AVCS-NoC: A Novel Adaptive Virtual Channel Sharing Networks-on-Chip Architecture

Baoliang Li, Zeljko Zilic, Wenhua Dou,

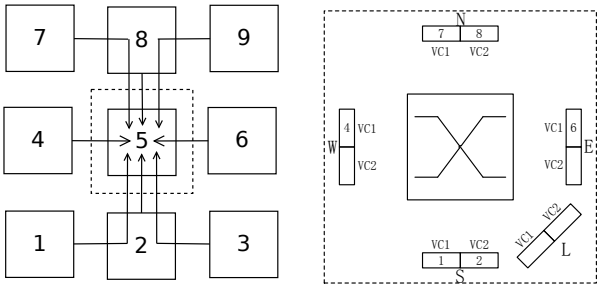
***Abstract*—The performance and hardware overhead of wormhole switched Networks-on-Chip (NoC) heavily depend on the organization and management scheme of input buffer. In order to achieve high performance without introducing significant power and area burden, the allocation of buffer resource should adapt the dynamical changing of traffic load. In this paper, we propose an inter-port Adaptive Virtual Channel Sharing NoC architecture, i.e. AVCS-NoC, which adjusts the VC number and buffer capacity each input port can use to improve the buffer utilization and network performance. Another contribution of this paper is that we introduce the idea of port sharing while implementing the VC Allocator (VCA) and SWitch Allocator (SWA), which utilizes a smaller VCA and SWA to meet the demand of large scale VC arbitration. Based on a detailed RTL implementation, we evaluate the performance and hardware overhead of our proposal. Experimental results show that the proposed micro-architecture provides x% and y% average latency reduction under hotpsot and transpose traffic pattern near the saturation point while saving z% area when compared to the linked-list based dynamical VC structure. For a real-work Video Object Plan Decoder (VOPD) application, our proposal improves the average latency by y%. Our proposal saves x% power and y% chip area when compared with the typical router architecture with similar performance.**

***Keywords*—Networks-on-Chip (NoC), VC sharing, buffer utilization**

I. Introduction

The wormhole switched Networks-on-Chip (NoC) provides a scalable, high performance and low cost interconnection fabric for the large scale Chip-MultiProcessor (CMP) and System-on-Chip (SoC). The performance, power and area of NoC router heavily depend on the buffer capacity and organization scheme. In a typical router, all the input ports have the same number of VCs and each VC is designated a fixed buffer capacity [1]. Although this static structure eases the implementation, the buffer utilization under some uneven traffic pattern is very poor. Because the traffic load is nonuniform and time variant. For the input port with lower traffic load, few VCs and buffer space are enough to temporarily hold the incoming packets that cannot be forwarded immediately, while more VCs and buffer space are necessary for the input ports with high traffic load to guarantee the throughput. Since the on-chip buffer consumes more than 60% power and chip area of a router [2][3], reserving large enough buffer space for each input port to ensure the performance under high traffic load is unfeasible.

The only way to implement high performance and low cost NoC is by improving the buffer utilization through appropriate buffer organization and management scheme. There have been significant works on this issue, e.g. [4], [5], [6], [7] [8]. To save buffer, these approaches dynamically allocate the buffer slots within an input port to each VCs according to the traffic conditions, which achieve 50% buffer saving without degenerating the network performance [4]. To further improve the performance and buffer utilization of router, inter-port buffer sharing was proposed in [8][6]. Although all these approach employ the dynamical buffer management, the number of VCs each input port can use is still fixed. We find that, under some circumstances, these approaches can not fully exploit the performance with limited buffer resource. We take a 3×3 mesh network employing Dimension Order Routing (DOR) as an example to illustrate one of these scenarios, as shown in Fig. 1. Packets from router 9 and 3 to router 5 are blocked due to lack of available VCs at the input port N and S. Whereas, 4 VCs in total at input port W, E and L are idle. This example reveals that only sharing buffer among VCs and port is not enough to maximize the performance of wormhole router, because the limited VC resource restricts the usage of buffer. Since the number of VC an input port need at any time instance is equal to the number of partial packets it holds, it changes dynamically with time and traffic pattern.



(a) All-to-1 Communication (b) VC Status of Router 5

Fig. 1: Blocking caused by VC insufficiency

To avoid the blocking caused by VC insufficiency, an intuitive solution is that reserving large amount of VCs for each input port to meet the worst-case VCs requirement. However, this solution is infeasible due to the follow two reasons: first, it introduces significant hardware overhead to manage and keep the status of each VC; second, it makes the VC allocator (VCA) and SWitch Allocator (SWA) very complex and further limits the frequency of router, because these two components usually lay on the critical path of entire router pipeline. We also noticed that, the port utilization of VC and switch allocator in the typical router micro-architecture is very low. For a $N \times N$ mesh network, supposing each

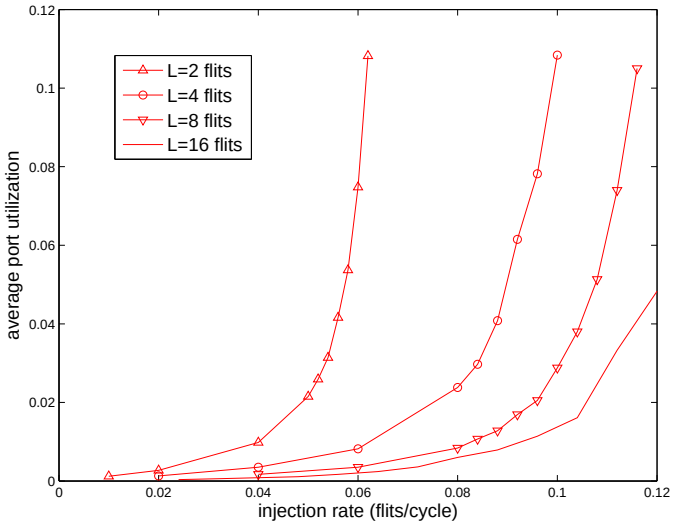


Fig. 2: The port utilization of VC allocator under different injection rate and packet length.

