Bandwidth-based Application-Aware Multipath Routing for NoCs

X.T. Ding
Institute of Artificial Intelligence and Robotics
Xi'an Jiaotong University
Shaanxi, China
State Key Laboratory of Mathematical Engineering and Advanced Computing
Wuxi, China

C.X. Yang, X.W. Ren, P.J. Ren
Institute of Artificial Intelligence and Robotics
Xi'an Jiaotong University
Shaanxi, China

Abstract--**Most of routing algorithms for On-chip communication are neither application-aware nor routing packets using multiple paths. In addition, they hardly consider link bandwidth variation resulting from widely applied global asynchronous local synchronous (GALS) mechanism. In this paper, we propose a bandwidth-based application-aware multipath routing (BAMR) algorithm to assign multiple routing paths by leveraging the knowledge of application and network bandwidth features. With the increase of number of flows resulting from the split of flows, we present a new method named dynamic-amount fixed-number (DAFN) flow control mechanism to avoid deadlock. We compare our algorithm with XY, YX, and O1TURN with synthetic traffic patterns and traffic trace of parallel implementation of H.264. Experiments demonstrate that BAMR achieves higher throughput with decrease in latency. Furthermore, the proposed algorithm achieves better workload balance by distributing traffic over multiple paths.**

Keywords-on-chip network; GALS; multiple routing

I. INTRODUCTION

The growth of deep sub-micron technology will emerge many challenges for next generation System-on-Chip (SoC) design. The tile-based Network-on-chip (NoC) has been proposed as a promising solution to the communication challenges [1]. Each module is attached with a local router which connects itself to its neighbours via a network, and modules can exchange messages through the on-chip network.

In order to increase routing efficiency, some problems have to be taken into consideration for a NoC. First, although NoCs are expected to be identical in the design phase, cores and links will have different frequencies and bandwidth, which will cause some unpredictable drawbacks and challenges [2]. According to the study in [3], an 80-core Intel chip has 28% variation between the fastest cores frequency, which results in that homogeneous design became heterogeneous. Inevitably, some designs for homogeneous are not suitable for NoC when on-die variation is taken into consideration. NoC approach offers a matchless platform for implementing the GALS paradigm [4]. Second, once a link's workload has achieved its capacity, more flows sharing this link will cause congestion and further degrade network performance. Some flows will transmit packets in very low throughput and the associated buffers will always be full. Even worse, congestion might propagate to the upstream, and leading to tremendous performance degradation. Furthermore, network has to guarantee deadlock freedom. Deadlock is catastrophic to a network, because a few resources are occupied by deadlocked packets and other packets block on these resources will paralyze the network operation.

To address the aforementioned three challenges, we propose a bandwidth-based application-aware multipath routing (BAMR) algorithm. Our algorithm is flexible to different kinds of NoC topologies and traffic patterns. We first decide routing order of flows based on application graph and the bandwidth requirement. Then, considering the transmission capacity of network links, we split some flows into multiple subflows to alleviate network congestion. Our algorithm can detect communication bottlenecks and make split based on link's residual bandwidth. For multipath routing, more flows may increase the possibility of deadlock. To this end, we propose a buffer management mechanism called dynamic-amount fixed-number (DAFN), the number of Virtual Channels (VCs) for each individual router port and VC depth are predetermined at design phase based on the BAMR's results. Specifically, the number of VCs for a specified port at a specified router is depending on the number of flows passing through it. DAFN allocates each VC at a router with an identical global label with assigned flow's label. DAFN achieves deadlock freedom by restricting each flow to its exclusive VC, it takes full advantage of buffer space and avoids deadlock without additional hardware resources.

Our work makes following contributions: We propose a multipath routing algorithm named BAMR for GALS NoC. A flow splitting method aiming at to alleviate network congestion is presented, as part of BAMR.A static VC regulator named DAFN, with the combination of BAMR to make the full utilization of VC resources and avoid deadlock is also proposed.

The remaining part of this paper is organized as follows. Section 2 shows related works and the motivation of our study. In section3, we present some definitions used in our algorithm. Section 4 describes our proposed algorithm. Section 5 evaluates BAMR performance and compares it with other previous algorithms. Section 6 concludes our work.

