A Fair and Maximal Allocator for Single-Cycle On-Chip Homogeneous Resource Allocation

Shaoteng Liu, Axel Jantsch, and Zhonghai Lu

***Abstract*— Traditional allocators for network-on-chip (NoC) routers suffer from either poor-matching quality or limited fairness. We propose a waterfall (WTF) allocator targeting homogeneous resource allocation, which provides single-cycle maximal matching while guaranteeing strong fairness based on the round-robin principle. It can be implemented with a loop-free structure. In 90 nm technology, the allocator operates at about 1 GHz clock frequency. We compare WTF with wave-front, separable-input-first, and separable-output-first allocators and find that it is at least 10% smaller, has 50% less delay under high load, and uses 3% less power than any of these alternatives. Also, WTF is at least as fair or clearly fairer. We also find that in a 4 × 4 circuit switched NoC the use of WTF gives up to 20% higher network performance.**

***Index Terms*— Allocator, fairness, maximal matching, network-on-chip (NoC), round-robin.**

I. Introduction

An allocator performs a matching between multiple resources and multiple requesters. A matching is an assignment of resources to requesters satisfying the following three constraints [1]: 1) a resource is granted to a requester only if the corresponding request exists; 2) each resource is granted to at most one requester; 3) a requester is at most granted once. A matching in which no additional requests can be served without removing one of the existing grants is called a maximal matching [2] and the one containing the maximum number of grants is called a maximum matching. Maximum matching is often too costly to be realized in hardware. However, maximal matching is achievable. In addition to matching quality, fairness is an important property for an allocator, and we can distinguish between strong fairness and weak fairness [2]. Intuitively, strong fairness guarantees that all requesters are served in proportion to their relative request rates. In practice, this means that persistently active requesters are served in a periodic sequence equally often within each reasonably short period. In contrast, weak fairness only requires that every request is eventually granted, without any guarantee at which rate or in which relative proportion different requesters are served.

Allocators used in network-on-chip (NoC) routers have limitations. Compared to large scale networks, the performance of NoCs is more sensitive to the latency of each router. This mandates the use of single cycle allocators in router design [2]. Consequently, conventional allocators in NoCs do not take into account the maximal matching quality and strong fairness at the same time. Strong fairness is usually provided by using a variation of the round-robin principle, which states that a request that was just served should have the lowest priority in the next round [1]. On one hand, allocators which adopt round-robin cannot ensure maximal matching. Separable-input-first (SIF) and separable-output-first (SOF) allocators [2], [3] are classified in this category. On the other hand, maximal matching allocators, such as wave-front (WVF) [4]–[6] or rectilinear-propagation-arbiter and diagonal-propagation-arbiter [7], do not use the round-robin principle and only provide week fairness. For general matching problems, no ideal solution is known to design a NoC allocator overcoming all these shortcomings.

Manuscript received January 1, 2013; revised June 26, 2013; accepted September 9, 2013.

The authors are with the Department of Electronic Systems, KTH Royal Institute of Technology, Stockholm 16440, Sweden (e-mail: liu2@kth.se; axel@kth.se; zhonghai@kth.se).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVLSI.2013.2284563

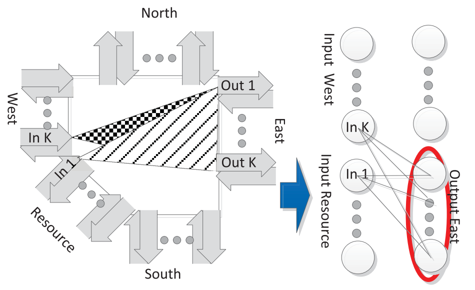


Fig. 1. Illustration of homogeneous allocation in NoC.

However, in NoC design practice, we frequently encounter a kind of special matching problem called homogeneous resources allocation (HRA). Here, homogeneous resources refer to a class of resources that have the same functionality and can be used interchangeably. This kind of matching problem obeys two more constraints besides the three constraints introduced above: 4) for each requester, all resources it desires belong to the same class; and 5) any resource of a class can be granted to the requester who have requests on this class.

