AVCS: Adaptive Virtual Channel Sharing Router Architecture for Networks-on-Chip

Baoliang Li, Zeljko Zilic, Wenhua Dou,

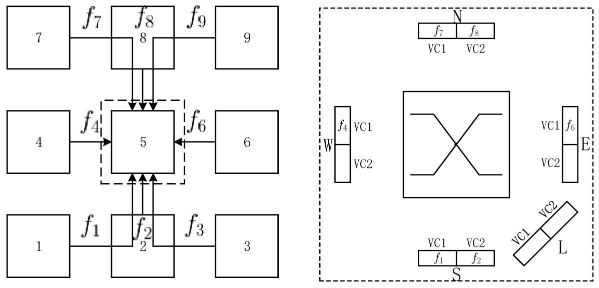
***Abstract*—The performance and hardware overhead of wormhole-switched Networks-on-Chip (NoC) router heavily depend on the organization and management scheme of input buffer. In order to achieve high performance without introducing significant power and area burden, the management of buffer resource should adapt the dynamical change of traffic load. In this paper, we propose an inter-port Adaptive Virtual Channel (VC) Sharing router architecture, i.e. AVCS, 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 and switch allocation. Based on a detailed RTL implementation, we evaluate the performance and hardware overhead of our proposal. Experimental results show that the proposed 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 dynamical buffer management scheme. 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 [1], all the input ports have the same number of VCs and each VC is designated a fixed buffer capacity. Although this static structure eases the implementation, the buffer resources can not be fully utilized under some nonuniform traffic pattern. For the input ports with low traffic load, few buffer resources are enough to temporarily accommodate the incoming packets that cannot be forwarded immediately, leaving the other buffer resources within the same input port under utilized. In contrast, many buffer resources are necessary for the input ports with high traffic load to guarantee throughput. Since the buffer consumes more than 60% power and chip area in a typical 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 router 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].



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

Fig. 1: Blocking caused by VC insufficiency

To improve buffer utilization, these approaches dynamically allocate the buffer slots within an input port to each VCs according to the traffic conditions, which can achieve 50% buffer saving without degenerating the network performance [4]. In [7][8], inter-port buffer sharing scheme was proposed to further improve the performance and buffer utilization of NoC router. Although all these approach employ the dynamical buffer management, the number of VCs each input port can use is fixed. We find that, under some scenarios, 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 special cases, 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 North (N) and South (S) of router 5. Whereas, 4 VCs in total at input port West (W), East (E) and Local (L) are idle. This example reveals that only sharing buffer resources among VCs and ports is not enough to maximize the performance of wormhole-switched NoC router, because the limited VC resources restrict the usage of buffer.

Since the number of VC an input port required at any time instance is equal to the number of partial packets it accommodates, the VC resources required at each input port changes dynamically with time and traffic pattern. To avoid the blocking caused by VC insufficiency and guarantee the worst-case VCs requirement, an intuitive solution is that reserving large amount of VCs at each input. 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

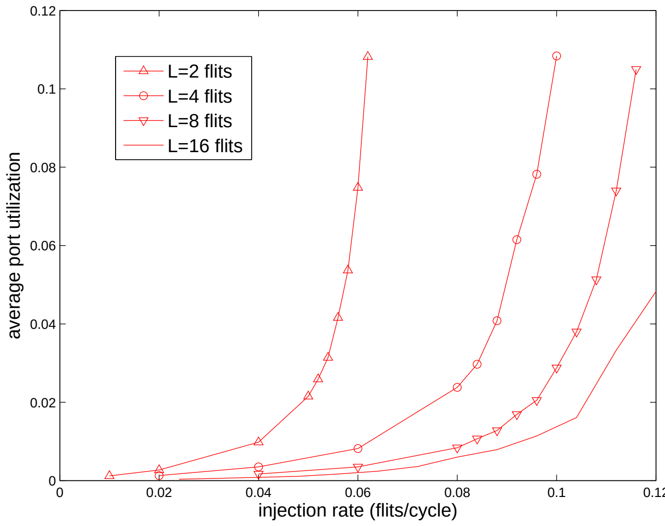


Fig. 2: The port utilization of VC allocator under different injection rate and packet length.