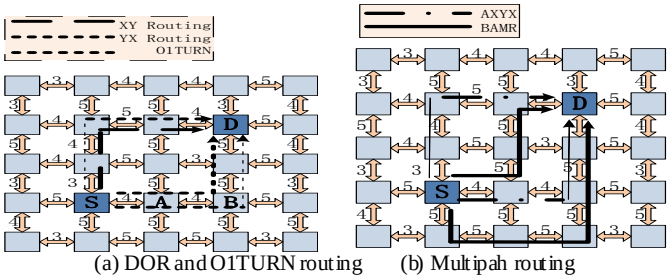
II. RELATED WORK

Routing algorithms have been extensively studied. Dimension-order routing (DOR) [5] is the most commonly

used routing algorithm due to its simplicity. DOR routes packets in one dimension, then moves to the next dimension, until the final destination is reached. In a 2-D mesh, DOR becomes XY (or YX) routing, which sends packets along the X (or Y)-dimension first, followed by the Y(or X)-dimension. A more flexible and sophisticated routing algorithm is adaptive routing whose routes options are selected depend on the state of network. Turn-model based routing [6] is a classic adaptive routing. It increases the flexibility of the algorithm by allowing six out of eight turns. Only one turn from each cycle is eliminated. In [7], the author presents a novel selection strategy called NoP that can be coupled with any adaptive routing. This algorithm is based on the concept of Neighbors-on-Path, aiming at exploiting the situations of indecision occurring when routing function returns several admissible output channels. Adaptive routing can potentially provide better throughput and fault tolerance by allowing alternative paths, depending on the network congestion and runtime faults. However, adaptive routing increases the complexity of the router implementation.

Deflective routing in [8] routes packets to one of the free output channels in minimal path. A bandwidth-aware routing for diastolic arrays and avoiding deadlock by assuming that each flow has its own private channel has been explored by Cho [9]. An adaptive multipath routing algorithm proposed in [10] selects minimal paths to send packets. In order to maintain high throughput and low latency, reducing network congestion with migrating some workload from busy links to idle links is an effective solution. In [11], Krimer makes a packet-level static timing analysis to advise optimization decision. When multiple flows share a same link, and the overall bandwidth requirement is beyond link's capacity.

Previous single path routing algorithms introduced in section 2 are not able to improve throughput when a flow's bandwidth demand is higher than the single link's capacity. Multipath routing algorithm named AXYX in [10] only sends packets in two minimal paths. Figure 1 demonstrates XY, YX, O1TURN, AXYX, and our proposed BAMR algorithm, and the underlying network is a GALS based 2D-mesh network. Links' bandwidth ranges from 3 to 5. Assuming message A is sent from node S to D, and the bandwidth requirement is 9. As shown in Figure 1a, link SA and AB are the bandwidth bottlenecks of XY routing path, where the bandwidth capacity is 4. Ideally, route from S to D without congestion can only acquire bandwidth at 4 at most. Similarly, when applying YX routing, the bottleneck bandwidth is 3. O1TURN chooses YX or XY routing with the same probability. So, the transmission bandwidth is 3.5 in average. Packets routed to destination by AXYX are routing in XY and YX at the same time. However, AXYX can achieve better bandwidth with the sum of XY and YX routing, which is 7 in this case. Previous routing algorithms only use links in the rectangles formed by node S and D. Once some links' bandwidth is fulfilled, continually sending packets to those overstressed links will lead to congestion. On the contrary, links out of the rectangle might are underutilized. Our proposed BAMR is inspired by making full use of link's available bandwidth and is not limited to minimal routing. BAMR is shown in Figure 1b, flow is split into two subflows and transmit at the same time, one in minimal path and the other in non-minimal path. By adopting our BAMR algorithm, the transmission bandwidth is 9 at most. Our proposed algorithm can route packets according to their bandwidth demands. We note that non-minimal paths have more hop count, but they may achieve higher throughput and lower latency if packets in this path will not contend with other flows. Our proposed algorithm avoids contending link resource between different flows as much as possible, which can reduce arbitration time and save buffer space. As can be expected, BAMR can achieve better workload balance by distributing traffic more evenly.



(a) DOR and O1TURN routing (b) Multipath routing

FIGURE I. FIVE KINDS OF ROUTING ALGORITHMS

III. PROBLEM FORMULATION

In this section, we first present some definitions used in our proposed routing algorithm, and then we state the optimization goals of routing algorithm.

Definition1*: An Application Communication Graph (ACG) G(T, E)* is a directed graph, where each vertex t_i represents a task and each directed edge(t_i, t_j), denoted as $e\_(i,j) \in E$, represents the communication between task t_i and t_j. Each edge is tagged with a weight $v(e\_(i,j))$ representing the communication bandwidth demand from t_i to t_j, and a communication task is represented by $c(t\_i, t\_j, v(e\_(i,j)))$ A set C contains all of the communication tasks of a given application.

