neighbors in a row, and similarly $N-1$ neighbors in a column. Therefore, the accumulated priority value for a cell ranges from $[0, 2N-2]$, and the required bits to store all the values are $N^2\lceil log_2(2N-2)\rceil$ bits. Note that since the entries in priority matrix are cleared every switch arbitration, values in cells will not be accumulated along time, so the analysis above guarantees that values will not overflow.

b) Critical path, area overhead, and power consumption.: Next, we estimate the critical path by synthesizing our designs to reach the maximum frequency without violating timing constraint. Currently, the slowest path involves selecting the highest priority input port without causing conflicts on output port grants. This selection is an iterative process involving a sequential logic path. Each iteration checks for the i-th highest priority and grants the input if the output is not already granted. The list of output ports already granted is used when selecting the $(i-1)$-th highest priority input port. To boost the speed, we can check grants of multiple input ports in parallel to speed up selection. To do this, each input port must also compare the priorities of other input ports, which adds additional comparators to the design. In Figure 6, the priority matrix is calculated and then the arbitration decision needs to be done based on the priority matrix. Take the arbitration of output port 4 as an example, input port 1, 2 and 3 are involved in arbitration. Note that the maximum priority is 6 in this example. First, they compare their priority values with 6,5,4,2 and 1 in parallel. Then, these compared results are sent to *Conflict Solver*, which can process conflicts in a round-robin manner when two or more request have the same priorities. After conflicts are solved, these results are treated as control signals to *Selector*, which chooses the request with highest priority. However, we only implement the selection process in a sequential way currently.

Based on our synthesis results, our router can be run up to 333 MHz and the baseline up to 500 MHz. This results in a 33% slower router. However, this is the maximum slowdown since we can optimize the design to be more parallel at the cost of extra area. Alternatively, the iterative process can be split by dividing the switch allocation into two pipeline stages. With a completely iterative process, the design requires additional 867 gates per router than the baseline router with an absolute additional area of $61 \times 61 \mu m^2$. Besides, considering the impacts of power consumption due to NoC [28], [27], the power consumption is also compared using the design compiler tool. We re-synthesized the baseline router to operate at 333 MHz to compare the power consumption. The result shows that our design consumes additional 0.325 mW per router.

V. Evaluation

We use GEM5, a full-system simulator [4], with enabled Ruby and Garnet to model the system [17], [1], including detailed memory model and interconnection network. The evaluation is threefold. First, we focus on the network performance by switch GEM5 to network only mode, in which processing elements (PE) act as traffic injectors, and we can synthesize several conventional traffic patterns to examine the performance limitations of the network. Second, we focus on the effectiveness of time-series consideration. We use *moving average* to observe the effectiveness by comparing TS-Router with Packet Chaining. Finally, we switch GEM5 to full-system mode for running the real benchmarks. The default settings are shown in Table I. If the settings are modified in a specific experiment, they will be mentioned in the paragraphs for avoiding confusion.

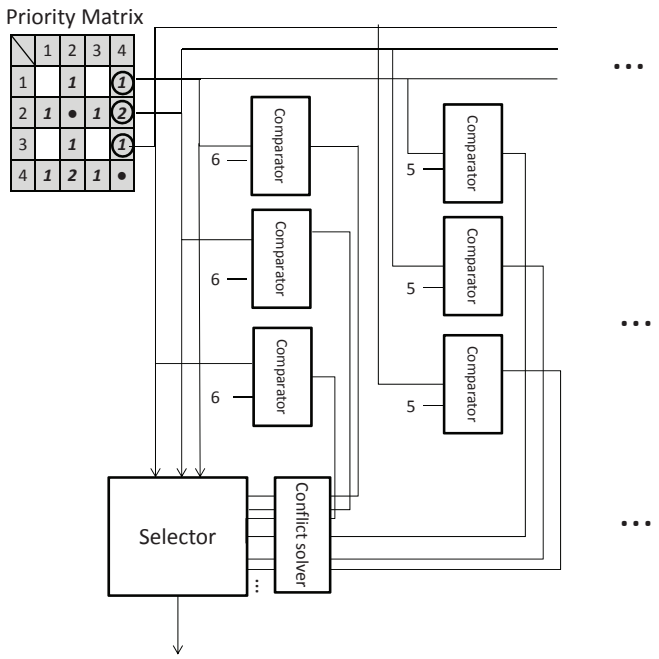


Fig. 6: Priority-based arbitration microarchitecture.