We use a router model to illustrate the HRA problem. The router in Fig. 1 has five directions and each direction contains k-duplexed channels. Channels in this model can either be regarded as virtual channels (VCs) that share one physical channel in a time division multiplexing way, or subphysical channels by splitting the wires of a link in an spacial division multiplexing (SDM) way. The output channels of each direction form one resource class. The routing algorithm is deterministic, e.g., dimension-order-routing, which assigns each arrival packet one and only one desired output direction. For example, in Fig. 1, both packets from input channel k of the west and input channel 1 of the resource desire an output channel to the east. The right part of Fig. 1 is the bipartite graph of this case with each line representing a request. Since all requests of a packet are confined to output channels of the east direction, constraint 4) holds, and because each packet asserts requests on every output channel of the east direction, constraint 5) can be satisfied. Therefore, channel allocation inside such a router is an HRA problem.

For HRA, we propose a single cycle allocator which guarantees both maximal matching and strong fairness. We call it "water-fall" (WTF) because it finds the matching in several consecutive steps. For an n-requester allocator, it requires n steps. WTF is implemented entirely as combinational logic, which means the allocation takes one cycle. It can be implemented free of combinational loops that are common in traditional maximal allocators, e.g., the wave-front allocator in [4]. We develop a fairness policy which inherits the principle of round-robin and name it as massive round-robin (MRR) for HRA.

HRA is an abstraction of a class of allocation problems which are frequently encountered in NoC designs. For example, either VC allocation in wormhole-based packet switched NoC [3] or subchannel allocation in circuited switched NoC using SDM [8] is an HRA problem. However, aside from our examples, we believe that our allocator can be further utilized by other kinds of on-chip applications which have HRA problems, even beyond the scope of NoC usage.

II. Design of the Allocator

A. Basic Structure of the Allocator

Intuitively WTF reduces the request matrix (rows in this matrix represent requesters and columns represent resources) by taking advantage of homogeneous resources. As the matrix in Fig. 2 suggests, input channels 0, 1, 3 each request a channel from output direction 1. Input channel 2 requests a channel from output direction 2. Each input packet just has one desirable output direction, thus constraints 4) and 5) hold and it is an HRA problem. For an HRA problem, first, the big matrix can be split into submatrices according to resource classes (output directions). Secondly, we can merge the requests of a submatrix. As described in Fig. 2, the 4 × 4 matrix is split up into two separate 4 × 1 matrices representing requests for output direction 1 and 2, respectively.

Continuing the example, we apply two separate homogeneous resource allocators to solve the two reduced matrices. We use a ripple carry arbitration scheme to design such an allocator. Taking the allocation in output direction 1 as an example, as described in Fig. 3(a), since it has two resources and four requesters (marked as 2 → 4), the allocator is made up of two columns and four rows of arbitration cells accordingly. The three active requesters are indicated by "1" in the reduced matrix. Two tokens are used to denote the availability and grant decisions of the two output channels (resources), respectively. In the current round, the arbitration for tokens starts from the row 2, moves counter-clock wise, and ends at row 1. Thus, the 4 requesters are served in this order: (r_2, r_3, r_0, r_1). r_3 will be the first to catch a token and r_0 the second. This means channel 1 is granted to r_3 and channel 2 is granted to r_0.

Considering fairness, we need to roll the start row. Our MRR fairness policy is illustrated in Fig. 3(a) and (b). The principle is that the end row in the next round is the last successful requester of the previous round. And the start row is acquired by incrementing the end row by 1, then modulo n. If no requester is granted, the start row remains the same as in the previous round. Applying this policy, active requesters r_0, r_1, r_3 in Fig. 3 (assuming they are persistently active) are granted in the periodic sequence: $\{(r_3, r_0)_{\text{round0}}, (r_1, r_3)_{\text{round1}}, (r_0, r_1)_{\text{round2}}, (r_3, r_0)\ldots\}$. The start row i is selected by asserting $p_i = 1, 0 \leq i \leq n-1$, where n is the number of requesters.