Definition2*: A NoC Topology Graph (NTG) M(P,L)* is a directed graph. Each vertex $p_i \in P$ represents a processor with a local router. Each directed arc between p_i and p_j represents the link that p_i can send packets to its adjacent processor p_j through their attached router, denoted as $l_{i,j} \in L$. We tag $l_{i,j}$ with a weight $b(l_{i,j})$, which represents link's capacity. Link set L contains all the links $l_{i,j}$.

Definition3*: A mapping set S is a set whose element s:t→ p(t ∈T,p ∈P) represents a mapping function, which indicates that task t is executed on processor p.*

Definition4*: A flow set F* contains all of the communication flows for a given application after mapping, $f(t\_i, t\_j, v(e(i,j)))$ is corresponding to $c(t\_i, t\_j, v(e(i,j)))$ in communication tasks set *C*.

Definition5*: A routing solution set R* is a set of paths, a path connects p_i to p_j is represents as (p_i , ... p_n , ... p_j), with the consumed bandwidth is *consume_bw*.

Therefore, the problem can be described as: **Given** an application graph *G(T,E)*, a NoC topology graph *M(P,L)*, and a mapping set *S*; **Find** a routing solution set R; **Such that**

achieving higher throughput, lower latency, and better load balance.

TABLE I. BANDWIDTH-BASED APPLICATION-AWARE MULTIPATH ROUTING (BAMR)

```
Input: G(T,E),M(P,L) and S
Result: Routes decisions for all the communication tasks
1 initialization;
2 D_l is empty ; /* Set contains temporally removed links */
3 Generate traffic flow set F according to G(T,E),M(P,L) and S
4 Get a new flow set F´ by sorting F in descending order according to
  flow's weight v(e_ij)
5 while len(F´) > 0 do
6   f_max = max(F´)     ;   /* f_max = (p_i , p_j , v(e_ij)) */
7   for link l_i,j in M(P,L) do
8       if b(l_ij) < v(e_ij) then
9           get M´(P,L) by deleting l_i,j from M(P,L)
10          add l_i,j to D_l
11  if f_max is routable in M´(P,L) then
12    find route_path for f_max by Dijkstra algorithm
13    remove f_max from F
14  else
15    split f_max into two subflows f_1 and f_2
16     f_1 ,consume bw,route path=
17    split_flow(D_l , f_max , M´(P,L)) ;     /* invoke alg 2 */
18    f_2 = (p_i , p_j , v(e_ij) – consume_bw)
19    remove f_max and insert f_2 into F
20  add route_path to R_s
21   consume_bw = min(route_path.links)
22  for link l_ij in route path do
23    if consume_bw = b(l_ij ) then
24        delete l_ij from M(P,L)
25    else
26        b(l_ij) = (b(l_ij ) − consume_bw)
27 Make XY routing for no_available_bw flows and add it to R
28 return R
```

We state that application mapping has to be accomplished before making the routes decision. As is known, mapping problem is a quadratic assignment problem which is *NP−hard* [12]. The search space increases factorially with the system size. In order to simplify our algorithm, we assume that task t_i is executed on processor p_i as fixed.

IV. METHODOLOGY

A. Bandwidth-based Application-Aware Routing Algorithm (BAMR)

In this section, we present our BAMR algorithm. The main idea of BAMR is splitting flows, whose bandwidth requirement cannot be satisfied in a single path into multiple subflows to transmit simultaneously to achieve the bandwidth demand. We describe our proposed routing algorithm by an example. Pseudo-code of BAMR is presented in algorithm 1. Application graph *G(T,E)* and NoC topology *M(P,L)* are showed in Figure 2(a) and 2(b). As we have discussed in section 3, we focus on routing algorithm rather than application mapping, so we assume the mapping procedure has already finished, and take t_i is executed at

p_i. The routing algorithm takes *G (T,E)*, *M(P,L)* and the mapping set *S* as inputs.

TABLE II. SPLIT UNROUTABLE FLOW INTO ROUTABLE SUBFLOWS

```
1 split_flow(D_l , f_max , M´(P,L))
2 BEGIN
3 while f_max is unroutable do
4     dl_max = max(dl)
5   add dl_max to M´(P,L)
6   if f_max is routable based M´(P,L) then
7       find route_path for f_max by Dijkstra algorithm
8       consume_bw = b(l_max)
9       f_1 = (p_i , p_j, consume_bw)
10  else
11      delete dl_max from dl
12   if len(dl) = 0 then
13      add f_max to no_available_bw_flows
14 return f_1 , consume_bw, route_path
15 END
```