TABLE I: Default simulation setup

Simulator settings	
Processor family	ALPHA ISA
Frequency	2 GHZ
Number of cores	64
Cache protocol	MESI protocol
NoC topology	8-by-8 2D mesh network
Average Packet size	6 flits
Number of VCs	4
Input buffer size	5 flits
Routing algorithm	Dimension-order

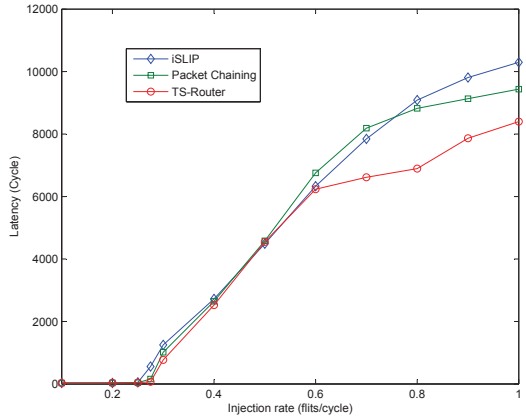


Fig. 7: The latency comparison under tornado traffic with different injection rates.

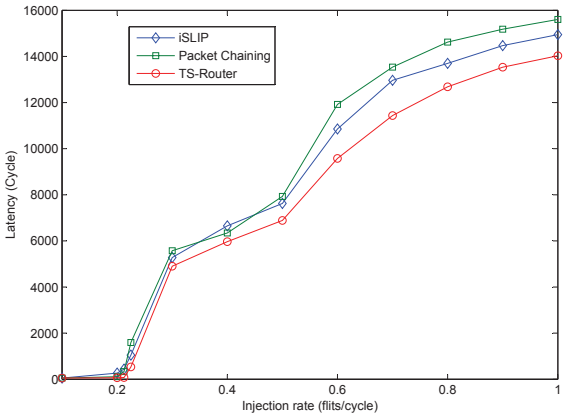


Fig. 8: The latency comparison under bit-complement traffic with different injection rates.

A. Evaluation for Network Performance

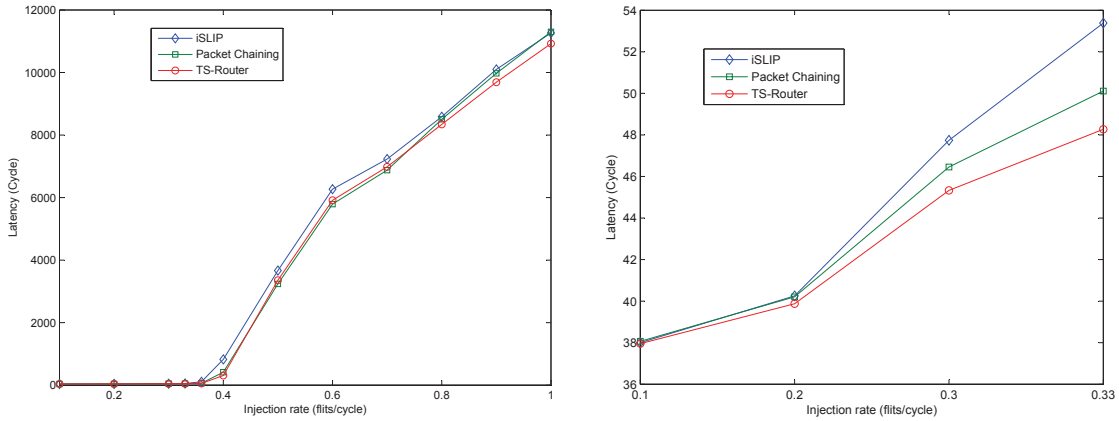
In this subsection, we investigate the synthetic traffic patterns with different injection rates to observe the performances of network latencies by iSLIP, Packet Chaining, and TS-Router, respectively. Note that in the following experiments we implement the second type of Packet Chaining (Same port, different VCs) since it has been shown that it strikes the best tradeoff between implementation overhead and performance gain [21].