In general, we derive the function of the MRR policy as follows. Given current grants g_i, $0 \leq i \leq n-1$ for each requester ($g_i = 1$ means granted), and suppose G is the set of granted requesters of the current round with $G = \{x | 0 \leq x \leq n-1, \ g_x = 1\}$ and b denotes any granted requester that $b \in G$, and the current start row is $k_{(t)}$ and the end row is f. Compute the start row of the next round $k_{(t+1)}$ as follows:

$$\text{(if } \exists f \text{ that)}$$
$$\begin{cases} (f + n - k_{(t)})_{\text{mod}n} = \max\{(b + n - k_{(t)})_{\text{mod}n}\} \\ f \in G \end{cases}$$
$$\text{(then } k_{(t+1)} = (f+1)_{\text{mod}n}, \text{ otherwise } k_{(t+1)} = k_{(t)}).$$

Although in WTF there is no actual logic feedback, we also need to avoid combinational circuits with loops which are undesirable due to issues in verification and testing [7]. In Fig. 3(c), we propose a loop-free structure by adding redundant logic. For an n-requester allocator, our loop-free structure contains $2n-1$ rows. The bottom $n-1$ rows replicate the top $n-1$ rows. In this way, rolling of the start row is equivalent to selecting an active area. Fig. 3(c) shows how to convert a loop structure into a loop-free structure. For example, suppose all requesters are served in the order (r_1, r_2, r_3, r_0) in the current round. Mapping into the loop-free structure, the area from row 1 to row 5 is activated, as the rectangular box in Fig. 3(c) suggests. Since row 5 replicates row 0, the allocation is also in the order (r_1, r_2, r_3, r_0). In this way, the functions of the loop and loop-free structures are equivalent.

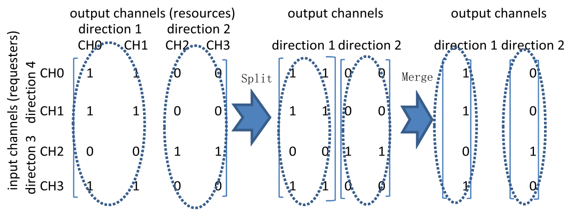


Fig. 2. Reduction of request matrix.

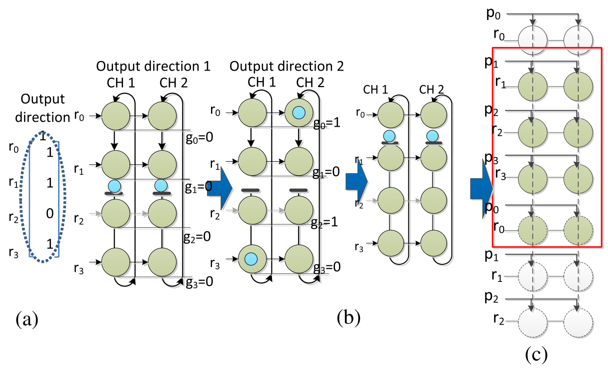


Fig. 3. Allocation mechanism, fairness policy and loop-free structure (circles represent arbitration cells, dots represent tokens. (a) Allocation in output direction 1. (b) Start row rotation. (c) Loop-free structure).

B. Implementation

A WTF allocator consists of two parts: allocation logic and priority updating logic. The allocation logic is used to generate grants of the current round. The priority updating logic is used to guarantee the fairness.

1) Implementation of the Allocation Logic: A loop-free design is described in Fig. 4(a). An n-requester m-resource allocator ($m \to n$) has $2n-1$ rows and m columns. The top n rows are made up of white cells, whereas the bottom $n-1$ rows are composed of dark cells. The right part of Fig. 4(a) depicts the internal logic of the two arbitration cells. A white cell plays a role as "token" starter when its priority p is asserted. It directly accepts the channel status as tokens. When p is de-asserted, its role is a "token" passer and it can receive a token passed by the upper cell from its north input. A request is injected from the W(west) input of a cell. When an arbitration cell receives a token and if it has one request asserted, it consumes the token and grants the request with $g_{ij} = 1$. Otherwise, it passes the token on from its south output. The role of a black cell is a token passer when p is de-asserted. When p is asserted, the black cell's function is that of a token terminator. In this situation, all its outputs are set to "0." Token passing ends at this cell.