We firstly get flows set *F* according to *G(T,E)*, *M(P,L)* and *S*. We state that different methods of ordering flows to make route will cause different routing solutions and performance. *A* flow with less bandwidth requirement (lower weight) can be satisfied more flexibly than the one with more bandwidth requirement (higher weight), so we sort flows in descending order by communication demand. After getting the ordered flow set F' from F, we assign routing path for each flow in F' sequentially. In Figure 2(a), we first handle message connecting t_8 to t_6 with bandwidth 13 ($c_{max} = (t_8, t_6, 13)$), which is from p_8 to p_6 with weight 13 in *M(P,L)* after mapping ($f_{max} = (p_8, p_6, 13)$ accordingly). We find that routing through a single path cannot satisfy the communication bandwidth demand, so we decide to split it and use multipath to make transmission. We split f_{max} into two subflows f_1 and f_2. f_1 is routing with maximal bandwidth that NoC topology can support, f_2 is a new flow with the residual bandwidth. We will then assign a qualified path for f_1, and then insert f_2 into the flow set F at an appropriate position as a new flow, and make the routes decision afterward.

We secondly delete links whose weights (residual bandwidth) are less than the bandwidth demand of $(p_8, p_6, 13)$. In Figure 2(b), we delete links that $b(l_{i,j})$ are less than 13. It is obvious that all links will be deleted so that there is no path satisfying communication from p_8 to p_6. In order to achieve $f(p_8, p_6, 13)$, we use Algorithm 2 to split $f(p_8, p_6, 13)$ into two subflows f_1 and f_2.

In Algorithm 2, we add temporally removed links in set D_l in descending order until p_8 has at least one path to communicate with p_6. We declare that communication bandwidth of a path is decided by the minimal bandwidth of links belonging to it. Once there is a routable path from p_8 to p_6 after adding deleted link $l_{i,j}$ with a weight $b(l_{i,j})$. Therefore, $f(p_8, p_6, 13)$ can get the maximal bandwidth of $b(l_{i,j})$. In Figure 2(c), can be routable when links with weight 9 are added, and we can get a path p_8, p_5, p_4, p_3, p_6 with bandwidth of 9 from p_8 to p_6. Accordingly, $f(p_8, p_6, 13)$ is split into $f1(p_8, p_6, 9)$ and $f2(p_8, p_6, 4)$, *f1* is routed using path p_8, p_5, p_4, p_3, p_6, and then, we insert *f2* as a new flow into flow set *F*.

We thirdly find the minimal weighted path using Dijkstra algorithm. We set link's weight as $1/b(l_{i,j})$ in initial. Notice that, links with higher residual bandwidth are more possible

selected than others with lower residual bandwidth. We find the minimal weighted path from p_8 to p_6 with bandwidth 9, as is highlighted with bold line in Figure 2(c). It is worth noting that traffic pattern usually gives average bandwidth that flows demand. However, according to [13], massages are often transferred in burst-mode, which requires higher bandwidth temporally, so link capacity with bandwidth slack is necessary. Our Algorithm can add an additional weight to *v(i,j)* to satisfy these situations.

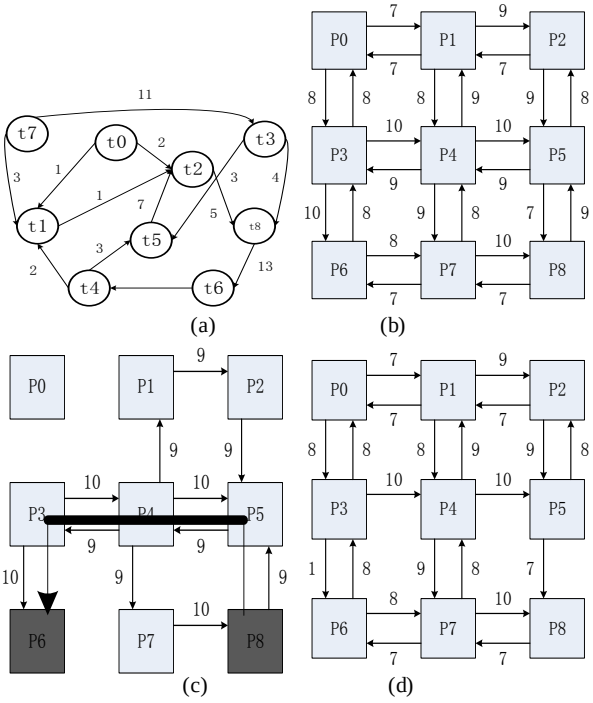


FIGURE II. ROUTING FOR FLOW WITH MAXIMAL BANDWIDTH DEMAND ((A) APPLICATION GRAPH (B) NOCTOPOLOGY GRAPH (C) ROUTING WITH BANDWIDTH 9 (D) RESIDUAL BANDWIDTH)