c) Network latency of synthetic traffic patterns.: Under tornado and bit-complement traffic, TS-Router can have the better performance than the other two allocators. This is because that a high queueing latency may be incurred in Packet Chaining due to the reservation of the previous allocation. This phenomenon is due to *starvation* which is discussed in Section V-B3. In Figure 7 and Figure 8, TS-Router has a 76% lower average latencies compared to the other two allocators at the saturation point. Figure 9a shows the network latencies under uniform distribution traffic. TS-Router outperforms the other two allocators except while the injection rate is between 0.5 and 0.7 due to too many future requests involved, which will be further discussed in the section V-B1.

As the injecting rate increases, the network latency increases exponentially due to the network saturation. Therefore, it is difficult to observe that TS-Router outperform the other two allocators in low injection rates. We zoomed in Figure 9a before the injection rate achieves the saturation point ($IR \leq 0.33$) under uniform traffic pattern. As depicted in Figure 9b, it shows that TS-Router has lower latency when the network is not saturated. The same trends can be found in all the synthetic workloads when the injection rate is low. This property is important since most parallel programs run with the injection rate before saturation point according to [12], [25], and therefore achieving low latency before the saturation point is quite critical for most programs.

B. Evaluation for Time-Series Switching

1) Number of Matchings: In the following two experiments, we investigate the effectiveness of time-series consideration. We use three different synthetic traffic distribution with different injection rates to see the total resulted numbers of matchings by iSLIP, Packet Chaining, and TS-Router, respectively. As Figure 10 and Figure 11 show, the X-axis is the injection



(a) Injection rates with saturation

(b) Injection rates without saturation (Zoom-in)

Fig. 9: The latency comparison under uniform traffic with different injection rates.

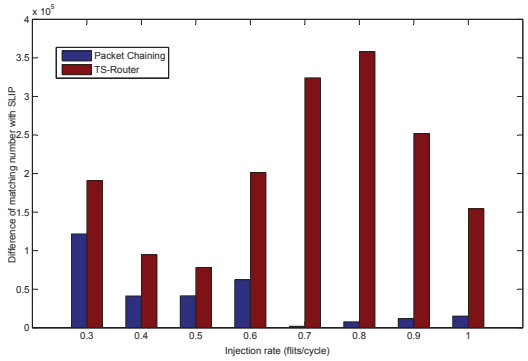


Fig. 10: The improved numbers of matchings by Packet Chaining and TS-Router under tornado distribution traffic.

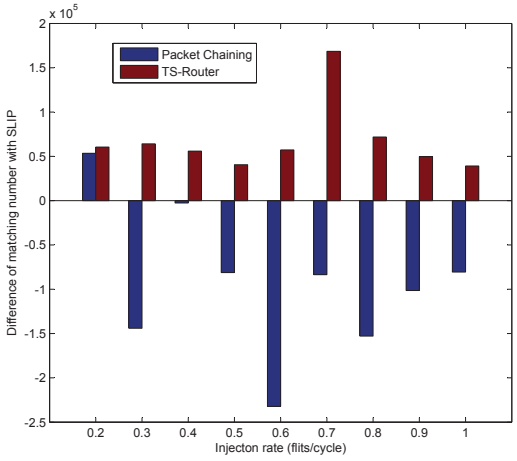


Fig. 11: The improved numbers of matchings by Packet Chaining and TS-Router under bit-complement distribution traffic.

rate (flits/cycle), and the Y-axis is the difference of matching numbers, and the results show that TS-Router outperforms Packet Chaining. In addition, the total number of allocation of Packet Chaining is worse than the baseline (iSLIP-1). This is because the traffic pattern of bit-complement is relatively more stable than the other two synthetic traffic, and the existing allocation is always prioritized than the newcomers. The resulted network behaves as a pseudo circuit-switch network and starve the newcomers, as we observed in the network latency. Further discussion is in Section V-B3.