router has P input port, and each port has V VCs. Then, the average port utilization of VC allocator when the network is unsaturated can be defined as:

$$\rho = \frac{1}{N^2PV} \times \sum_{n=1}^{N}\sum_{p=1}^{P}\sum_{v=1}^{V} \frac{ActiveCycles_{n,p,v}}{SampleCycles}$$

where $SampleCycles$ is the length of sample period in cycles and $ActiveCycles_{n,p,v}$ is the number of cycles that the pth allocator port of pth port in router n receives a request. For the many-to-1 communication scheme shown in Fig. 1, supposing each input port of router is deployed with 16 buffer slots and 2 VCs. We change the packet length L from 2 flits to 16 flits, and collect the port utilization of VC allocator under different injection rate. As indicated in Fig. 2, the average port utilization of VC allocator is very poor, and it decreases significantly while increasing the packet length. This observation motivates us to design new VC and switch allocator structure to improve the port utilization, because this improvement makes it possible to meet the same arbitration demands with lower hardware cost.

In this paper, we propose an inter-port Adaptive VC and buffer Sharing NoC micro-architecture (AVCS-NoC), which alleviates the blocking caused by both flow control and VC insufficiency and improve network performance. Our approach reserves some buffer and VC resource for each input port and leaves the other buffer and VC shared by all the input ports. Under low traffic load, the private buffer and VC are sufficient to hold the incoming packets that can not been forwarded immediately. While under high traffic load, the shared VC and buffer can be employed to alleviate the blocking caused by both flow control and VC insufficiency. Another contribution of this paper is that we proposed the idea of allocator port sharing to decrease the complexity of VCA and SWA, which employs a smaller allocator than the typical router to arbitrate among large amount of VC request. All the newly added hardware components in our approach works in parallel with the original router pipeline, which introduces no negative effect to the frequency.

The reset of this paper is organized as follows: A summary of related work follows in Section II; We introduce the micro-architecture of PVCS-NoC in Section III; Related experimental results are presented in Section IV. Finally, we conclude this paper and give the future work in Section V.

II. Related Work

The buffers located at each input port of a router account for a large fraction of the overall area and power budget of typical NoC router [2][3]. Thus, reduce the buffer capacity and improve the buffer utilization is essential to implement a high performance NoC without introducing significant hardware overhead. Proper buffer sizing and organization are essential to increase the buffer utilization of wormhole switched NoC. There have been significant works to address this problem, and several dynamical buffer structure have been proposed.

Virtual Channel Regulator (ViChaR) was proposed in [4], which utilizes a table-based Unified Buffer Structure (UBS) to dynamically allocate the buffer slots for each VC according to the traffic conditions. Although the buffer utilization is greatly improved, it introduces significant power burden. A novel Dynamically Allocated Multiple Queue (DAMQ) was proposed for NoC systems [9], which provides the same performance while keep the hardware cost very low. To eliminate the additional three-cycle delay of conventional DAMQ structure, a prefetch mechanism was proposed in [7]. In [5], the authors proposed a linked-list based buffer structure and a congestion avoidance scheme to improve the network performance and reduce the power and area cost of NoC.

In [8][6], inter-port buffer sharing was proposed to further improve the buffer utilization of typical router. The control logic of this fully shared scheme is more complex than the intro-port buffer sharing schemes and introduces significant hardware overhead. Accordingly, a similar inter-port buffer-stealing scheme for the normal router without VC was proposed in [10]. To alleviate the addition control overhead of fine-grained buffer sharing, a Partial Virtual Channel Sharing NoC (PVS-NoC) was proposed in [11], which shares some FIFO buffer between neighbor input port. An VC renaming mechanism was proposed in [12] for NoC to virtualize the physical VC and increase the robustness of NoC in case of failure. In addition, to alleviate the performance degradation caused by coupling among VCs, an adaptive back-pressure mechanism was proposed in [13].

The common characteristic of all these buffer sharing schemes is that they only share the buffer resource and reserving large amount of VC for each input port to guarantee the worst case performance. Whereas, these statically allocated VC scheme introduces significant hardware overhead and the utilization of VC and switch allocator is very low. In addition, under some real-world traffic pattern, the even distributed VC architecture is low efficient due to lack of VC at some input ports. Thus, only combining the buffer sharing and VC

sharing can we implement a high performance router without introducing high hardware overhead.

III. Proposed AVCS-NoC Micro-Architecture

A. Key Design Consideration

Virtual channel provides a way to multiplex the physical channel in wormhole switched NoC to reduce the Head-of-Line (HoL) blocking. In avoid packets interleaving, a VC can be reallocated only when the previous packet has been completely received. If the packet occupying a VC is blocked due to some reason, this VC keeps on idle and can not be utilized by other packets. Thus, the degradation caused by VC insufficiency is much serious than that of flow control, especially when the packet is very long. One of the solution to this problem is that reserving large amount of VCs for each input, e.g. [4][5][6][8][7]. Whereas, it makes the VC and switch allocator very complex, and a large amount of registers should be used to track the state of each VC. Another promising approach is by sharing the buffer and VCs among all the input ports, and allocate the buffer and VCs dynamically on demand according to the traffic conditions and the VC status of each input port.

While designing the inter-port buffer and VC sharing scheme, the following four issues profoundly affect the design consideration of our AVCS-NoC micro-architecture:

1) Interference caused by buffer sharing: When the buffer resource are shared among all the input ports, an adversarial workload injected from one input port might monopolize all the buffer space of other input ports and cause global congestion. Thus, special care should be taken to prevent the buffer blockage from spreading to the other ports.
2) Read/Write conflicts caused by buffer sharing: Since each input buffer can be accessed by multiple input ports, potential conflicts might occur when multiple input ports trying to write flits to or read flits from the same buffer. Thus, additional read/write control logic should be implemented to avoid hazard.
3) Trade-off between hardware overhead and performance: Although the inter-port buffer sharing can further improve the performance of typical router, additional hardware overhead increases significantly. Thus, the trade-off between performance and hardware overhead is necessary to design a high performance router with acceptable hardware overhead.
4) The complexity of VC and switch allocator: When the inter-port VC sharing is implemented, the maximal number of VC each input port can use is equal to the number of private VCs plus the number of shared VC, which is larger than that of non-shared scheme. Thus, special care should be taken while designing these two allocators, since they usually occupy large chip area and restrict the frequency of router.

