

An Adaptive Physical Channel Regulator for High Performance and Low Power Network-On-Chip Routers

Lei Wang Poornachandran Kumar Ki Hwan Yum
Eun Jung Kim

Department of Computer Science and Engineering
Texas A&M University
College Station, TX, 77843 USA
{wanglei, poorna, yum, ejkim}@cse.tamu.edu

August 20, 2010

Abstract

Chip Multi-Processor (CMP) architectures have become mainstream for designing processors. With a large number of cores, Network-On-Chip (NOC) provides a scalable communication method for CMP architectures, where wires become abundant resources available inside the chip. NOC must be carefully designed to meet constraints of power and area, and provide ultra low latencies. In this paper, we propose an Adaptive Physical Channel Regulator (APCR) for NOC routers to exploit the huge wiring resources for high performance and low power. The flit size in an APCR router is no longer equivalent to the physical channel width (phit size) providing finer granularity flow control. An APCR router allows flits from different packets or flows to share the same physical channel in a single cycle. The three regulation schemes (Monopolizing, Fair-sharing and Channel-stealing) intelligently allocate the output channel resources considering not only the availability of physical channels but the occupancy of input buffers. In an APCR router, each Virtual Channel can forward dynamic number of flits every cycle depending on the run-time network status. We also introduce Generalized NOC Router Design (GNRD) – a frame-work for exploring the design space of NOC routers. Our simulation results using a detailed cycle-accurate simulator show that an APCR router improves the network throughput by over 100%, compared with a baseline router design with the same buffer size. An APCR can outperform the baseline router even if the buffer size is halved. Furthermore, an APCR router enjoys over 33% total power savings with a little area overhead.

1 Introduction

Moore's law has steadily increased on-chip transistor density and integrated dozens of components on a single die. Providing efficient communication in a single die is becoming a critical factor for high performance CMPs [1]. Traditional shared buses and dedicated wires do not meet the communication demands for future multi-core architectures. Moreover, the shrinking technology exacerbates the imbalance between transistors and wires in terms of delay, and power has embarked on a fervent search for efficient communication designs [2]. In this regime, Network-On-Chip (NOC) with packet-switching is a promising architecture that orchestrates chip-wide communications towards future many-core processors.

Although interconnection network design has matured in the context of multiprocessor architectures, NOC has different characteristics for chip-wide communication support, making its design unique. NOC can benefit

from high wire density due to no limits on the number of pins and faster signaling rates. However, the cost of NOC is constrained in terms of power and area. In fact, NOC power consumption is considered to be significant since 28% of the tile power in Teraflop [3] and 36% of the total chip power in 16-tile RAW chip [4] are consumed by NOC. As feature size shrinking, there have been a handful studies exploiting abundant wire resources to explore different topologies with high degree channels such as Flattened butterfly [5] and Express cube [6]. However, high radix routers require more buffer resources and complex arbitration, resulting in more power consumption and area overhead. It is very critical in the NOC router design to find a way that fully utilizes the wire resources to provide high performance, while saving power and buffer area. One can suggest to provide a wide transmission channel between routers, which facilitates low latency due to small serialization delay [7, 8, 9]. However, as Figure 1 shows, simply increasing the channel width and defining the flit size the same as the channel width or the phit size do not deliver much performance benefit with the same router buffer budget. Due to the fixed buffer size, increasing flit size will proportionately decrease the buffer depth. Even with wormhole flow control, the performance still suffers degradation. Meanwhile, the majority of on-chip communication emanates from cache traffic, such as cache coherence messages or L1/L2 cache blocks. On one side a coherence message, like the request or invalidation, only consists of a little header and a memory address, which are around 64 bits. On the other side, a packet which carries a L1/L2 cache block is as large as 512 bits (64 Bytes). Diverse packet sizes limit the usage of wide channels.

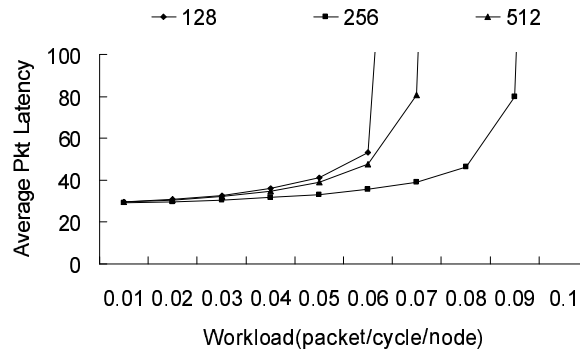


Figure 1: Average Packet Latency with Three (128, 256 and 512 bits) Different Channel Widths and Fixed Router Buffer Sizes in a (8×8) Mesh Network.