Suppose p_i, $0 \leq i \leq n-1$, is set as "1," then the effective tokens are injected from row i and passed downward. Meanwhile, row $i+n$ in the dark region also gets its priority line p_i asserted and thus stops tokens passing. In such a way, the active region starts from row i and ends at row $i+n-1$.

2) Implementation of the Priority Updating Logic: The key point of the priority updating algorithm is to find the end row for the new round. This is a $1 \to n$ allocation problem. The n-bit grant vector generated by the allocation logic denotes requesters. The requester which gets the token will turn into the end row, as shown in Fig. 4(b). This time the token passing is clock-wise, which is opposite to the allocation logic. The start row is the same start row

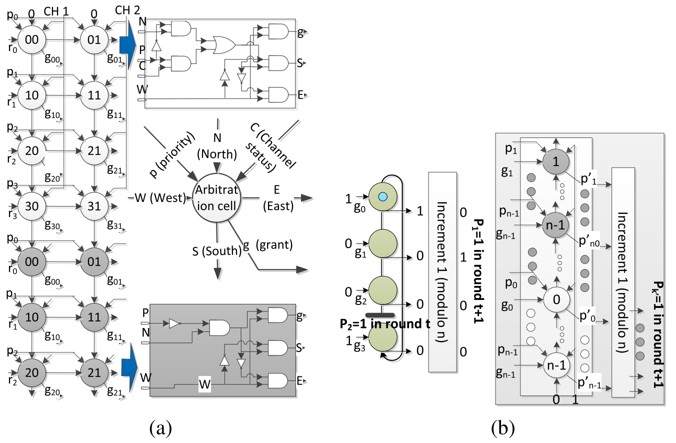


Fig. 4. Hardware implementation of the WTF allocator. (a) Allocation logic. (b) Priority updating logic.

used by allocation logic for the current round. For example, in a $(2 \rightarrow 4)$ allocator, suppose p_2 is asserted and thus the grant vector generated is 1001. For the priority updating logic, the token is also injected from row 2 but passed in clock-wise order. As a result, row 0 (g_0) catches the token. Then row 1 will be the start row in the next round (incrementing the end row by 1, then modulo n). The whole priority updating logic is implemented as illustrated in the right part of Fig. 4(b).

III. Evaluation and Analysis

It is known that fairness and performance are two key metrics for an allocator. We evaluate and analyze both aspects. The router model described in Fig. 1 is simulated. The arrival packets at each input channel are queued up. When an output channel is granted to an input channel, one packet leaves the queue. Each packet just needs 1 cycle to be delivered. The arrival time of packets obey a Poison process or an on-off process. The arrival packets are uniformly and randomly distributed among all input channels.

In our setting, each direction of the router contains four duplex-channels. Thus the channel allocation inside this router forms a $20 \rightarrow 20$ allocation problem. However, considering HRA, the $20 \rightarrow 20$ matrix can be split into five $4 \rightarrow 16$ submatrices, each of which represents the channel allocation of one output direction. Accordingly, we can assign each output direction an allocator. Since our purpose is to evaluate the fairness and performance of an allocator, in order to avoid influences such as head of line blocking, we assign every arrival packet the same target output direction.

In addition to the WTF allocator, we modeled several other allocators for comparison, including SIF allocator, SOF allocator, and WVF allocator. They are reported as representative allocators in NoC design in [2]. Both the two separable allocators adopt two stages of round-robin arbitration. The wave-front allocator adopts rotating policy [1], [9] by incrementing the priority each round. All allocators are tested under the same packet injection test bench for each utilization. Here utilization equals the duration of an output channel occupied by a packet (1 cycle in this case) multiplied by the sum of injection rates of all inputs and divided by the number of resources (4 in this case). We simulated 40 000 cycles at each utilization for each allocator.