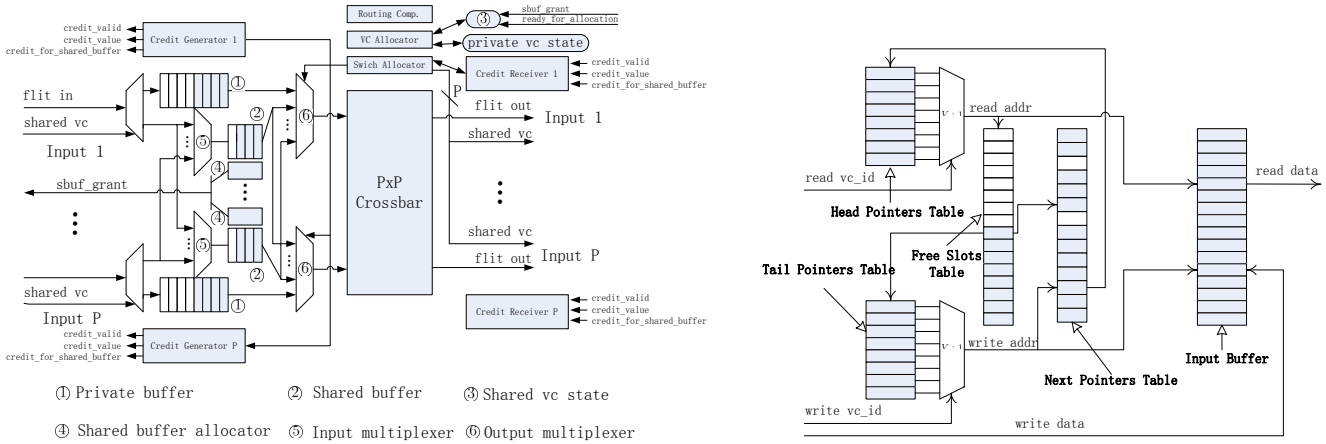
B. Proposed Micro-architecture

Taking all these aspects into consideration, we proposed the Adaptive VC Sharing NoC micro-architecture (AVCS-NoC), as shown in Fig. 3a. This proposal realizes the partial buffer and VC sharing: each input port contributes a small portion of their buffer and VCs resource to form the shared buffer banks, which can be accessed by all the input ports of a router. For an AVCS-NoC router with five input ports, there are five banks in total, and each bank manages their own buffer space independently. The owner of each shared buffer bank is determined by their own shared buffer allocator (④ in Fig. 3a). The buffer allocator is a Finite State Machine (FSM), it grants a new owner according to the traffic conditions of each port whenever this bank keeps idle for several cycles.

After buffer allocation, the allocation results are sent to the upstream routers by raising the shared_buffer_grant signal. Each output port of upstream router updates the shared VC state upon receiving the signal. The shared VC are allocated at the granularity of bank, only when a buffer bank is granted to an input port, the corresponding shared VC can be allocated at upstream router. The switch allocator schedules the flits located at both shared VC and private VC to traverse the crossbar and enter the input port of downstream router. Upon arriving the input port of a router, a flit is written into the shared/private buffer according to the shared_vc signal. Each shared buffer bank utilizes a multiplexer (⑤ in Fig. 3a) controlled by the shared buffer allocator to select the flits coming from the owner port. Similarly, each input port of the crossbar uses a multiplexer (⑥ in Fig. 3a) controlled by the switch allocator to select the flits from either private buffer or one of the shared buffers.

The credit based flow control requires a credit receiving module at the output port of a router and a credit generating module at the input port of downstream router. In our AVCS-NoC architecture, since the private buffer and shared buffer are managed independently, we need a dedicated credit counter for each memory bank. To distinguish whether the credit is for the shared VC, an additional signal credit_for_shared_vc driven by credit generator is connected to the corresponding credit receiver besides the credit_valid and credit_value signal.

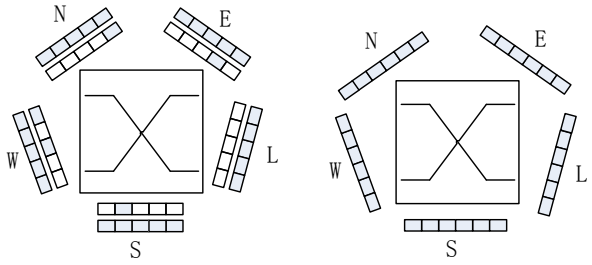
To improve the performance with smaller buffer space, we must manage and allocate the buffer resource dynamically according to the traffic conditions. Two methods for this aim include the Unified Buffer Structure (UBS) based [4][6] and the linked-list based [5][8] dynamical buffer management scheme. Since the UBS is only applicable to the fixed-length packet network, we adapt the linked-list based management scheme for both shared memory bank and the private buffer in our approach to guarantee the buffer utilization. Five tables are necessary to implement the linked-list based buffer management scheme, as shown in Fig. 3b. A Next Pointers Table records the next slot address for each buffer slot; A Tail Pointers Table maintains the writing address of each VC; A Header Pointers Table keeps the reading address for each VC; A Free Slots Table tracks the address of all the unused



(a) Proposed AVCS-NoC micro-architecture.

(b) The internal structure of shared/private buffer

Fig. 3: The proposed AVCS-NoC architecture and the buffer mangement scheme



(a) AVCS-NoC

(b) typical router

Fig. 4: Buffer organization of AVCS-NoC and typical router.

buffer slot; In addition, a VC State Table is used to identify the occupied VCs.