noticed that, the average port utilization of VCA and SWA in the typical router is very low. For an $N \times N$ mesh network, supposing each router has P input and output ports, and each port has V VCs. Then, an VCA matches $P \times V$ request to $P \times V$ VC resources. We can define the average utilization of VCA as

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

where $SampleCycles$ is the duration of sample period and $ActiveCycles_{n,p,v}$ is the number of cycles that the vth VC at th pth input port of router n request for an output VC. 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 average utilization of VCA under different injection rate. As indicated in Fig. 2, for the same injection rate, the average utilization decreases significantly while increasing the packet length. Even near the saturation point of each configuration, the average utilization of VCA is less than 10%. This observation motivates us to design new VC and switch allocator to improve the utilization and meet the same arbitration demands with lower hardware cost.

In this paper, we propose an inter-port Adaptive VC and buffer Sharing router architecture (AVCS), which improves network performance and alleviates the blocking caused by both flow control and VC insufficiency by sharing VC and buffer resources among different input ports. Our approach reserves some buffer and VC resources for each input port and leaves the other VC and buffer resources shared by all the input ports. Under low traffic load, the private buffer and VCs are enough to hold all the incoming packets. While under high traffic load, the shared VCs 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 propose the idea of request port sharing scheme to reduce the complexity of VCA and SWA, which multiplexes the request ports of VCA and SWA between shared VCs and private VCs. This scheme allows us to employ small-scale VCA and SWA 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 input buffer accounts for a large fraction of the overall area and power budget of typical NoC router [2][3]. Thus, reducing the buffer capacity and improving the buffer utilization is essential to implement a high-performance NoC without introducing significant hardware overhead. A proper buffer organization and management scheme is necessary to increase the buffer utilization of wormhole-switched NoC router. There have been significant works focusing on this issue, and several solutions have been proposed.

Virtual Channel Regulator (ViChaR) proposed in [4] utilizes a table-based Unified Buffer Structure (UBS) to dynamically allocate the buffer slots for each VC according to the traffic condition. Although the buffer utilization and network performance are greatly improved, it introduces significant power burden due to the introduction of additional control logic. The router architecture based on Dynamically Allocated Multiple Queue (DAMQ) [9] can provide better performance than typical router while keep the hardware cost low. To eliminate the additional three-cycle delay of conventional DAMQ structure, a prefetch mechanism was proposed in [6]. In [5], the authors proposed a Linked-List based buffer Structure (LLS) and a congestion avoidance scheme to improve the network performance and reduce the power and area cost of NoC. An VC renaming mechanism was proposed in [10] 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 [11].

In [7][8], 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, which introduces significant hardware overhead [12]. Accordingly, a similar inter-port buffer-stealing scheme for the normal router without VC was proposed in [13]. To alleviate the addition control overhead of fine-grained buffer sharing, a Partial Virtual Channel Sharing NoC (PVS-NoC) was proposed in [14], which shares some FIFO buffer between neighbor input port.

The common characteristic of all these proposals is that they reserves large amount of VCs to guarantee the worst-case performance and share the buffer resources among VCS or input ports. Whereas, these statically allocated VC scheme introduces significant hardware overhead and the utilization of VCA and SWA is very low. In addition, under some real-world traffic pattern, the uniformly distributed VC architecture is inefficient because it can not adapt the dynamical change of network traffic. Thus, only combining the buffer sharing and VC sharing can we implement a high-performance NoC router without introducing significant 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 router to reduce the Head-of-Line (HoL) blocking and deadlock. To avoid packets interleaving, a VC can only be reallocated when the tail flit of previous packet has been completely transmitted to the downstream router. If all the VCs have been occupied, blocking occurs because the packets from upstream router can not find an available VC at this router. The degradation caused by VC insufficiency is much serious than that of flow control, especially when the packet is very long. One solution to this problem is that reserving large amount of VCs for each input port, e.g. [4][5][8][7][6]. Whereas, it makes the VCA and SWA 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 on demand according to the traffic condition and the VC status of each input port.