A. Performance

Performance is affected by matching quality of an allocator. Fig. 5(a) shows the average delay in terms of cycles versus the

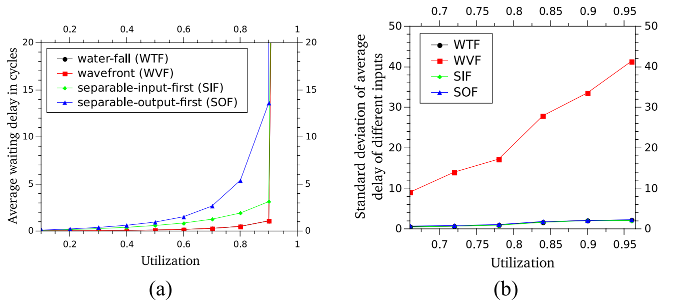


Fig. 5. Performance comparison between allocators [the waiting delay of a packet is the time that the packet waits in the input queue. The total delay (response time) of a packet equals the waiting delay plus one cycle]. (a) Average waiting delay in cycles (The curves of WVF and WTF are almost identical.) (b) Random case fairness. (The curves of WTF, SIF, and SOF are almost identical.)

TABLE I
SYNTHESIS RESULTS OF DIFFERENT ALLOCATORS

Structure	WTF	SIF	SOF	WVF
Area (um^2)	29990	33549	34196	120466
Delay (ns)	1.04	1.0	1.0	2.0
Power (mW)	9.2	9.5	11.2	9.6

utilization. It suggests that WTF exhibits the same average waiting delay in cycles as WVF. This is because both WVF and WTF are maximal allocators, they generate the same number of grants every cycle, thus the average packet delay is the same. The performances of the two separable allocators are worse than WTF and WVF because no maximal allocation guarantee is provided. At utilization 0.9, the waiting delay of WTF and WVF are both 1.2 cycles. And it is 3.1 cycles for SIF and 13.3 cycles for SOF. For separable allocators, in some cases, resources are left unassigned even in the presence of requests waiting for resources, and thus performances are degraded. Although the delay in cycles of WTF and WVF are very close, when the differences in clock frequency are considered, WTF has 50% of the actual delay in ns of WVF (WTF is about as twice fast as WVF, as Table I suggests.)

B. Fairness

In addition to lower performance, WVF is also less fair.

1) Example Study: Continuing our previous example shown in Fig. 3(a), for an $2 \rightarrow 4$ allocator, suppose r_0, r_1, r_3 is continuously asserted. For WTF, three active requesters are served in a periodic sequence (r_3, r_0), (r_1, r_3), (r_0, r_1), $(r_3, r_0)\ldots$. Each period has three allocation rounds. Inside a period, every requester is served twice. The normalized throughputs are (0.66, 0.66, 0.66), respectively. However, for WVF with priority rotating policy, the periodic sequence is (r_0, r_3), (r_1, r_0), (r_3, r_1), (r_3, r_0), $(r_0, r_3)\ldots$. Each period has four allocation rounds. Inside a period, r_0 and r_3 are served three times, but r_1 is only served two times. The throughputs are (0.75, 0.75, 0.5), respectively.

2) Worst Case Analysis: In general, we assume an allocator has n inputs and m resources ($m \leq n$). Suppose during a time interval (t_1, t_2), p requesters ($m \leq p \leq n$), numbering $r_1, r_2, \ldots, r_p$, are constantly active. Each requester will occupy the resource for L cycles when it is granted. Let $V_i(t_1, t_2)$ denote the amount of service received by requester i in (t_1, t_2).