However, Packet Chaining can take advantages of inheriting the pervious allocation while the injection is larger than 0.4 in uniform traffic, as illustrated in Figure 12. We observe that the allocation of TS-Router may behave similarly as iSLIP while the network becomes congested in uniform pattern. This is because that the priorities of the requests are almost the same since too many future requests, which are generated in uniform pattern with high injection rate, participate in the priority propagation. Nevertheless, it cannot happen in tornado and bit-complement traffic since the requests can be only generated in some inputs and outputs in these two traffic patterns.

2) Analysis of time series: To further analyze the effectiveness of time-series consideration, we focus on the saturation point (IR=0.4) and use *moving average*, which is a common technique in statistics for analyzing the trend in time series. We compared TS-Router with Packet Chaining since both of them are based on the observation that consecutive allocations affect each other.

For the readability, we sampled 500 cycles of the router in the center of the network. However, the following observation can also be found when sampling other routers with longer period. In each cycle the number of matchings is recorded, denoted as m_{pc} and m_{ts} for Packet Chaining and TS-Router, respectively. As Figure 13 shows, the Y-axis is the difference between the matching number, i.e., $m_{ts} - m_{pc}$, and the X-axis is the cycles. The light gray lines shows the fluctuation and it is hard to observe the trend. However, by applying moving average with period 10 (the black lines), the effects of time-series can be easily observed. As Figure 13 shows, the black lines are above the X-axis and thus positive at most time, which means considering the time-series effect, TS-Router outperforms Packet Chaining. With larger period, the effects are more obvious. Similar trends are found under tornado and bit-complement traffic distribution, as shown

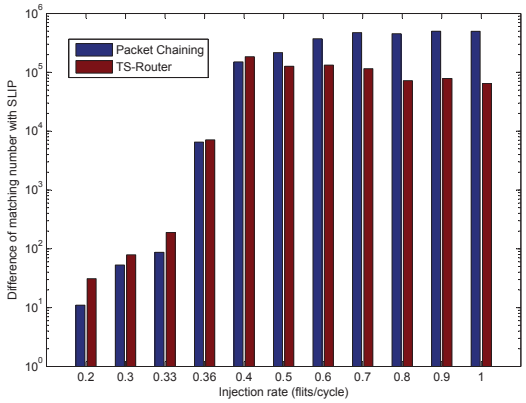


Fig. 12: The improved numbers of matchings by Packet Chaining and TS-Router under uniform distribution traffic.

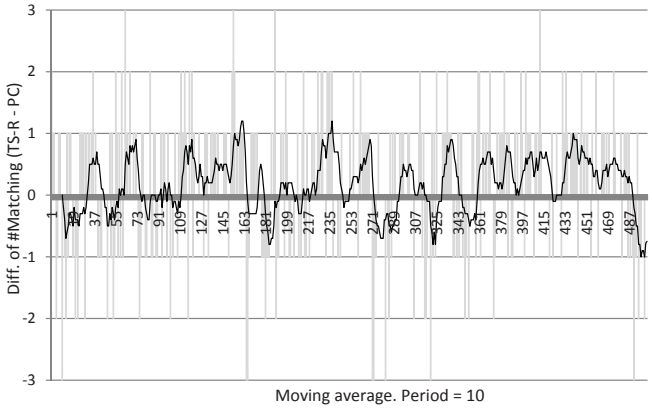


Fig. 13: The moving average for the difference of matching numbers between Packet Chaining and TS-Router under uniform distribution.

in Figure 14 and Figure 15.

The above two experiments conclude that TS-Router's *look-ahead* strategy is more effective than Packet Chaining's *look-behind* strategy. This is because look-behind is empirical inferring from past allocation to current allocation. In contrast, TS-Router accurately predicts the requestors in the following allocations and uses the future requestors as the clues to do the priorities assignment in current allocation, which is intrinsically time-series-considered rather than empirical inferring.