While designing the inter-port buffer and VC sharing scheme, the following four issues should be considered:

1) Interference caused by buffer sharing: When the buffer resources are shared among input ports, an adversarial workload injected from one input port might monopolize all the buffer space of the other input ports and cause global congestion. Thus, special care should be taken to prevent the blockage from spreading to the other ports.
2) Read/Write conflicts caused by buffer sharing: Since each buffer can be accessed by multiple input ports, potential conflicts might occur when multiple input ports trying to access the same buffer. Thus, additional read/write control logic and scheduling policies 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, it introduces additional hardware overhead. 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 VCA and SWA: 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 VCs, which is larger than that of typical router. Thus, special care should be taken while designing these two allocators, since they occupy large chip area and restrict the frequency of router.

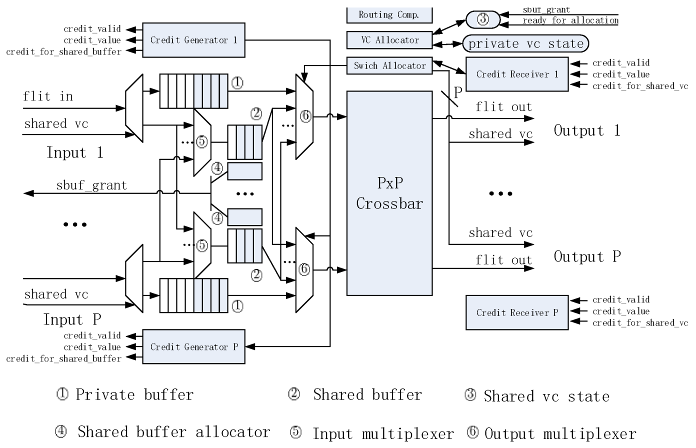
B. Proposed Micro-architecture

Taking all these aspects into consideration, we proposed the Adaptive VC Sharing NoC router architecture (AVCS), as shown in Fig. 3a. This proposal realizes partial buffer and VC sharing: each input port contributes a small portion of their buffer and VC resources to form shared buffer banks. All these shared buffer banks have an unique bank ID, which can be accessed by all the input ports of a router. For an AVCS 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), which grants a new owner according to the traffic condition of each input port whenever this bank keeps idle for several cycles.

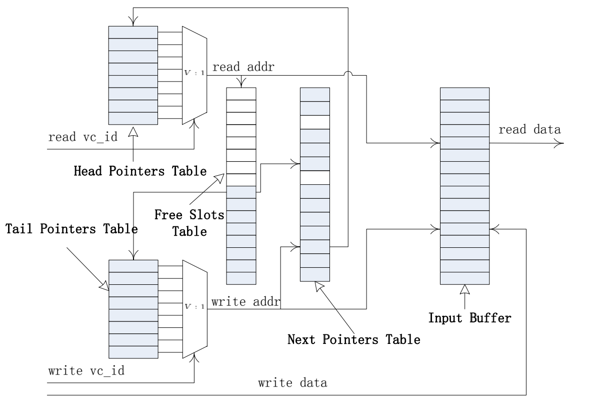
After the buffer allocator determines the new owner of shared bank, the allocation results are sent to the upstream routers by raising the sbuf_grant signal. Each output port of neighbor router updates the shared VC state upon receiving the signal. The shared VCs are allocated at the granularity of bank: Only when a shared buffer bank is granted to an input port, the corresponding shared VCs can be utilized by this input port. The SWA schedules the flits located at both shared VCs and private VCs 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 or private buffer according to the shared_vc signal. Each shared buffer bank utilizes a multiplexer (⑤ in Fig. 3a) controlled by the 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 SWA to select the flits from either private buffer or shared buffers. Although we can use a higher radix crossbar, e.g. $V_{max} \times P$, to achieve higher performance, the hardware overhead is much higher than our proposal.

The credit-based flow control requires a credit receiving module at the output port of upstream router and a credit generating module at the input port of downstream router. Since the private buffer and shared buffer are managed independently in our AVCS router, a dedicated credit counter for each shared buffer bank is necessary. To distinguish whether the credit is for the shared VCs, 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, as shown in Fig. 3a.