Applying our MRR policy, we can ensure that between any two service opportunities given to requester r_i, requester r_j must have had an opportunity. Thus, suppose during (t_1, t_2), requester i gets

z service opportunities and requester j gets z', then $|\ z - z'\ | \leq 1$. Therefore

$$|V_i(t_1, t_2) - V_j(t_1, t_2)| = |zL - z'L| \leq |z - z'|L = L. \qquad (1)$$

As interval $(t_1, t_2) \rightarrow \infty$, the difference in amount of services of any two requesters is always bounded by L.

For the WVF, the p requesters are served in a periodic sequence and each period contains n allocation rounds. Thus, each period is nL cycles long. During every period, in the worst case, the most favorite requester r_i can have $m+n-p$ service opportunities, while the least served requester r_j just has m service opportunities. Suppose there are q periods during interval (t_1, t_2) that $q = \lfloor t_2 - t_1/nL \rfloor$, then

$$(m+n-p)(q-1)L \ \leq \ V_i(t_1, t_2) \leq (m+n-p)(q+1)L \qquad (2)$$

$$m(q-1)L \ \leq \ V_j(t_1, t_2) \leq m(q+1)L \qquad (3)$$

$$|V_i(t_1, t_2) - V_j(t_1, t_2)| \ \leq \ (n-p)qL + (2m+n-p)L. \qquad (4)$$

As interval $(t_1, t_2) \rightarrow \infty$ and $q \rightarrow \infty$, there is no worst case bound for WVF.

3) Random Case Study: The simulation model used for performance is used to study the stochastic fairness behavior.

Under certain traffic patterns, an unfair allocator may generate significant biased allocations. For example, as long as frequently active requesters are not evenly spaced among all requesters, the allocation may be unfair with WVF. We will take one of such traffic pattern for a study. The traffic pattern is set as follows: the packets are injected from only 12 of the total 16 inputs: from input 0, 1, ... to input 12. The packets injection process is an on-off process. This two-state Markov modulated process has probability α to switch from off to on, and a probability to β switch from on to off. In the on state, each input has the same probability r_1 of injecting a packet. While in the off state, no packets can be injected. Thus, the average injection rate of each input is $\alpha r_1/(\alpha + \beta)$.

Under a certain utilization, the average packet waiting time of input queue i is marked as $D_i, 0 \leq i \leq 12$. We use standard deviation of $D_i, 0 \leq i \leq 12$ [denoted as $\sigma(D)$] as a metric of fairness. As we can imagine, unfair allocators will result in significant variance of these D_i values. Therefore, its $\sigma(D)$ should be higher than that of fair allocators.

Fig. 5(b) suggests the $\sigma(D)$ values of several allocators under different utilizations. Generally speaking, under every utilization, the value of WVF is much higher than the others. For example, at utilization 0.78, the $\sigma(D)$ of WVF is 17.2, while that of WTF is only 1.0, SIF is 0.90 and SOF is 1.1. This means that WVF does not treat every input queue fairly. In other words, certain inputs are served more often than the others.

C. Synthesis Results

We synthesize the allocators used in the router model with TSMC 90 nm technology with Synopsys Design Compiler. The results are listed in Table I. The power numbers are obtained by assuming 50% switching activity of each input signal. Note that the WVF used for synthesis also has a loop-free implementation by replicating the array for each possible priority diagonal and selecting the grant matrix generated by the currently active one. For details of this WVF allocator implementation, refer to [2].

We find that our WTF allocator is slightly slower than SIF and SOF, but it consumes less area and power. It is much faster than WVF. This is because WVF has to be square. In this case, it is more efficient to be implemented as one $20 \rightarrow 20$ allocator rather than five $4 \rightarrow 16$ allocators. As a result, its critical path is longer than WTF. Besides, since the WVF avoids combinational loops by replicating the entire 20×20 array for each priority, it consumes much more area than other allocators.