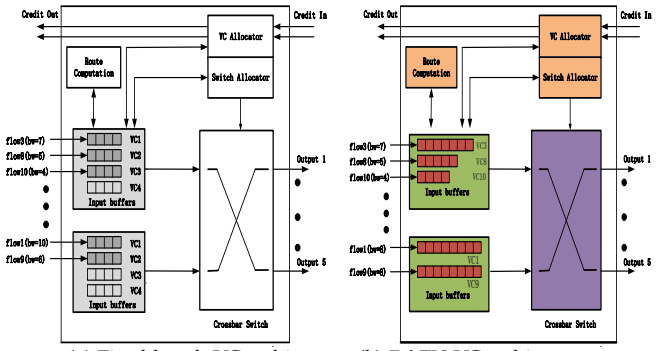
After that, we update *F*, *M(P,L)* and *S*. In the procedure of updating *F*, f_{max} is deleted from *F* if bandwidth of f_{max} can be satisfied, otherwise, f_{max} is divided as two subflows f_1 and f_2, f_1 is routable and assigned a path, but f_2 with the residual bandwidth will be inserted in *F* and treated as a new flow. In updating *M(P,L)*, we update link's weight $b(l_{i,j})$ with its residual bandwidth for links belonging to the route path. After updating *F* and *M(P,L)*, links $l_{p8,p5}$, $l_{p5,p4}$ and $l_{p4,p3}$ are removed, because these links have no residual bandwidth, and weight of $l_{p3,p6}$ is updated as the residual bandwidth 1, as shown in Figure 2(d). At last, routing path $(p_8, p_5, p_4, p_3, p_6)$ for $f(p_8, p_6, 9)$ is added to routing solution set *R*.

We find routing path for flows in the descending order of communication bandwidth requirements. In another word, flows with higher bandwidth demand are considered earlier. However, links might be deleted in *M(P,L)* when their bandwidth has been used up according to Algorithm 1.In this situation, no matter how to design routing algorithm, other preassigned flows which share link bandwidth with these unassigned flows will be affected. So, we invoke XY routing hereafter, because XY routing can provide the minimal hop counts.

B. Dynamic-Amount Fixed-Number Virtual Channel (DAFN)

After splitting one flow into multiple subflows, the probability of deadlock increases. We propose a novel method to avoid deadlock which is suitable for multipath routing. Firstly we demonstrate the algorithm is deadlock free. The assumptions are as follows:

• Each flow can only get into one fixed VC with the same global label;



(a) Fixed-length VC architecture. (b) DAFN VC architecture

FIGURE III. COMPARISON OFTWO ROUTERARCHITECTURES

• Each VC can only store the flow with the same global label.

Theorem 1: Flows control satisfying aforementioned assumptions will be deadlock free.

Proof: First, we prove that any two flows are deadlock free by contradiction. Assuming flow1 and flow2 generate deadlock, packets belonging to flow1 and flow2 are holding onto a set of network resources in a cyclic manner. However, flow1 and flow2 are using the different VCs according to assumptions, which creates contradiction. Therefore, flow1 and flow2 are deadlock free. In a similar way, any flows will not produce a cycle because every two flows can't get into the same VC, there is no circular resource dependencies among flows.

Based on the aforementioned theorem, we explore a novel mechanism to allocate VC resources to avoid deadlock. According to traditional static buffer management [14], as shown in Figure 3a, every input port has multiple queues with fixed-length. The dynamic buffer management named ViChaR is introduced in [15]. We implement DAFN by modifying ViChaR to achieve more optimization and better efficiency by leverage the knowledge of application communication requirements.

In Figure 3b, we present our DAFN mechanism. As pointed by Nicopoulos [15], routers with fixed buffer structures will either be underperformed or underutilized under certain traffic conditions. We organize VC number by considering traffic workload of every input port. In order to avoid head-of-line blocking, we provide every flow with only one VC. Necessarily, a VC can only be used by one flow and each flow can only get into one VC (note that split subflows coming from the same flow are regarded as two different flows). We label every VC globally to guarantee flows not contend the same VC with others, as shown on the left of Figure 3b. In the process of transmission, packets belonging to a given flow can only get

into the specified VC with fixed label, which is identical to flow number.