To improve the performance with smaller buffer space, we must manage and allocate the buffer resource dynamically according to the traffic conditions. Two methods has been proposed to achieve this aim, e.g. Unified Buffer Structure (UBS) [4][8] and LLS [5][7]. Since the UBS is only applicable to the fixed-length packet network, we adapt the LLB scheme for both shared memory bank and the private buffer in our approach to increase the buffer utilization. Five tables are necessary to implement the LLB 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

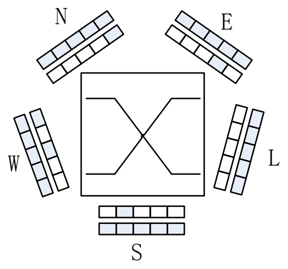


(a) Proposed AVCS-NoC micro-architecture.

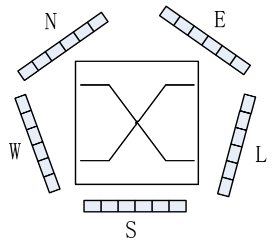


(b) The internal structure of shared/private buffer

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.

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 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 and VC number V, each input port of our AVCS router reserves $5/6B$ slots as private buffer and shares $B/6$ slots with the other input port. Meantime, the VC resources within the shared buffer bank, 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. Under low traffic load, the private buffer and VC of each input port is capable of delivering all the incoming traffic. While under high traffic load, if congestion occurred at some input port, the shared buffer of idle port will be designated to the congested port later to alleviate the congestion. The amount of buffer slots each input port can use lies between $5/6B$ and $10/6B$, in contrast to B of typical router. Similarly, the number of VCs each input port can use lies between $5/6V$ and $10/6V$.

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 of our proposal is much lower than the fully-shared scheme, e.g. [7][8], which realizes better trade-off between cost and performance. We will discuss this issue in subsection IV-A.
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 VCA and SWA, we let the shared VCs and private VCs sharing the same request port, as explained in subsection III-D. This approach makes it possible to arbitrate large amount of VCs by using a smaller VCA and SWA, which reduces the complexity of these two allocators 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 shared VC, and a faulty shared VC does not interrupt the the functionality of corresponding private VC.

This AVCS router architecture is different from the previously proposed ViChaR [4], which need fully re-implement the router pipeline. In contrast, our approach augments the typical router architecture [1] by adding a shared buffer allocator for each shared buffer bank and re-implementing the VCA and SWA. The buffer allocator works in parallel with the router pipeline, which imposes no negative effect on the critical path of entire router. We will introduce the implementation of buffer allocator, VCA and SWA in the following two subsections.

C. Shared Buffer Allocation Module

The shared buffer allocator in our AVCS router is an FSM, which has three states: Enable_Allocation(EA), Disable_Allocation(DA) and Change_Allocation(CA), as shown in Fig. 5. The ready_for_allocation signal keeps on valid while in Enable_Allocation state. In this state, the shared buffer and VCs can be used by the input port indicated by the sbuf_grant signal. We say that a shared buffer bank is idle when it is empty and no output VC of upstream router was designated to it. When the shared buffer bank keeps on idle for K cycles, we can infer that this input port does not need this bank temporally. Thus, we can reallocate it to other input ports which require more buffer and VC resources. The threshold K is a design parameter, which affects the allocation results and the performance of entire network. For a real-world application, the optimal K can be chosen by experiments. To avoid the shared buffer is reallocated while changing status, the FSM first enters the Disable_Allocation state. At this state, the ready_for_allocation (RA) signal goes down to disable the output VC allocation at upstream router. When the shared buffer bank is empty and no shared VCs on this bank are 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 allocation period.

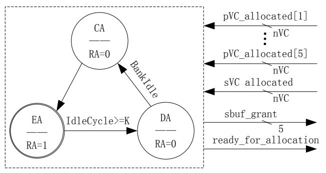


Fig. 5: The FSM of buffer allocator

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

Fig. 6: Port Encoding

To simplify the allocation algorithm, we choose the owner of each shared buffer bank in round-robin order. This is achieved by using a five-bit register port_mask to filter out the ports can not be chosen as candidate at current allocation period (Step 4 and Step 7). Initially, these five buffer banks belongs to different input ports (Step 3 and Step 6). When the FSM enters Change_Allocation state, the five-bit port_busy (generated in Step 11) and port_mask registers are used to determine the new owner of this bank (Step 13 to 21). 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 Fig. 6, the shared buffer is granted to this port. The encoding scheme also indicates that 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.