3) Starvation effects.: In this experiment, we compare the starvation effect when applying Packet Chaining and TS-Router. Since Packet Chaining is based on the assumption that the previous allocation is similar to the current allocation, the existing granted requests are prioritized to avoid joining the current allocation, which leads to the possibility of starving the new requests. For evaluating this affection, we compare Packet Chaining and TS-Router by increasing the packet length under the bit-complement traffic. As Figure 16 shows, when the packet length is between 1 and 3, Packet Chaining and TS-Router have the same performance in terms of the total number of matchings. When the packet length is between 4 and 5, TS-Router outperforms Packet

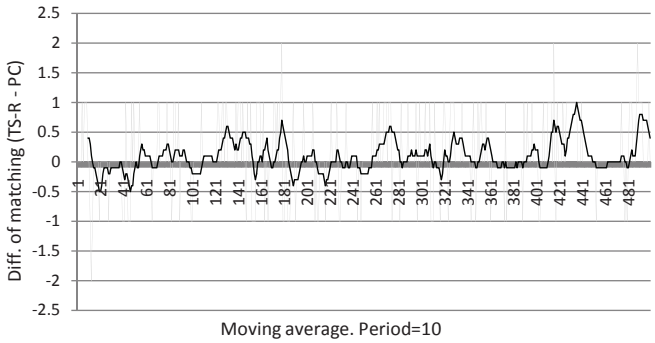


Fig. 14: The moving average for the difference of matching numbers between Packet Chaining and TS-Router under tornado traffic.

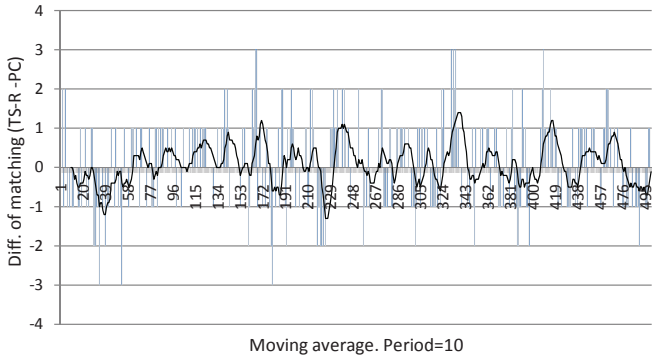


Fig. 15: The moving average for the difference of matching numbers between Packet Chaining and TS-Router under bit-complement traffic.

Chaining because the former leads to larger network capacity. Note that the network is getting saturated when the packet length is larger than 5. Unfortunately, when the packet length continues to increase, Packet Chaining inherits the previous allocation and the hit rate is very high due to the longer packet length. It makes the network behaves as a pseudo circuit-switch network, and the inherited allocations starve the new requests, which lead to exponential queuing delay in the network. Although the starvation effect can be avoided by setting a predefined value to *cut* the chain, i.e., limit the maximum number of chained flits, to avoid the starvation effect. However, it is related to the traffic behavior and therefore finding a suitable value for various traffic is not even possible.

Nevertheless, TS-Router relies on the predicted allocation rather than the existing ones, and therefore it does not starve the new requests. The priority is assigned according to whether or not more parallel connections can be granted. To conclude, TS-Router is more general and independent of the packet length, which leads to a starvation-free network.

C. Benchmark Evaluation

In this experiment, we configure GEM5 as a ALPHA CMP with 64 CPUs which is connected by an 8×8 mesh network.

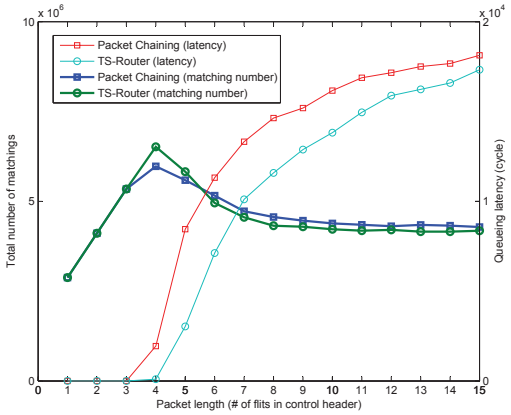


Fig. 16: The comparisons of total number of matchings and queueing latency between Packet Chaining and TS-Router under bit-complement traffic.