As shown in Figure 3b, flow8 can only get into VC8 in every router along its path. In this means, flow will not require other VCs so that arbitration is simple and fast without an additional pipeline stage. Our organization can satisfy all the conditions of theorem 1, so that deadlock-free is guaranteed. The buffer size of VC is variable according to flows of input ports. Considering making full use of the link bandwidth, we allocate more buffer space to flows with higher bandwidth demand. To ensure fairness, flows get buffer space is proportional to their bandwidth demand. In Figure 3, we assume amount buffer space of every input port is 16 flits. According to various of flows' required bandwidth, we allocate VC3 to flow3 with 7 flits, VC8 to flow8 with 6 flits and VC10 to flow10 with 3 flits. Compared to fixed number and length VC, our method makes full utilization of buffer space. Different to ViChaR with avoiding deadlock by escape VC and dynamic allocate of both VCs and their associated buffer depth with sophisticated mechanism, our method allocates VC and buffer resources statically according to network traffic conditions, and achieves deadlock free by fixed-label globally, which can make full use of buffer space.

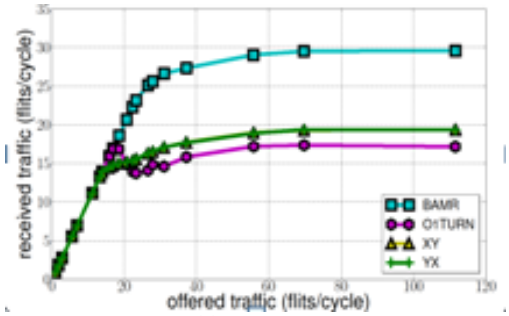
V. EVALUATION

A. Simulation Configuration

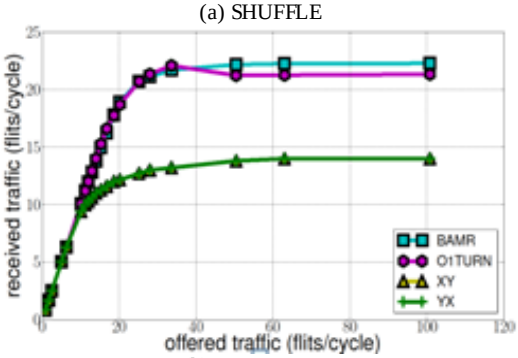
For all the experiments, we use HORNET [16]. We configure the total number of 32 flits buffer size for each port and then simulate for 100,000 cycles to collect statistics. In order to evaluate our algorithm, we measure the throughput of the network under different injection rates under a set of synthetic traffic patterns, including TRANSPOSE, BIT-COMPLEMENT, SHUFFLE and a multimedia traffic trace of parallel implementation of H.264. For synthetic benchmarks, all flows with the same bandwidth demands. H.264 is derived from a real application so that flows have different bandwidth demands.

B. Throughput Analysis

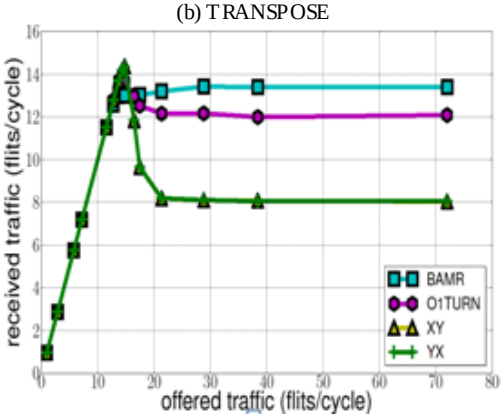
Figure 4 shows throughput results for different synthetic traffic patterns. As is shown, BAMR algorithm out-performs O1TURN, XY and YX. In the case of SHUFFLE , the improvement of saturation throughput is 71.4%, 51.3% and 51.3%. For TRANSPOSE, the improvement is 4.76%, 60.7% and 60.7%. For BIT-COMPLEMENT, the improvement is 15%, 72.5% and 72.5%. Note, in the case of BIT-COMPLEMENT, the result is very interesting. In Figure 4c, We can see O1TURN, XY and YX have a obvious throughput reduction when input injection rate is around 15 flits/cycle. This performance degradation is because of Head-of-Line(HoL) blocking. However, BAMR performs well without performance reduction when injection rate around the saturation. We note the reason is that flows in BIT-COMPLEMENT have many source and destination positions across the mesh, making it highly symmetric in the X and Y direction. As a result, a lot of flows will share same links. Flows congest in the share link and throughput degrade immediately due to the HoL blocking in O1TURN, XY and YX. Our algorithm assigns links according to links' residual bandwidth, so BAMR successfully escapes from the congested link and keeps relatively high throughput around saturation injection rate (15 flits/cycle). Figure 4d presents our algorithm has improvement of 32.6%, 10.7% and 25.7% compared to O1TURN, XY and YX. Our algorithm is attractive because it makes full use of underutilized links and avoid contending around shared links.