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
```

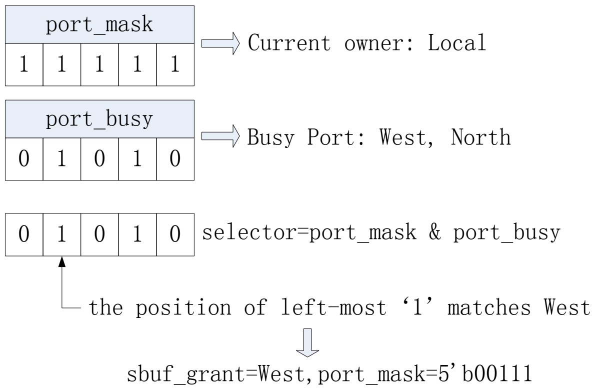

port_mask
1 1 1 1 1
Current owner: Local
port_busy
0 1 0 1 0
Busy Port: West, North
0 1 0 1 0
selector=port_mask & port_busy
the position of left-most '1' matches West
sbuf_grant=West,port_mask=5'b00111

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

D. VC and SW Allocator

The goal of inter-port buffer and VC sharing is to alleviate the performance degradation caused by flow control and 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 different router ports. The VCs each input port can use in AVCS router includes the private VCs owned by this port and the VCs shared among all the ports. Thus, 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 shared buffer banks. Denote the number of input ports and the maximal shared VCs (including private and shared VCs) each input port can use as P and V_{max}. The VCA matches $P \times V_{max}$ requests from P input ports with $P \times V_{max}$ output VCs. Similarly, the SWA 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 $PV_{max} \times PV_{max}$ VCA and a $PV_{max} \times P$ SWA. However, this makes these two allocators complex and introduce significant hardware overhead.

In this paper, we employ another approach. Our approach lies on two properties of AVCS router: (1) The maximum number of shared VCs in AVCS-NoC is equal to the number of private VCs owned by each input port, and (2) the minimum number of shared VCs each input port can use is zero. Thus, the actually available VCs of 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 for the request ports with private VCs. Then, we can use a $1/2PV_{max} \times 1/2PV_{max}$ VCA and a $1/2PV_{max} \times P$ SWA to perform the VC and switch allocation. The reason we employ a smaller VCA and SWA with port sharing rather than a larger allocator lies in the fact that the port utilization of VCA and SWA is very poor, especially for the VCA, as demonstrated in Fig. 2. This is because only the head flit of a packet can issue an VC allocation request and the corresponding request port will keep on idle until the tail flit departure from this VC. The idle request port can be reused to improve the utilization.

Specifically, when a shared buffer is designated to an input port, the elig_svc flags of all the corresponding shared VCs are asserted to allow the granting of a request. To enable the sharing of request port, we add a $2:1$ selector to each request port to choose a request from either private VC or shared VC, as shown in Fig. 8. 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 [15] 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 an unique input VC. We will discuss the power and area improvement of our proposal over the existing method in subsection IV-C.

Similarly, for the SWA, if one of the VC is blocked due to some reasons, the corresponding request port can be reused by the shared VC. As shown in Fig. 9, we first use a $2:1$ arbiter to select on request from either shared VC or private VC at the first stage. The selection is performed in round-robin order to avoid livelock. 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. 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-C.

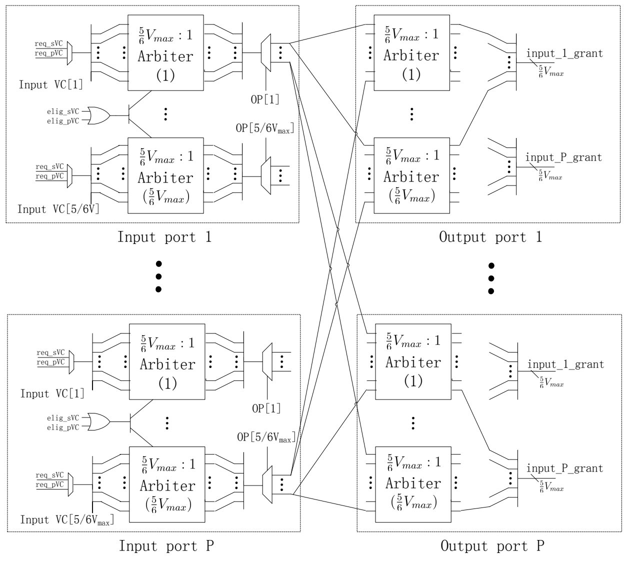


Fig. 8: The Implementation of VCA

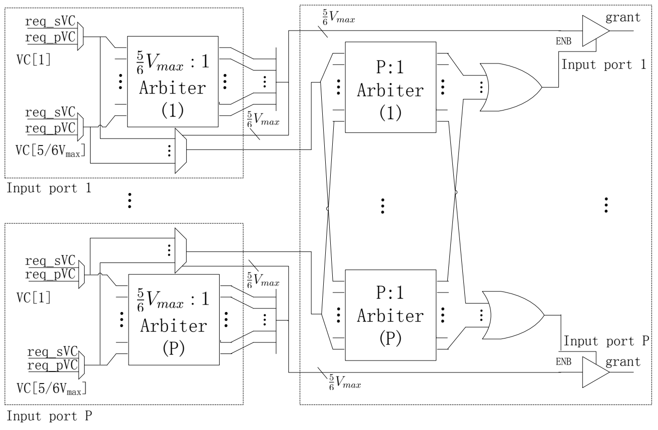