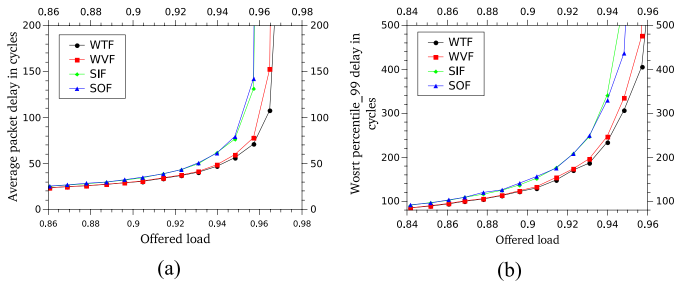


Fig. 6. Performance comparison in circuit switched NoC in cycles. (a) Average delay in cycles. (b) Percentile delay in cycles.

IV. Application Study

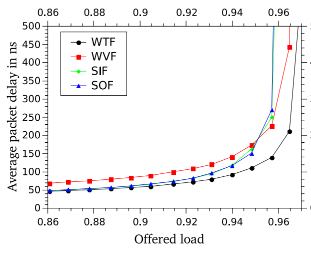
Our allocator is used in a circuit switched NoC design. The circuit switched NoC consists of 4×4 routers and adopts a mesh topology. Inside each router, every direction has four-duplexed SDM subchannels. A parallel probing method [10] is used for path set-up. Probes are used to set-up a minimal path for data transfer. At the beginning, one probe carrying a set-up request is sent out by the source node. At each hop, when a probe enters into a router, it can split into multiple probes if it has multiple preferable output directions. As probes travel, they will reserve the output channels which they have been allocated inside each router for future data transfer. Whenever two probes carrying the same request meet, one of them is regarded as redundant and is cancelled and all channels used only by the cancelled probe are released. Each probe takes two cycles for a hop. For detailed description of this router architecture and set-up method, refer to [10].

Since each probe is assigned only one target direction, allocating output channels to probes is an HRA problem. For evaluation, uniform random traffic is applied. When a path is established, a packet with eight flits is delivered. After data transfer the path is released.

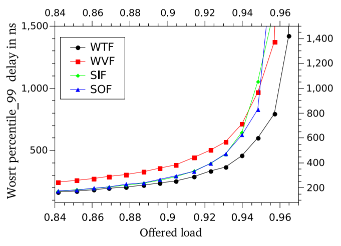
The results are shown in Fig. 6. Fig. 6(a) gives the average packet delay in cycles versus offered load, by assuming that routers with different allocators are working at the same clock frequency. Offered load refers to the time used for data transfer of a path (8 cycles in this case) multiplied by per node set-up request injection rate. In this case, WTF has the best performance, e.g., at offered load 0.95, the average packet delay by using WTF is 56 cycles. By using WVF, it is 59 cycles. For SIF and SOF, they are 76 and 79 cycles, respectively. Although WTF and WVF are both maximal allocators, the unfairness of WVF might cause unbalanced congestion of channels. In this situation, some channels become saturated earlier than others. Thus, we observe that the performance of WVF is slightly inferior to WTF. This difference will be enlarged at high load, e.g., at load 0.96, the average delay of WTF is 107 cycles while it is 152 cycles for WVF.

We also compared their worst packet delay by using a statistical method. We use percentile as a measurement. For example, Percentile(n) is the minimum delay of the $(1-n)\%$ packets that experience longest delays in our simulation.[1] In Fig. 6(b), the percentile(99) delay of WTF is always smaller than WVF. For example, at load 0.95, percentile(99) delay of WTF is 307 cycles, and it is

[1]We choose this metric because it is less susceptible to statistical abnormalities [6]. Each percentile value is taken from a large amount of samples that come from 2 million simulation cycles. Thus, the upper and lower bound of the confidence interval of a percentile value are believed to be very close. For example, at load 0.95, for WTF, we ran several simulations with different random seeds and total simulation cycles, ranging from 250 thousand to 5 million cycles. The percentile(99) values of each simulation were all around 307, with less than 1% difference.



(a)



(b)

Fig. 7. Performance comparison in circuit switched NoC in ns. (a) Average delay in ns. (b) Percentile delay in ns.