Each CPU is the out-of-order architecture and only supports single thread. The related setting is listed at Table I. We use nine programs from SPEC2006 [11] benchmarks. For each evaluation, the benchmark program is duplicated to 64 instances and distributed to the corresponding 64 cores as one-to-one mapping. The programs on each CPU run independently and generate the cache traffic, and the cache controller follows the cache protocols to generate packets into the three independent networks as *LD* (load) network, *IFETCH* (instruction fetch) network, and *ST* (store) network according to the packet type.

As Figure 17a shows, the network latency of TS-Router in LD network can be improved up to 8.5% compared with Packet Chaining. Also, we calculate the Instruction-Per-Cycle (IPC) when using TS-Router and Packet Chaining. As Figure 17b shows, TS-Router outperforms Packet Chaining in all the benchmarks, and the improvement is 2% on average.

VI. Related Works

To maximize the number of matchings in allocation, previous works such as Wavefront and Augmenting Path allocators can achieve maximal matchings and maximum matchings [26], [9], respectively. However, the hardware cost is much larger and therefore not practical in real implementations [13]. iSLIP separate allocators [20], which arbitrates inport and outport separately with round-robin priority, keeps the allocation simple but sacrifices the efficiency of matching. These allocators focus on each allocation independently and therefore can only achieve local optimal solutions.

In [29], the out-of-order requests to DRAM caused by NoC is discussed. In this paper, they propose the idea of *hold grant* which reduces the interleaving memory requests to keep the row-buffer hit-rate in the memory controller. Holding grants allows previously granted input to have higher priority than others and thereby avoids the locality of memory request being broken.

Different from the viewpoint of improving the efficiency of memory access, Pseudo-Circuit and Packet Chaining are more from the viewpoint of interconnection design. Both of them take advantages of the most recent allocation based on the observation that the last allocation tends to be similar to the current allocation [21], [2], so the last allocation is kept which results a circuit-switch-like behavior. The experiment results show that the performance can outperform Wavefront and Augmenting Path allocators due to the implicit time-series consideration. TS-Router shares similar concept, i.e., consecutive allocations, but with the explicit time-series consideration in the allocation algorithm. Instead of inheriting the past allocation which benefits the existing connections, TS-Router predicts the following requests for exploring more parallel connections between current allocation and next allocation. Due to the native time-series consideration, TS-Router outperforms these history-based allocators.

Other research works have explored the related problem with different points of views, such as running pipeline stage in parallel by speculation [24], [23], bypassing pipeline stages or having express channel for prioritized packets [15], [19], [18], and speed up the routing latency by looking-ahead design [10]. Most of these works are orthogonal to TS-Router. In other words, the concept of TS-Router can be implemented in these state-of-the-art routers and the existing advantages can be kept simultaneously.

VII. Conclusion and Future Works

In this paper, we first summarize the state-of-the-art works and TS-Router with our time-series model, which includes the past allocation, current allocation, as well as next allocation. Next, we propose TS-Router, a time-series effects considered router, which leverages the forwarding information from the previous pipeline stage to make a foresighted arbitration. By the foresighted arbitrations, the numbers of parallel connections (pairs of inputs and outputs) of the routers can be increased and also the whole system performance.

In the future work, we will apply TS-Router to many-core accelerator and hybrid design, such as the interconnection for GPU and Multi-Processor System-on-Chip (MPSoC). With more various processing elements interconnected by NoC, more factors are necessary to be explored, such as the impact of different traffic patterns, the different topologies, and the performance of high-radix TS-Router when being the communication fabric for the large-scale chip multiprocessors [8], [14], [22].