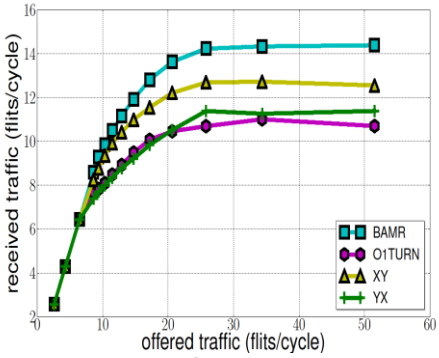
(a) SHUFFLE



(b) TRANSPOSE

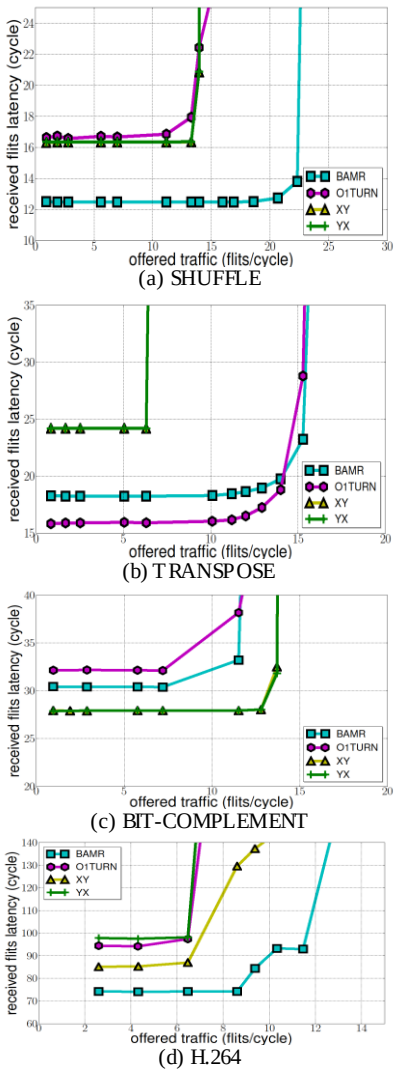


(c) BIT COMPLEMENT



(d) H.264

FIGURE IV. THROUGHPUT OF FOUR ALGORITHMS IN DIFFERENT TRAFFIC PATTERNS



(a) SHUFFLE

(b) TRANSPOSE

(c) BIT-COMPLEMENT

(d) H.264

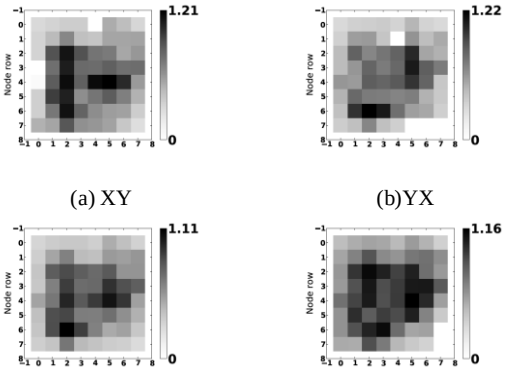
FIGURE V. LATENCY OF FOUR ALGORITHMS IN DIFFERENT TRAFFIC PATTERNS

C. Latency Analysis

Figure 5 presents latency results of various routing algorithms. BAMR has better performance in SHUFFLE and H.264 compared to other algorithms. The average latency reduction is 9.7%, 12.4% and 15.3%, compared to O1TURN, XY and YX. Similar to throughput trend, the performance of latency for BIT-COMPLEMENT and TRANSPOSE are medium. Because our algorithm may find longer paths for routing, resulting in more hop counts flows can be routed to their destinations.

D. Workload Analysis

Our algorithm not only achieves the highest throughput and lower latency, but also better workload balance. Figure 6 shows spatial power distribution of different algorithms used in H.264. As we can see BAMR leverages more underutilized links and release hotspot. By migrating workload from heavy links to idle links, our algorithm achieves better workload balance. Power results are achieved by Orion2.0 [17] that is integrated in HORNET.



(a) XY (b)YX

(c) O1TURN (d) BAMR

FIGURE VI. WORKLOAD DISTRIBUTION IN DIFFERENT ALGORITHMS

VI. CONCLUSION

We have proposed an efficient algorithm to route packets, based on the knowledge of property of communication, which making full use of link's bandwidth, so that network achieves high throughput and low latency. We also explore a new method by using dynamic amount and fixed number VC to avoid deadlock and help improve routing performance, especially for GALS NoC.