The main difference between the typical router and our proposal lies in the buffer and VC organization, as shown in Fig. 4. For the same amount of buffer slots B, each input port in AVCS-NoC reserve $5/6B$ as private buffer and shares $B/6$ with the other input port. Meantime, the number of VCs within the shared buffer bank, i.e. $nVC = V_{max}/P$, are shared among all these input ports. The shared buffer can change their owner according to the traffic condition of each input port to improve the buffer utilization and network performance. While under low traffic load, the private buffer and VC of each input port is capable of delivering all the incoming traffic. Under high traffic load, if congestion occurred at some input port, the shared buffer of idle port will be designated to the congested port several cycles later to alleviate the congestion.. The amount of buffer space each input port can use lies between $5/6B$ and $10/6B$, in contrast to B in typical router. The memory allocation module works in parallel with the router pipeline, which imposes no negative effect on the working frequency of entire router.

Our proposal resolves all the aforementioned questions:

1) This architecture is a partial buffer and VC sharing scheme, even when all the shared buffer is occupied by some input ports, the other ports can still use their private buffer to guarantee the transmission of packets, which alleviates the interference caused by inter-port buffer and VC sharing.
2) The hardware overhead is much lower than the fully-shared scheme, which realized better trade-off between cost and performance.
3) The shared buffer is divided into several small banks to support the parallel access. Each bank can only be granted to one input port at any time, which avoid the multiple read/write conflicts.
4) In order to minimize the complexity and overhead of VC and switch allocator, we let the shared VC and private VC sharing the same allocator port, as explained in subsection III-D. This approach makes it possible to arbitrate large amount of VC use a smaller allocator, which reduces the complexity of VC and switch allocator significantly.

In addition, our proposal can withstand the presence of faulty of both private and shared VCs, since a faulty private VC can be replaced by a non-faulty shadow VC, and a faulty shared VC does not interrupt the the functionality of corresponding private VC.

This micro-architecture is different from the previously proposed ViChaR [4], which need fully re-implement the router pipeline. Instead, our approach augments the typical router micro-architecture by adding a shared buffer allocation module for each shared buffer bank and re-implementing the VC and switch arbitration module, which will be introduced in the following two subsections.

C. Shared Buffer Allocation Module

The FSM of buffer allocator in AVCS-NoC has three states: Enable_Allocation(EA), Disable_Allocation(DA) and Change_Allocation(CA), as shown in Fig. 5. The Ready_for_Allocation(RA) signals keep on valid while in Enable_Allocation state, and the shared buffer and VCs can be used by this input port. When the shared buffer keeps on

idle for K cycles, we can infer that this input port does not need this shared buffer temporally. Thus, we can reallocate this buffer to another input port. The threshold K is a design parameter, which affects the allocation of buffer among input port and the performance of entire network. For a real-world application, the optimal K can be chosen by experiments. To avoid the buffer is allocated while changing status, the FSM first enters the Disable_Allocation state. At this state, the ready_for_allocation (RA) signals goes down to disable the output VC allocation of upstream router. When the shared buffer is empty and no shared VC is allocated, the FSM enters the Change_Allocation state. At this state, the buffer allocator chooses the new owner for the shared buffer according to the congestion status of each input port and the current owner of this bank (sbuf_grant), as described in Alg. 1. Then, the FSM enters Enable_Allocation state and begins another cycle.

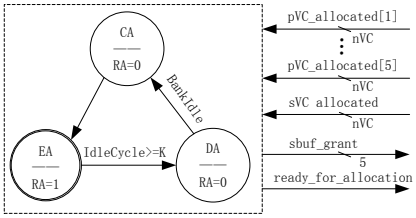


Fig. 5: The FSM of buffer allocator

Port	Encode
East	10000
West	01000
South	00100
North	00010
Local	00001

Fig. 6: Port Encoding

To simplify the allocation algorithm, we allocate the buffer bank to an input port in the round-robin order. Initially, these five buffer banks belongs to different input ports (Step 3 and Step 6). We also use a five bits port_mask to filter out the ports can not be chosen as candidate at current allocation cycle (Step 4 and Step 7). When the FSM enters Change_Allocation state, the five bits port_busy (generated in Step 11) and the port_mask signals are used to determine the new owner of this bank (Step 15 to 19). The fundamental idea of this algorithm is that choosing the candidate in order and skip the idle input ports to accelerate this process. If the position of the left-most '1' in the selector signal (generated at Step 13) matches one of the input ports listed in Table 6, the shared buffer is granted to this port. Thus, the choosing order of this algorithm is: East $\rightarrow$ West $\rightarrow$ South $\rightarrow$ North $\rightarrow$ Local $\rightarrow$ East. One example of how this algorithm functions is illustrated in Fig. 7. For a real-world application, we can also change the allocation algorithm to adapt the characteristic of traffic pattern.

D. VC and SW Allocator

The goal of inter-port VC sharing is to alleviate the performance degradation caused by VC blocking. Because the allocation results changes with traffic load, our architecture is more suitable to address the blocking caused by unbalanced VC and buffer requirement of router ports. After implement the VC sharing in AVCS-NoC, the VCs each input port can use includes the private VCs of this port and the VCs shared among all these ports. The number of VCs each input port can use varies with traffic condition, since an input port with heavy traffic load is likely to own more buffer banks. Denote

Algorithm 1 Shared buffer allocation

```
1:  Initialized:
2:      if(bank_id==5)
3:       sbuf_grant = 5'b00001;
4:       port_mask = 5'b11111;
5:      else
6:       sbuf_grant = East >> bank_id;
7:       port_mask = 5'b11111 >> (bank_id+1);
8:      endif
9:  Change_Allocation:
10:     for(p=1;p<=5;p=p+1)
11:      port_busy[p] = | private_vc_allocated[p];
12:     endfor
13:     selector = port_mask & port_busy;
14:     case(selector)
15:        5'b1xxxx: sbuf_grant = East; port_mask=5'b01111;
16:       5'b01xxx: sbuf_grant = West; port_mask=5'b00111;
17:      5'b001xx: sbuf_grant = South; port_mask=5'b00011;
18:      5'b0001x: sbuf_grant = North; port_mask=5'b00001;
19:      5'b00001: sbuf_grant = Local; port_mask=5'b11111;
20:        default: port_mask=5'b11111;
21:     endcase
```