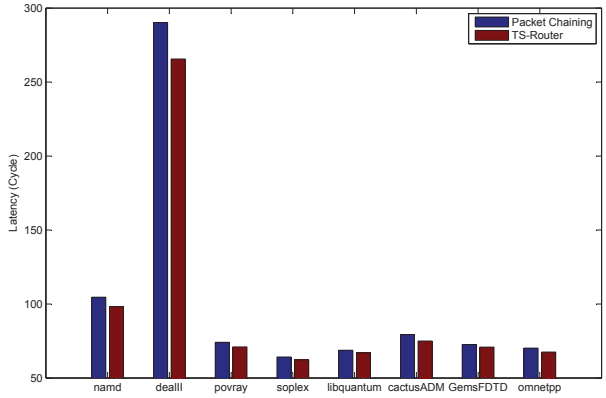
VIII. Acknowledgements

This work was supported in part by NSF grants 1205618, 1213052, 1147388, 0916887, 0905365 and 0903432 as well as Industrial Technology Research Institute and National Science Council grant NSC 101-2220-E-007-025.

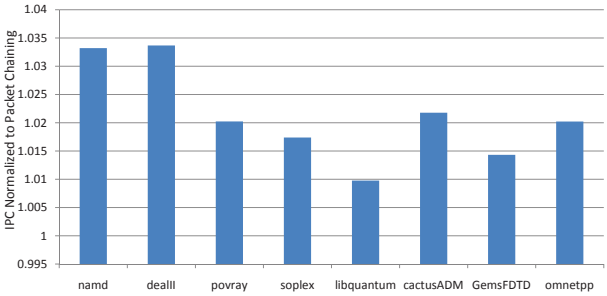
References

[1] N. Agarwal, T. Krishna, L. Peh, and N. Jha. Garnet: A detailed on-chip network model inside a full-system simulator. In *Performance Analysis of Systems and Software, 2009. ISPASS 2009. IEEE International Symposium on*, pages 33–42. Ieee, 2009.

[2] M. Ahn and E. J. Kim. Pseudo-circuit: Accelerating communication for on-chip interconnection networks. In *Proceedings of the 2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO '43, pages 399–408, Washington, DC, USA, 2010. IEEE Computer Society.



(a) The latency comparison between TS-Router and Packet Chaining in LD network.



(b) The IPC performance of SPEC CPU2006 (Normalized to Packet Chaining).

Fig. 17: The evaluation of network latency and IPC of SPEC CPU2006 in mesh network topology.

[3] D. U. Becker and W. J. Dally. Allocator implementations for network-on-chip routers. In *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, SC '09, pages 52:1–52:12, New York, NY, USA, 2009. ACM.

[4] N. Binkert, B. Beckmann, G. Black, S. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. Hower, T. Krishna, S. Sardashti, et al. The gem5 simulator. *ACM SIGARCH Computer Architecture News*, 39(2):1–7, 2011.

[5] W. Dally. Virtual-channel flow control. *Parallel and Distributed Systems, IEEE Transactions on*, 3(2):194–205, 1992.

[6] W. Dally and B. Towles. Route packets, not wires: On-chip interconnection networks. In *Design Automation Conference, 2001. Proceedings*, pages 684–689. IEEE, 2001.

[7] W. Dally and B. Towles. *Principles and practices of interconnection networks*. Morgan Kaufmann, 2004.

[8] H. Eberle, P. Garcia, J. Flich, J. Duato, R. Drost, N. Gura, D. Hopkins, and W. Olesinski. High-radix crossbar switches enabled by proximity communication. In *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, page 32. IEEE Press, 2008.

[9] L. Ford and D. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8(3):399–404, 1956.

[10] M. Galles. Spider: a high-speed network interconnect. *Micro, IEEE*, 17(1):34–39, 1997.

[11] J. L. Henning. Spec cpu2006 benchmark descriptions. *SIGARCH Comput. Archit. News*, 34(4):1–17, Sept. 2006.

[12] J. Hestness, B. Grot, and S. Keckler. Netrace: dependency-driven trace-based network-on-chip simulation. In *Proceedings of the Third International Workshop on Network on Chip Architectures*, pages 31–36. ACM, 2010.

[13] R. Hoare, Z. Ding, and A. Jones. A near-optimal real-time hardware scheduler for large cardinality crossbar switches. In *Proceedings of the 2006 ACM/IEEE Conference on Supercomputing*, page 94. ACM, 2006.