The packet latency in NOC is the time required for a packet to traverse the network, which includes router delay, time of flight, serialization latency and contention delay. Serialization latency arises from the organization of packets. The more flits one packet has, the bigger is its serialization latency. In light of this, a big flit size will decrease the packet serialization latency. However, the sharing and multiplexing of physical channels among different source-destination pairs result in the contention delay. It is easier to release the network congestion with a finer granularity flow control by defining a smaller flit size. The trade-off between serialization latency and contention delay is very critical for NOC router design.

In this paper, we propose an Adaptive Physical Channel Regulator (APCR) for NOC routers. Even though there may exist different lengths of packets (short or long) in the same network, an APCR router only defines one constant flit size, which is typically smaller than the channel width. An APCR router allows a virtual channel (VC) to transmit multiple small flits in one cycle. The number of flits being transmitted every cycle dynamically changes according to the network status. We present three regulation schemes for APCR: monopolizing, fair-sharing and channel-stealing. Routers become run-time configurable to different lengths of packets with these three regulation schemes. It is similar to providing routers with diverse flit size definitions to meet the requirement of different lengths of packets. On one hand, long packets with multiple flits can transmit more than one flit in a cycle when no other packets want to use the output channel (This occurs in a low workload network). On the other hand, short packets, normally containing one flit, use only a portion of the output channel and let the other portions to either be utilized by other packets or in lower power state, thus saving link power. Furthermore, APCR allocates the channel resources using the occupancy of input buffers, thoroughly utilizing the wide channel and providing high network throughput. To better understand the design space of NOC routers, we propose Generalized NOC Router Design (GNRD) – a framework that demonstrates how various network parameters affect the NOC router design. Our simulation results show that an APCR router outperforms a baseline router with double the buffer size. When buffer sizes are identical, an APCR router achieves more than 100% throughput improvement without negative effect on zero-load latency. Moreover, an APCR router saves router power consumption by over 33% even with a little area overhead.

The rest of this paper is organized as follows. We briefly analyze the recent work in Section 2. We present the three channel regulation schemes in Section 3. In Section 4 we detail the microarchitecture of an APCR router design. We generalize the NOC router design in Section 5. In Section 6, we describe the evaluation methodology and summarize the simulation results. Finally, we draw conclusions in Section 7.

2 Related Work

Several previous NOC designs have been proposed to explore the abundant channel resources. Work by Das et al [10] looks at multiple flits sharing a channel. In their network, there are two kinds of flits, short and long. The sharing condition is simple: if two short flits are routed to the same output port, they can simultaneously traverse the crossbar and output channel. For long flits, no sharing is applied. One concern about this design is that to support two kinds of flit sizes in the same network, the flow control can be challenging. Since a flit is the basic flow control unit, providing two kinds of flits will also make the credit management complicated. [11] introduces the concept, Spatial Division Multiplexing (SDM), into the NOC design. The results show that SDM is a more interesting approach than Time Division Multiplexing (TDM), due to the high complexity and power needed by buffers to store the TDM configuration for each clock cycle. Nevertheless, their work does

not study the impact on performance and analyze how to share the channel resources in detail. In [12, 13], abundant wire resources are also explored. The main goal of these studies is to provide more path diversity and reduce the inside buffer size. To the best of our knowledge, our work is the first one to introduce an adaptive physical channel regulator into the NOC router design.

3 Adaptive Physical Channel Regulation Schemes

In this section, we primarily elucidate the three regulation schemes used in an APCR Router¹ while Section 4 delves into the detailed microarchitecture of an APCR router.

3.1 Monopolizing

Similar to a generic router, monopolizing allows only one VC to use the total bandwidth of the output channel every cycle. In a generic router design, the flit size is usually the same as the phit size. A VC can fully use the whole bandwidth of the output channel. However, in an APCR router design the flit size is smaller than the phit size. In other words, potentially multiple flits can be in the same channel concurrently. Considering this characteristic, an APCR router allows a VC to transmit multiple flits in the same cycle. There is one restriction on this situation. Wormhole flow control allows different packets stored in a VC without interleaving, and the basic routing unit is a packet not a flit. Therefore, a VC is not allowed to transmit as many flits as it has. The maximum number of transmitted flits depends on how many of flits stored in a VC belong to the same packet, and not on how many flits the output channel can process. For example, in Figure 2 the phit size is four times the flit size, which implies the physical channel can transmit at most four flits in one cycle. Meanwhile, the VC depth is also four, implying each VC can buffer four flits. VC_0 wins the arbitration of the output channel, and it contains four flits—one head flit, two middle flits and one tail flit. The four flits are stored like this way: one middle flit stays at the head of the VC followed by the tail flit belonging to the same packet, but the head flit and the other middle flit belong to a different packet and are located at the tail of the VC. In this case, even though the output channel can transmit four flits at a time, we only allow the first middle flit and the tail flit to be transmitted. The head flit and the second middle flit should wait until the next cycle because they belong to a different packet which may have a different routing direction.