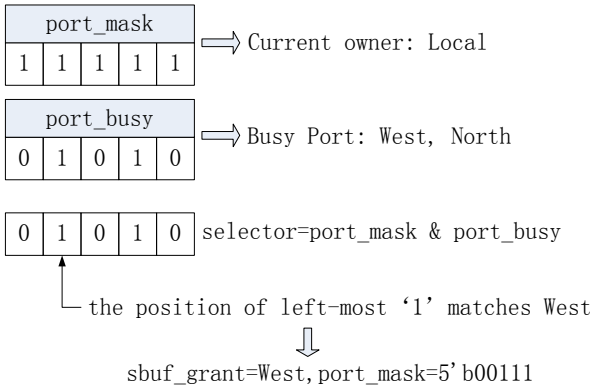


Fig. 7: A step-by-step example of buffer allocation algorithm

the number of input ports and the maximal VCs each input port can use as P and V_{max}. The VC allocator matches $P \times V_{max}$ requests from P input ports with $P \times V_{max}$ output VCs. Similarly, the switch allocator matches the $P \times V_{max}$ requests from P input ports to P output ports. To support the worst-case VC and switch allocation, one solution is that by adapting a $PV_{max} \times PV_{max}$ VC allocator and a $PV_{max} \times P$ switch allocator. However, this will make the entire allocator complex and introduce significant hardware overhead.

In this paper, we employ another approach. Our approach relies on two properties of AVCS-NoC: The number of shared VCs in AVCS-NoC is equal to the number of private VCs of each input port, and the number of VCs each input port can use lies between $1/2V_{max}$ and V_{max}. We assign an unique ID to each shared VC, and let these shared VCs compete the allocator ports with private VCs. Thus, our approach use a

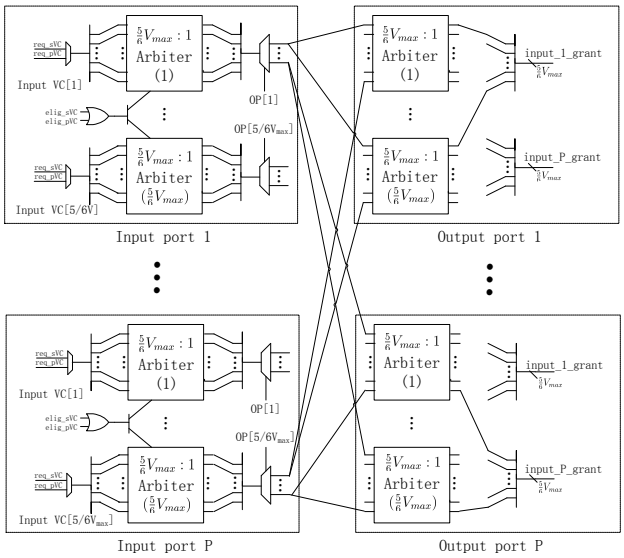


Fig. 8: The Implementation of VC allocator

$1/2PV_{max} \times 1/2PV_{max}$ VC allocator and a $1/2PV_{max} \times P$ switch allocator to perform the VC and switch allocation. The reason we employ a smaller VC and switch allocator with port sharing rather than a larger allocator lies in the fact that the port utilization of VC and switch allocator is very low, especially for the VC allocation, as indicated in Fig. 2. Because only when a head flit can issue an allocation request and this port will keeps on idle until the tail flit departure. The idle request port in our approach can be reused to improve the utilization: When the shared VC has no request, this port can be reused by the private VC and vice versa. This improvement makes it possible to utilize a smaller allocator to perform the VC and switch arbitration.

Specifically, when a shared buffer is designated to an input port, the eligible flags of all the corresponding shared VCs are asserted to allow the granting of a request. To enable the sharing of allocator port, we add a $2:1$ selector to each request port to choose a request from either private VC or shared VC. This selector chooses these two input VCs in round-robin order to prevent livelock. Then a $1/2PV_{max} \times 1/2PV_{max}$ separable input first allocator is used to perform the subsequent arbitration. The separable input first allocator employs $1/2V_{max}$ $1/2V_{max} : 1$ arbiters at each input port to grant at most 1 output VC to this request port, and each output port uses $1/2V_{max}$ $1/2PV_{max} : 1$ arbiters to select one input request for each output VC. In order to make the allocation of shared buffer sensitive to the dynamical changing of traffic, we assign lower priority to the shared output VC. A shared output VC is allocated only when all the private VCs have been occupied. In contrast, to meet the same arbitration demand, the separable input first allocator used in the typical router needs V_{max} $V_{max} : 1$ arbiters to guarantee each input VC can only request at most 1 output VC, and each output port employs V_{max} $PV_{max} : 1$ arbiters to grant each output VC to a unique input VC.

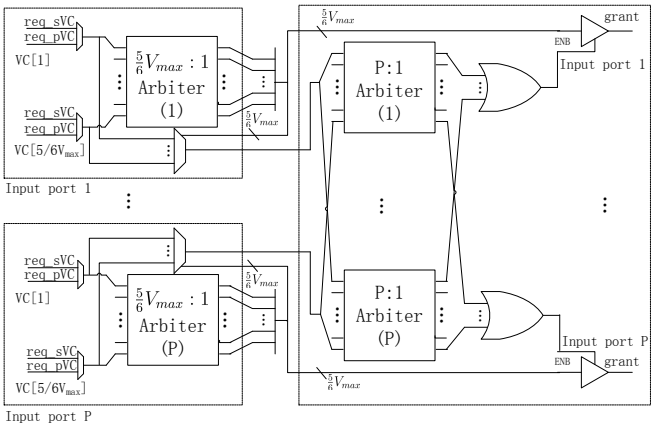


Fig. 9: The Implementation of SW allocator

Similarly, for the switch allocator, if one of the VC is blocked due to some reasons, the corresponding allocator port can be used to perform switch arbitration for the shared VC. As shown in Fig. 9, we first use a $2:1$ arbiter to select on request from either shadow VC or private VC at the first stage. We select these two VCs in round-robin order to avoid live lock. The second stage utilizes a $1/2V_{max} : 1$ arbiter at each input port to select one winner from all the request. Then, a third stage with a $P:1$ arbiter at each output port is applied to select one request for each output. While performing the switch allocation, the buffer availability of the downstream routers is considered. In contrast, to meet the same arbitration demand, the separable input first allocator used in the typical router needs V_{max} $V_{max} : 1$ arbiters to guarantee at most 1 input VC be granted, and each output port employs a $P:1$ arbiters to grant to a unique input VC.