Fig. 9: The Implementation of SWA

IV. Experimental results

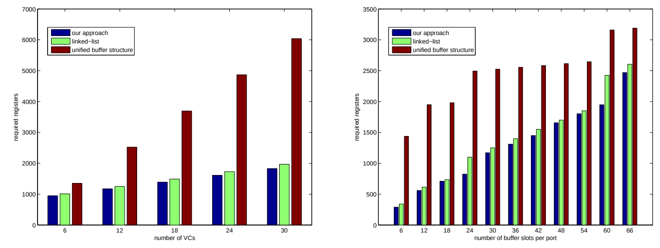
To compare the performance and hardware overhead of our approach with the existing proposals, we implement the AVCS router, the typical router [1] and the dynamical buffer management architecture [5] in Verilog HDL. The comparison results on hardware cost, performance, power and chip area are presented in the following three subsections.

A. Control Overhead Comparison

Both UBS [4][8] and LLB [5][7] require large amount of control logic to keep track of the buffer usage. For the inter-port buffer sharing scheme [7], the hardware overhead is much higher than the intra-port sharing scheme since the Next

TABLE I: Hardware Overhead Comparison

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



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

Fig. 10: Hardware Overhead Comparison

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][8][5][7] has B buffer slots and V VCs at each input port, 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, denote by L the packet length. Under the same condition, 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 LLB [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 I.

To make concrete comparison, for the same packet length $L = 8$ flits, we plot the hardware overhead of each scheme listed above under different VC number and buffer capacity in Fig. 10. As shown in this figure, our approach uses the lest control logic, because we divide the buffer resources into several banks, and each bank manages their own buffer resource independently. The total hardware overhead is greatly reduced since the addressing space of each buffer bank is smaller than the other two approach. 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%, in contrast with 7.16% and 16.24% of LLB and UBS approach, 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*

In this subsection, we compare the performance of our architecture with the typical router architecture [1] and the dynamical buffer management router architecture [4][5] under synthesis traffic pattern and realistic traffic pattern.

1) Synthesis Traffic: Hotspot, uniform and transpose traffic pattern are taken as examples to demonstrate the performance improvement of our proposal over the other two kinds of architectures. The configuration used in our experiment are presented in Table II. For the typical router architecture, two configurations, i.e. Gen(8,24) and Gen(8,48), are used in our comparison according to their VCs number and buffer capacity. The dynamical buffer management router architecture is deployed with 12 VCs and 24 buffer slots per input port, denoted as DBM(12,24). To make a fair comparison, we let each input port of our approach have 20 buffer slots and 10 VCs. The shared buffer have 20 buffer slots and 10 VCs in total, organized as five banks. In this experiment, we set the threshold $K = 10$.

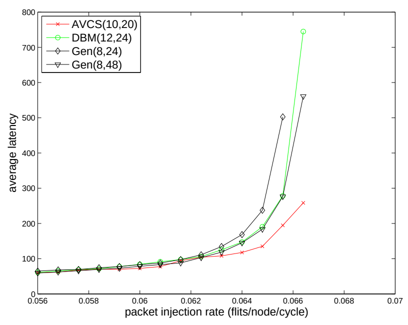
TABLE II: Configuration used in the experiments

network topology	4×4 mesh	routing algorithm	DOR
flit width	256 bits	warmup period	1×10^4 cycles
packet length	8 flits	sampling period	1×10^6 cycles

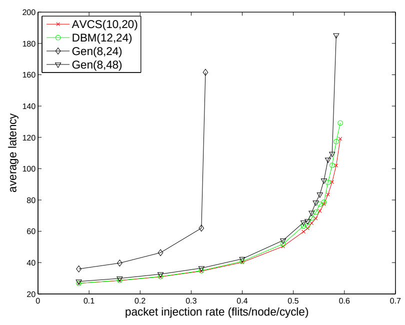