[14] J. Kim, W. Dally, B. Towles, and A. Gupta. Microarchitecture of a high-radix router. In *ACM SIGARCH Computer Architecture News*, volume 33, pages 420–431. IEEE Computer Society, 2005.

[15] A. Kumar, L.-S. Peh, P. Kundu, and N. K. Jha. Express virtual channels: towards the ideal interconnection fabric. In *Proceedings of the 34th annual international symposium on Computer architecture*, ISCA '07, pages 150–161, New York, NY, USA, 2007. ACM.

[16] R. Kumar, V. Zyuban, and D. Tullsen. Interconnections in multi-core architectures: Understanding mechanisms, overheads and scaling. In *Computer Architecture, 2005. ISCA'05. Proceedings. 32nd International Symposium on*, pages 408–419. IEEE, 2005.

[17] M. M. K. Martin, D. J. Sorin, B. M. Beckmann, M. R. Marty, M. Xu, A. R. Alameldeen, K. E. Moore, M. D. Hill, and D. A. Wood. Multifacet's general execution-driven multiprocessor simulator (gems) toolset. *SIGARCH Comput. Archit. News*, 33(4):92–99, Nov. 2005.

[18] H. Matsutani, M. Koibuchi, H. Amano, and T. Yoshinaga. Prediction router: Yet another low latency on-chip router architecture. In *High Performance Computer Architecture, 2009. HPCA 2009. IEEE 15th International Symposium on*, pages 367–378. IEEE, 2009.

[19] H. Matsutani, M. Koibuchi, H. Amano, and T. Yoshinaga. Prediction router: A low-latency on-chip router architecture with multiple predictors. volume 60, pages 783–799. IEEE, 2011.

[20] N. McKeown. The islip scheduling algorithm for input-queued switches. *Networking, IEEE/ACM Transactions on*, 7(2):188–201, 1999.

[21] G. Michelogiannakis, N. Jiang, D. Becker, and W. J. Dally. Packet chaining: efficient single-cycle allocation for on-chip networks. In *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-44 '11, pages 83–94, New York, NY, USA, 2011. ACM.

[22] C. Minkenberg and M. Gusat. Speculative flow control for high-radix datacenter interconnect routers. In *Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International*, pages 1–10. IEEE, 2007.

[23] R. Mullins, A. West, and S. Moore. Low-latency virtual-channel routers for on-chip networks. In *Proceedings of the 31st annual international symposium on Computer architecture*, ISCA '04, pages 188–, Washington, DC, USA, 2004. IEEE Computer Society.

[24] L.-S. Peh and W. Dally. A delay model and speculative architecture for pipelined routers. In *High-Performance Computer Architecture, 2001. HPCA. The Seventh International Symposium on*, pages 255 –266, 2001.

[25] D. Sanchez, G. Michelogiannakis, and C. Kozyrakis. An analysis of on-chip interconnection networks for large-scale chip multiprocessors. *ACM Transactions on Architecture and Code Optimization (TACO)*, 7(1):4, 2010.

[26] Y. Tamir and H. C. Chi. Symmetric crossbar arbiters for vlsi communication switches. *IEEE Trans. Parallel Distrib. Syst.*, 4(1):13–27, Jan. 1993.

[27] H. Wang, L. Peh, and S. Malik. Power-driven design of router microarchitectures in on-chip networks. In *Proceedings of the 36th annual IEEE/ACM International Symposium on Microarchitecture*, page 105. IEEE Computer Society, 2003.

[28] H. Wang, X. Zhu, L. Peh, and S. Malik. Orion: a power-performance simulator for interconnection networks. In *Microarchitecture, 2002.(MICRO-35). Proceedings. 35th Annual IEEE/ACM International Symposium on*, pages 294–305. IEEE, 2002.

[29] G. Yuan, A. Bakhoda, and T. Aamodt. Complexity effective memory access scheduling for many-core accelerator architectures. In *Microarchitecture, 2009. MICRO-42. 42nd Annual IEEE/ACM International Symposium on*, pages 34–44. IEEE, 2009.