We will demonstrate the hardware saving of our proposal in subsection IV. We also need to emphasize that although this sharing scheme might impose additional latency to the packet failed the arbitration at the first stage, the average network latency is greatly improved due to the inter-port VC sharing.

IV. Experimental results

To compare the performance and hardware overhead of our approach with the existing proposals, we implement the AVCS-NoC micro-architecture, the typical router architecture and the linked-list based dynamical buffer architecture. Under the same configuration, shown in Table I, we report the comparison results on hardware cost, performance, power and chip area in the following three subsections.

TABLE I: Architecture parameters used in the experiments

network topology	4×4 mesh	routing algorithm	DOR
channel width	128 bits	buffer size	24 flits
flit width	128 bits	packet length	8 flits
sampling period	1×10^5 cycles	warmup period	3×10^5 cycles

A. Control Overhead Comparison

Both the UBS [4][6] and linked-list based [5][8] buffer management scheme require large amount of control logic

Structure	Control Register
UBS[4]	$5((V+B)+(1+LV)\lceil log_2 B\rceil+\lceil log_2 V\rceil+2V\lceil log_2 L$
Linked List[5]	$5(B+(1+B+2V)\lceil log_2 B\rceil)$
our approach	$5(B+(1+\frac{5}{6}B+\frac{10}{6}V)\lceil log_2 \frac{5}{6}B\rceil+(1+\frac{B}{6}+\frac{2}{6}V)\lceil log_2$

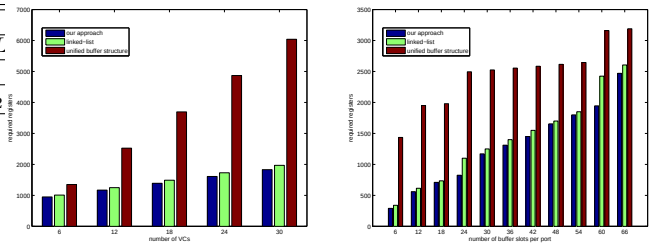
TABLE II: Hardware Overhead Comparison

and registers to keep track of the buffer usage. For the inter-port buffer sharing scheme [8], the hardware overhead is even more since the Next Pointer and Free Slot Pointer for each input port, the Header Pointer and Tail Pointer for each VC must be able to index all the buffer slot within a router. Supposing the typical router [4][6][5][8] has B buffer slots and V VCs at each input port, and the packet length is L. Under the same condition, our approach has $5/6V$ private VCs and $5/6B$ buffer slots at each input port, and each shared buffer bank has $1/6B$ buffer slots and $1/6V$ shared VCs. For each input port of the UBS architecture, the VC Availability Tracker and Slots Availability Tracker require $V+\lceil log_2 V\rceil$ and $B+\lceil log_2 B\rceil$ registers, respectively; The VC Control Table needs $VL\lceil log_2 B\rceil$ registers; Both the Arriving and Departing Flit Pointers require $2V\lceil log_2 L\rceil$ registers. For each input port of the linked-list based architecture [5], the Free Slots Table requires B registers to track the unused slots and $\lceil \log_2 B\rceil$ registers to point to the first available slot. The Next Pointers Table requires $B\times\lceil log_2 B\rceil$ registers to maintain the flits order of all the VCs. Both the Head Pointers Table and Tail Pointers Table need $V\times\lceil log_2 B\rceil$ registers. Similarly, the hardware overhead of our proposal can be derived, as shown in Table II.

To make concrete comparison, for the same packet length $L=8$ flits, we plot the hardware overhead of each scheme listed above under different buffer capacity and VC number in Fig. 10. As shown in this figure, our approach uses the lest control logic among all these proposals. because our approach divides the buffer resource into several banks, and each bank manages their own buffer resource independent. The total hardware overhead is greatly reduced since the addressing space of each buffer bank decreases. When the flit width, VC number, buffer size, packet length are 128bits, 12, 24 flits, 8 flits, respectively, the total hardware overhead of our approach is only 5.37%, whereas the hardware overhead of linked-list and UBS approach are 7.16% and 16.24%, respectively. We also need to emphasize that although our proposal introduces additional hardware overhead, the utilization improvement makes it possible to halve the buffer slots without degrading the performance, as presented in the following subsection. Thus, our proposal is cost-efficient.

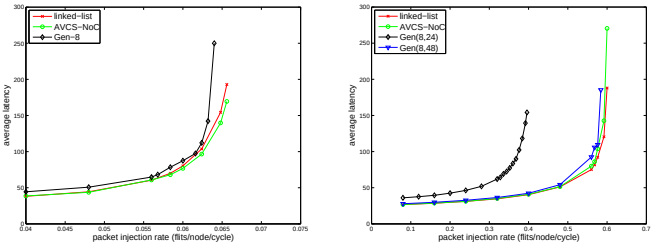
B. Performance Comparison

1) Synthesis Traffic: For the typical router architecture, we suppose each input port has 8 VCs and 24 buffer slots, denoted as Gen(8,24). The dynamical buffer management router architecture is deployed with 12 VCs per input port. To make a fair comparison, we let each input port of our approach have 10 VCs and the shared buffer have 10 VCs in