As shown in Fig. 11, we find that our approach improves the performance of typical router and dynamical buffer management router under hotspot traffic significantly, take the injection rate 0.00656 flits/node/cycle as an example, our router reduces the average latency over DBM(12,24), Gen(8,24) and Gen(8,48) by 30.1%, 61.3% and 29.6%, respectively. For the uniform traffic, our approach achieves the similar performance as DBM(12,24) and Gen(8,48), but improves the performance of Gen(8,24) significantly. The inter-port VC sharing does not achieve too much gain, because the congestion status of each input port under uniform traffic pattern is similar, which makes the inter-port buffer sharing less promising.

2) Realistic Traffic: For the realistic applications, the workload of each router port is different and changes with time. Our proposal can adapt this dynamical change and allocate more buffer and VC resources to meet the resource requirement of input ports with higher traffic load. This property makes our approach achieves better performance than the other two router architectures. We take three real-world applications (Video Object Plan Decoder (VOPD) [16], MPEG4 decoder [17] and Multi-Window Display (MWD) [18]) as examples to compare the performance of our proposal with Gen(8,24), Gen(8,48) and DBM(12,24). These three applications are mapped to 4×3, 4×4 and 4×3 mesh NoC, respectively, as shown in Fig. 12. The communication rate (flits/cycle) is labelled on the channel connecting two routers.

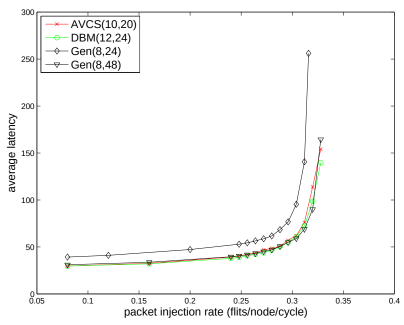
For the given configuration listed in Table III, we run the simulation with the detailed RTL implementation under three router architectures for all these applications. The experimental presented in Table IV results show that, AVCS can achieve better performance than LLB(12,24), Gen(8,24) and Gen(8,48)



(a) hotspot traffic pattern

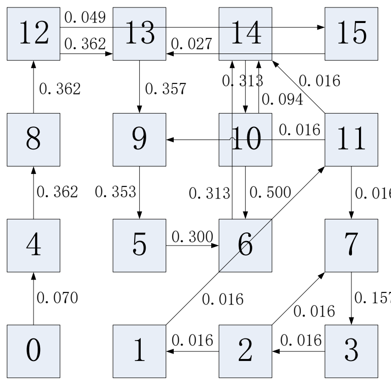


(b) uniform traffic pattern

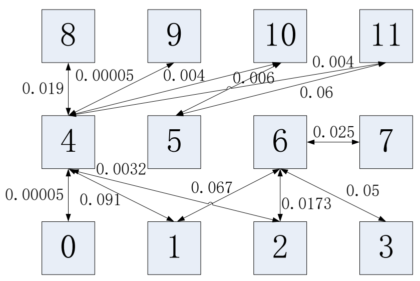


(c) transpose traffic pattern

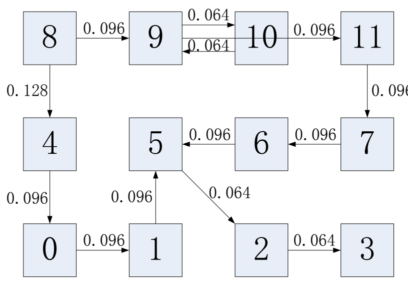
Fig. 11: Latency comparison



(a) Video Object Plan Decoder



(b) MPEG4 Decoder



(c) Multi-Window Display

Fig. 12: Task mapping of three applications on mesh topology and corresponding injection rate (flits/cycle)

TABLE IV: Average latency comparison (in cycles)

	AVCS(10,20)	Gen(8,24)	Gen(8,48)	DBM(12,24)
VOPD	29.3805	606.31 (↓ 91.5%)	29.9462 (↓ 1.9%)	29.5593 (↓ 0.6%)
MPEG4	25.4454	32.0747 (↓ 20.7%)	25.4935 (↓ 0.2%)	25.4578 (↓ 0.04%)
MWD	22.4489	30.0246 (↓ 25.2%)	22.5405 (↓ 0.4%)	22.4777 (↓ 0.1%)

TABLE III: Simulation configuration

Network topology	4×4 mesh	routing algorithm	DOR
Packet length	10 flits	threshold	1 cycle
warmup period	1×10^4 cycles	sampling period	1×10^5 cycles

significantly. This is because our structure improve the buffer utilization and reduce the blocking caused by VC insufficiency, which makes it possible to achieve better performance by utilizing fewer buffer space.

C. Power and Area Comparison

The detailed RTL implementation of AVCS and typical router are synthesized with Synopsys Design Compiler under FreePDK 45nm standard cell library [19]. We configure the medium effort optimization option for the synthesis tool and set the working voltage and frequency to 1.1V and 200MHz. For AVCS(10,20) router, the power consumption with the assumed switching activity factor 10% is 141.445 mW, chip area is 1002486.625 $\mu^2 m$. In contrast, Gen(8,48) consumes 208.224 mW and 1135084.5 $\mu^2 m$ under the same conditions, which saves 11.7% chip area and reduces 32.1% power consumption. We compared the power and area overhead of AVCS(10,24) with Gen(8,48) since it achieves the similar performance under both synthesis and realistic traffic patterns.