The primary feature of our algorithm is finding suitable routing paths for packets according to application's characteristics and link bandwidth of NoC. We find routing paths for flows in descending order according to bandwidth requirement. We split some flows beyond link band-width into multiple subflows to balance traffic of network. However, the main limitation of the proposed method is that, our algorithm has to address out-of-order problem resulting from multipath routing, re-order buffers are needed at the receiver side, because packets belonging to same flow might reach the destination in an out-of-order fashion. The other limitation is that our proposed algorithm is based on fixed mapping strategy, which sacrifices some potentials and restricts routing flexibility.

ACKNOWLEDGEMENTS

This research is partially funded by NSFC grant No.610303036 and the Open Project Program of the State Key Laboratory of Mathematical Engineering and Advanced Computing No.2014A09.

REFERENCES

[1] W. J. Dally and B. Towles, Route packets, not wires: On-chip interconnect-ion networks. Design Automation Conference, IEEE, pp. 684–689, 2001.

[2] E. Humenay, D. Tarjan, and K. Skadron, Impact of process variations on multicore performance symmetry. Proceedings of the conference on Design, automation and test in Europe. EDA Consortium, pp. 1653–1658, 2007.

[3] S. Dighe, S. Vangal, P. Aseron, S. Kumar et al. Within-die variation-aware dynamic-voltage-frequency scaling core mapping and thread hopping for an 80-core processor. Solid-State Circuits Conference Digest of Technical Papers (ISSCC), International.IEEE, pp. 174–175, 2010.

[4] U. Y. Ogras, R. Marculescu, P. Choudhary, andD.Marculescu, Voltage-frequency island partitioning for gals-based networks-on-chip. Design Automation Conference, 44th ACM/IEEE. IEEE, pp. 110–115, 2007.

[5] W. J. Dally and C. L. Seitz, Deadlock-free message routing in

multiprocessor interconnection networks. Computers, IEEE Transactions on, 100(5), pp. 547–553, 1987.

[6] C. J. Glass and L. M. Ni, The turn model for adaptive routing. ACM SIGARCH Computer Architecture News, 20(2). ACM, pp. 278–287, 1992.

[7] G. Ascia, V. Catania, M. Palesi, and D. Patti, Implementation and analysis of a new selection strategy for adaptive routing in networks-onchip. Computers. IEEE Transactions on, vol(6), pp. 809–820, 2008.

[8] E. Nilsson, M. Millberg, J. Oberg, and A. Jantsch, Load distribution with the proximity congestion awareness in a network on chip. Design, Automation and Test in Europe Conference and Exhibition, IEEE, pp. 1126–1127, 2003..

[9] M. H. Cho, C.-C. Cheng, M. Kinsy, G. E. Suh, and S. Devadas, Diastolic arrays: throughput-driven reconfigurable computing. Computer-Aided Design. ICCAD 2008. IEEE/ACM International Conference on. IEEE, pp. 457–464, , 2008.

[10] M. Morvarid, M. Fathy, and R. Berangi, Adaptive multipath routing algorithm for network on chips. Work-in-Progress Proceedings, p. 9, 2010.

[11] E. Krimer, M. Erez, I. Keslassy, A. Kolodny, and I. Walter, Packet-level static timing analysis for nocs. Networks-on-Chip, 2009. NoCS 2009. 3rd ACM/IEEE International Symposium on. IEEE, pp. 88–88, 2009.

[12] M. R. Garey and D. S. Johnson, Computers and intractability. Freeman San Francisco, 1979, vol. 174.

[13] I. Cohen, O. Rottenstreich, and I. Keslassy, Statistical approach to networks-on-chip. Computers, IEEE Transactions on, 59(6), pp. 748–761, 2010.

[14] N. E. Jerger and L.-S. Peh, On-chip networks. Synthesis Lectures on Computer Architecture, 4(1), pp. 1–141, 2009.

[15] C. A. Nicopoulos, D. Park, J. Kim, N. Vijaykrishnan, M. S. Yousif, and C. R. Das, Vichar: A dynamic virtual channel regulator for network-on-chip routers. Microarchitecture, 2006. MICRO-39. 39th Annual IEEE/ACM International Symposium on. IEEE, pp. 333–346, 2006.

[16] P. Ren, M. Lis, M. H. Cho, K. S. Shim, C. W. Fletcher, O. Khan, N. Zheng, and S. Devadas, Hornet: A cycle-level multicore simulator. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, vol. 31, no. 6, pp. 890–903, 2012.

[17] A. B. Kahng, B. Li, L.-S. Peh, and K. Samadi, Orion 2.0: A power-area simulator for interconnection networks. Very Large Scale Integration Systems, IEEE Transactions on, vol. 20, no. 1, pp. 191–196, 2012.