(a) 30 buffer slots per port (b) 12 VCs per port

Fig. 10: Hardware Overhead Comparison



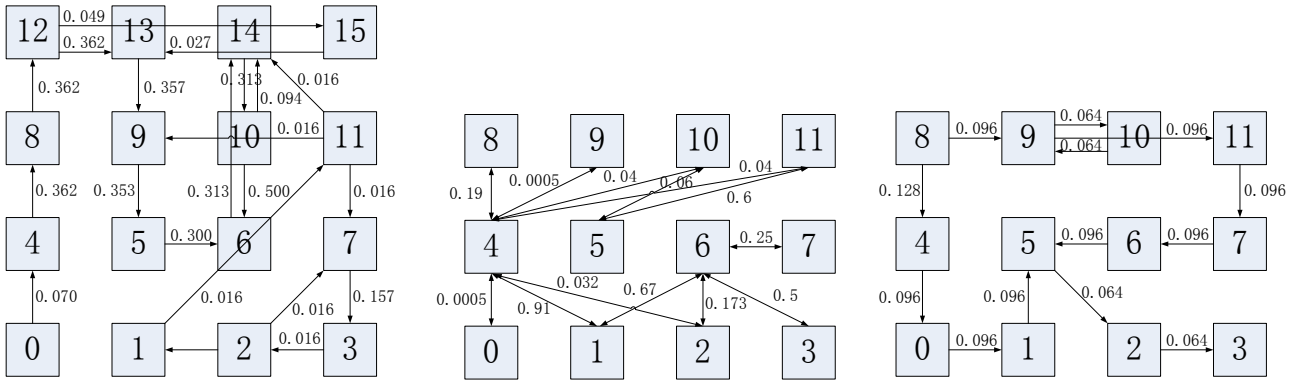
(a) hotspot traffic pattern (b) uniform traffic pattern

Fig. 11: Latency Comparison

total. Throughout this paper, we set the threshold $K=10$. Uniform and hotspots traffic pattern are taken as examples to demonstrate the performance of these three micro-architecture.

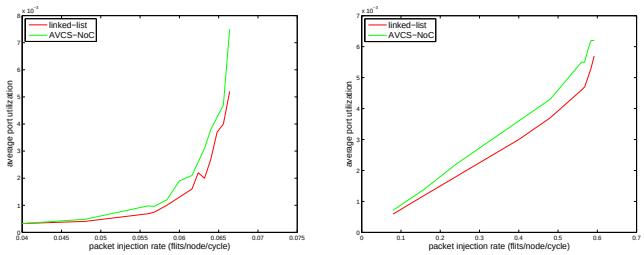
As shown in Fig. 11, we find that our approach improves the performance under hotspot traffic significantly, take the injection rate 0.00656 flits/node/cycle as an example, our router reduce the average latency over the dynamical scheme by 12.16%. For the uniform traffic, our approach achieves the similar performance as the dynamic buffer structure and improves the throughput significantly as compared with the Gen(8,24). The slightly performance degradation when compared with the dynamical scheme lies in the fact that the congestion status of each input port under uniform traffic pattern is similar, which makes the inter-port buffer sharing approach less promising. In addition, we also compared the performance of our router with the other routers under transpose and xxx traffic pattern, the average latency reduction near the saturation point improves about x% and x% over typical router, and x% and y% over the dynamical scheme. For the same configuration, we also compared the utilization of VC allocator. As shown in Fig. 12, our approach indeed improved the allocator utilization significantly. In addition, we also compared the performance of our router with the typical router architecture with twice many buffer slots, denoted as Gen(8,48). As shown in Fig.11b, our router achieves better performance than that of typical router with twice buffer resource.

2) Realistic Traffic: For the realistic application, the workload at each input port is usually different and vary with time, which makes our approach achieve better performance than the typical router architecture and dynamical buffer scheme. We take three real-world applications (Multi-Window Display



(a) Video Object Plan Decoder (b) MPEG4 Decoder (c) Multi-Window Display

Fig. 13: Task mapping of three applications on mesh topology and corresponding injection rate (flits/cycle)



(a) hotspot traffic pattern (b) uniform traffic pattern

Fig. 12: VC Allocator Utilization Comparison

Configuration	MPEG4 decoder	VOPD	MWD
Network topology	4×3 mesh	4×4 mesh	4×3 mesh
Routing algorithm	DOR	DOR	DOR
VCs number	2	6	2
Mapping policy	x	NMAP [16]	x
Buffer size	20 flits	20 flits	20 flits
Packet length	8 flits	8 flits	8 flits
Flit width	64 bits	64 bits	64 bits

TABLE III: Simulation configuration

(MWD) [14], Video Object Plan Decoder (VOPD) [15] and MPEG4 decoder [15]) as examples to verify our deduction. These applications are mapped to 4×3, 4×4 and 4×3 mesh NoC, as shown in Fig. 13a. The communication rate is labelled on the channel connecting two routers. For the given configuration listed in Table III, we run the simulation under three router architectures. We found that, for the typical router architecture, the average latency of VOPD application is one-magnitude-order larger than that of our approach. For the given three applications, our approach improves the linked-list based scheme by 9%, x% and x%, respectively. In addition, we also find that the average latency of VOPD application on typical router with twice larger buffer space is as similar as that of our approach. Thus, For the 64bit flit width, the total hardware saving of our approach is about x% although the control logic introduces x% hardware overhead.

C. Power and Area Comparison

The detailed RTL implementation of AVCS-NoC and the other two router micro-architectures are synthesized with Synopsys Design Compiler under FreePDK 45nm Standard Cell Library [17]. We configure the medium effort optimization option for the synthesis tool and set the working voltage and frequency to 1.1V and 200MHz. For the dynamically managed buffer micro-architecture, the power consumption with the assumed switching activity factor 10% is 83.927mW , chip area is $622744.1875\mu^2m$. In contrast, our approach consumes 97.456mW and 730190.5625 μ^2m under the same conditions, which saves x% chip area and introduce x% power overhead compared with the dynamical scheme. We also compared the power and area overhead of our approach with the typical router micro-architecture with twice more buffer space since they achieve the similar performance under the synthesis and realistic traffic pattern. The power consumption and chip area of the typical router is 112.291mW and $662987.9537\mu^2m$, which occupies x% more chip area and consumes y% power than our approach.

Finally, we compared the chip area and power consumption of our VC and switch allocator with the baseline routers. Supposing each router has P input and output ports, each port has V VCs, then a $PV \times PV$ VC allocator and a $PV \times P$ switch allocator are required for the typical router micro-architecture, while our approach use a $5/6PV \times 5/6PV$ VC allocator and a $5/6PV \times P$ switch allocator. We let $P = 5$ and change V from 6 to 24 with step 6, the power and chip area of these two schemes are listed in Table IV. As shown in this table, our approach saves more chip area than the typical allocator structure. In addition, our approach tends to save more power consumption than the baseline allocator structure when the more VCs are supported.