Finally, we compared the chip area and power consumption of our revised VCA and SWA with that of used in the typical router. Supposing each router has P input and output ports, each port has V VCs, then a $PV \times PV$ VCA and a $PV \times P$ SWA should be deployed in the typical router micro-architecture, while our approach use a $5/6PV \times 5/6PV$ VCA and a $5/6PV \times P$ SWA instead. 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 V. 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.

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

#VC	Area(μm^2)			Power(mW)		
	Baseline	AVCS	AVCS vs. Baseline	Baseline	AVCS	AVCS vs. Baseline
6	62013.8828	58659.8906	$\downarrow$ 5.6%	5.846	6.292	$\uparrow$ 7%
12	221112.5312	206178.4062	$\downarrow$ 6.0%	14.916	15.276	$\uparrow$ 2%
18	492946.6562	426018.5625	$\downarrow$ 13.57%	26.962	24.870	$\downarrow$ 7.75%
24	882065.3750	822519.5000	$\downarrow$ 6.75%	38.999	38.732	$\downarrow$ 0.68%

V. Conclusion and Future Works

The buffer organization and management scheme is a key factor which determines the performance of wormhole-switched NoC router. In this paper, we propose a runtime adaptive buffer and VC sharing approach, which dynamically allocates the shared buffer and VCs to the desired input ports to better adapt the traffic condition and improve the network performance. Comparing with the existing DBM approach, our method introduces the lest hardware overhead and further improve the network performance. Another contribution of this paper is that we proposed the idea of request port sharing, which uses a smaller VCA and SWA to perform the VC and switch allocation. By applying this sharing scheme, the chip area and power overhead of these two components are reduced significantly. The synthesis results show that, the hardware cost of our method is as same as the LLB, but improve network performance under hotspot and uniform traffic pattern by x% and y%. To achieve the similar performance with the typical 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] 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.

[7] 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.

[8] 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.

[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] 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.

[11] 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.

[12] Sungho Park. A verilog-hdl implementation of virtual channels in a network-on-chip router. Master's thesis, Texas A&M University, 2008.

[13] 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.

[14] 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.

[15] William Dally and Brian Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

[16] A.T. Tran and B.M. Baas. Achieving high-performance on-chip networks with shared-buffer routers. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 22(6):1391–1403, June 2014.

[17] 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.

[18] 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.

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