335 cycles for WVF. At load 0.96, the WTF is 406 cycles, and it is 476 cycles for WVF. This result also support that WTF is fairer than WVF. We also notice that the percentile(99) delay curves of SIF and SOF are worse than WVF and WTF, because their performances cannot catch up with maximal allocators such as WVF and WTF.

In Fig. 7, we evaluate the influences of using different allocators on the critical timing path, which mainly consists of one allocator latency plus one crossbar latency. In this case, a router with a WTF allocator can work at 510 MHz clock frequency. A router with SOF or SIF can work at 526 MHz. And a router with WVF operates at 345 MHz. As a result, we can measure average packet delay in ns. In Fig. 7(a) we find that the WTF has the best performance and WVF the worst. At load 0.94, the average packet delay is 92 ns for WTF. And it is 117 and 116 ns for SIF and SOF, respectively. For WVF, it is 141 ns. We also measure percentile(99) of the worst case delay in ns, as shown in Fig. 7(b). For example, at load 0.93, for WTF, it is 366 ns. For SIF and SOF, it is 469 and 473, respectively. It is 569 ns for WVF. Hence, WTF is superior to the three alternatives.

V. Conclusion

Matching quality and fairness are two important concerns for designing an allocator. While achieving one or the other, traditional allocators for NoCs fail to succeed in both aspects. In this brief, we propose an allocator called WTF for homogeneous resource allocation. This allocator guarantees both maximal matching quality and strong fairness. Furthermore, our allocator can be implemented in hardware without combinational loops. The abilities in performance and fairness of our allocator are analyzed and demonstrated in simulation. We also use WTF in a 4×4 circuit switched NoC design. Experiment results suggest that WTF offers better performance and lower area than traditional allocators while achieving strong fairness.

References

[1] W. J. Dally and B. Towles, *Principles and Practices of Interconnection Networks* (The Morgan Kaufmann Series in Computer Architecture and Design). San Mateo, CA, USA: Morgan Kaufmann, Dec. 2003, pp. 351–375.

[2] D. Becker and W. Dally, "Allocator implementations for network-on-chip routers," in *Proc. Conf. High Perform. Comput. Netw., Storage Anal.*, 2009, pp. 1–12.

[3] S. Park, T. Krishna, C. Chen, B. Daya, A. Chandrakasan, and L. Peh, "Approaching the theoretical limits of a mesh NoC with a 16-node chip prototype in 45 nm SOI," in *Proc. 49th Annu. DAC*, 2012, pp. 398–405.

[4] J. Delgado-Frias and G. Ratanpal, "A VLSI crossbar switch with wrapped wave front arbitration," *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.*, vol. 50, no. 1, pp. 135–141, Jan. 2003.

[5] W. Olesinski, H. Eberle, and N. Gura, "PWWFA: The parallel wrapped wave front arbiter for large switches," in *Proc. Workshop HPSR*, Jun. 2007, pp. 1–6.

[6] Y. Tamir and H.-C. Chi, "Symmetric crossbar arbiters for VLSI communication switches," *IEEE Trans. Parallel Distrib. Syst.*, vol. 4, no. 1, pp. 13–27, Jan. 1993.

[7] J. Hurt, A. May, X. Zhu, and B. Lin, "Design and implementation of high-speed symmetric crossbar schedulers," in *Proc. IEEE ICC*. vol. 3. Jun. 1999, pp. 1478–1483.

[8] A. Lusala and J.-D. Legat, "Combining sdm-based circuit switching with packet switching in a NoC for real-time applications," in *Proc. IEEE ISCAS*, May 2011, pp. 2505–2508.

[9] H. Chi and Y. Tamir, "Decomposed arbiters for large crossbars with multi-queue input buffers," in *Proc. IEEE ICCD VLSI Comput. Processors*, Oct. 1991, pp. 233–238.

[10] S. Liu, A. Jantsch, and Z. Lu, "Parallel probing: Dynamic and constant time setup procedure in circuit switching NoC," in *Proc. IEEE DATE*, Mar. 2012, pp. 1289–1294.