V. Conclusion and Future Works

The buffer organization and management scheme is a key factor that determines the router performance. In this paper, we propose a runtime adaptive buffer and VC sharing approach, which dynamically allocates the shared buffer and VCs to

#VC	Area(μm^2)			Power(mW)		
	Baseline	AVCS	AVCS vs. Baseline	Baseline	AVCS	AVCS vs. Baseline
6	62013.8828	58659.8906	↓ 5.6%	5.846	6.292	↑ 7%
10	221112.5312	206178.4062	↓ 6.0%	14.916	15.276	↑ 2%
15	492946.6562	426018.5625	↓ 13.57%	26.962	24.870	↓ 7.75%
20	882065.3750	822519.5000	↓ 6.75%	38.999	38.732	↓ 0.68%

TABLE IV: Chip area and power consumption comparison with baseline allocator

the desired input port to better adapt the traffic condition and improve the network performance. Compared with the existing dynamical buffer sharing approach, our method introduces the lest hardware overhead and further improve the performance. Another contribution of this paper is the proposal of shadow VC, which use a smaller allocation to perform the VC and switch allocation. By applying this sharing scheme, the port utilization of each allocation is greatly improve, and the chip area and power overhead reduce significantly. The synthesis results show that, the hardware cost of our method is as same as the linked-list based method, but improve network performance under hotspot traffic pattern by x%. To achieve the similar performance with the typical static router architecture, our approach saves x% power and y% chip area. For the future work, we plan to applying the network calculus theory to this architecture, and realize the fault tolerant and QoS isolation.

ACKNOWLEDGEMENT

The authors thank the reviewers for their suggestions and comments, and all the experiments are carried out at the Integrated Microsystem Lab (IML) of McGill University. This research is supported by High Technology Research and Development Program of China (Grant No. 2012AA012201, 2012AA011902).

CONFLICT OF INTERESTS

The authors declare that there is no conflict of interests regarding the publication of this paper.

REFERENCES

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

[2] Li Shang, Li-Shiuan Peh, and N.K. Jha. Power-efficient interconnection networks: Dynamic voltage scaling with links. *Computer Architecture Letters*, 1(1):6–6, January 2002.

[3] Xuning Chen and Li-Shiuan Peh. Leakage power modeling and optimization in interconnection networks. In *Proceedings of the 2003 international symposium on Low power electronics and design*, pages 90–95, Seoul, Korea, Aug. 2003.

[4] 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. In *Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 333–346, Dec. 2006.

[5] Mingche Lai, Zhiying Wang, Lei Gao, Hongyi Lu, and Kui Dai. A dynamically-allocated virtual channel architecture with congestion awareness for on-chip routers. In *Design Automation Conference, 2008. DAC 2008. 45th ACM/IEEE*, pages 630–633, June 2008.

[6] M.H. Neishaburi and Z. Zilic. Eravc: Enhanced reliability aware noc router. In *Quality Electronic Design (ISQED), 2011 12th International Symposium on*, pages 1–6, March 2011.

[7] Heying Zhang, Kefei Wang, Yi Dai, and Lu Liu. A multi-vc dynamically shared buffer with prefetch for network on chip. In *Networking, Architecture and Storage (NAS), 2012 IEEE 7th International Conference on*, pages 320–327, June 2012.

[8] M. H. Neishaburi and Zeljko Zilic. Reliability aware noc router architecture using input channel buffer sharing. In *Proceedings of the 19th ACM Great Lakes Symposium on VLSI*, GLSVLSI '09, pages 511–516, New York, NY, USA, 2009. ACM.

[9] Jin Liu and José G Delgado-Frias. A shared self-compacting buffer for network-on-chip systems. In *Circuits and Systems, 2006. MWSCAS'06. 49th IEEE International Midwest Symposium on*, volume 2, pages 26–30. IEEE, 2006.

[10] Wan-Ting Su, Jih-Sheng Shen, and Pao-Ann Hsiung. Network-on-chip router design with buffer-stealing. In *Design Automation Conference (ASP-DAC), 2011 16th Asia and South Pacific*, pages 160–164, Jan 2011.

[11] K. Latif, A-M. Rahmani, Liang Guang, T. Seceleanu, and H. Tenhunen. Pvs-noc: Partial virtual channel sharing noc architecture. In *Parallel, Distributed and Network-Based Processing (PDP), 2011 19th Euromicro International Conference on*, pages 470–477, Feb 2011.

[12] M. Evripidou, C. Nicopoulos, V. Soteriou, and Jongman Kim. Virtualizing virtual channels for increased network-on-chip robustness and upgradeability. In *VLSI (ISVLSI), 2012 IEEE Computer Society Annual Symposium on*, pages 21–26, Aug 2012.

[13] Daniel U. Becker, Nan Jiang, George Michelogiannakis, and William J. Dally. Adaptive backpressure: Efficient buffer management for on-chip networks. In *ICCD*, pages 419–426. IEEE Computer Society, 2012.

[14] D. Bertozzi, A Jalabert, S. Murali, R. Tamhankar, S. Stergiou, L. Benini, and G. De Micheli. Noc synthesis flow for customized domain specific multiprocessor systems-on-chip. *Parallel and Distributed Systems, IEEE Transactions on*, 16(2):113–129, Feb 2005.

[15] Erik B. Van Der Tol and Egbert G. T. Jaspers. Mapping of mpeg-4 decoding on a flexible architecture platform. In *Proc. SPIE*, volume 4674, pages 1–13, Dec. 2001.

[16] S. Murali and G. De Micheli. Bandwidth-constrained mapping of cores onto noc architectures. In *Design, Automation and Test in Europe Conference and Exhibition, 2004. Proceedings*, volume 2, pages 896–901 Vol.2, Feb 2004.

[17] Nangate open cell library. http://www.nangate.com/.