Compared with the generic router design, a smaller flit size can provide finer granularity of traffic control. In Figure 2, if VC_0 has three flits belonging to the same packet, it can forward all three flits together in one cycle, while at another time it may only be able to send one or two flits. An extreme case occurs in which VC_0 always buffers four flits belonging to the same packet. In this scenario VC_0 can transmit four flits every cycle which will occupy the whole bandwidth of the output channel. This situation is similar to defining a big flit

¹In our discussion we assume the downstream router always has enough credits.

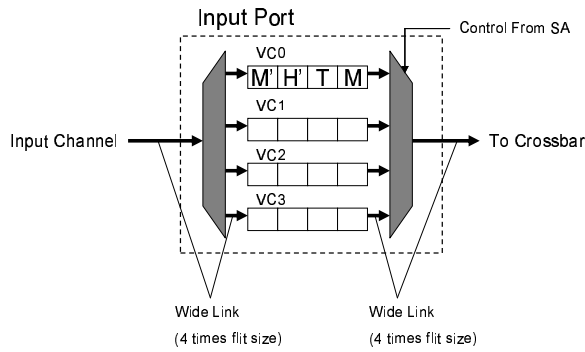


Figure 2: Flit M and T belong to a packet, and flit H' and M' are from another packet. VC₀ wins the arbitration of the output channel. However, only M and T can be transmitted in the same cycle.

size equal to the phit size. Monopolizing allows a VC to transmit different numbers of flits in one cycle. If we treat the term “flit” as the unit which a VC sends every cycle, An APCR router makes the network “flit” size run-time configurable according to the status of the network, and each router can even use different “flit” sizes at the same time. However, monopolizing can also potentially waste the channel bandwidth. Figure 3 shows such a case. The relationship between the phit size and the flit size is the same as in the previous example. There are four VCs for each input port. Each VC has at least one flit to be sent. We assume the flits in each VC routed to the same direction. VC₂ wins the output channel. According to monopolizing, only one quarter of the output channel is utilized even though other VCs have flits waiting to be sent.

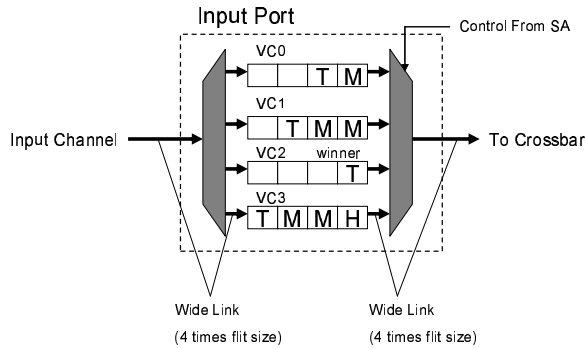


Figure 3: VC₂ wins the arbitration of the output channel. However, VC₂ only has one flit to be sent. It wastes three quarters of the channel bandwidth.

3.2 Fair-sharing

Considering the inefficient channel utilization scenario described above, we propose fair-sharing. VCs fairly share the output channel resources. To achieve fair-sharing, a wide physical channel needs to be divided into several small parts, called sub-channels. We reserve a different sub-channel for VCs of the same input port. VCs of different input ports share sub-channels. Assuming that a physical channel is divided into four sub-channels and each input port has four VCs, then each VC of the same input port can have one sub-channel of its own.

However, VCs of different input ports should share sub-channels. For example, VC_0 uses sub-channel₀, VC_1 uses sub-channel₁ . . . and all the VC_0 s of different input ports share the sub-channel₀. To make the hardware design simple, the ID of a VC is bound with the ID of a sub-channel. Since the width of a sub-channel is fit with a flit, VCs can send one flit concurrently even though they belong to the same input port. If the number of sub-channels is bigger than the number of VCs, each VC can be assigned more than one sub-channel.

Compared with monopolizing, fair-sharing utilizes the wide channel more efficiently. This is because the chance of sending flits from multiple VCs is normally higher than sending from a single VC especially during high workload. Fair-sharing still has its own limitations, as illustrated by an example in Figure 4. VC_0 has two flits, while VC_1 has three, but VC_2 and VC_3 are empty. According to fair-sharing, VC_0 and VC_1 will use sub-channel₀ and sub-channel₁ to send one flit each. Since VC_2 and VC_3 have no flits, sub-channel₂ and sub-channel₃ are empty. Even though VC_0 and VC_1 have more flits to be sent, they cannot use sub-channel₂ and sub-channel₃ because they are bound to other sub-channels.

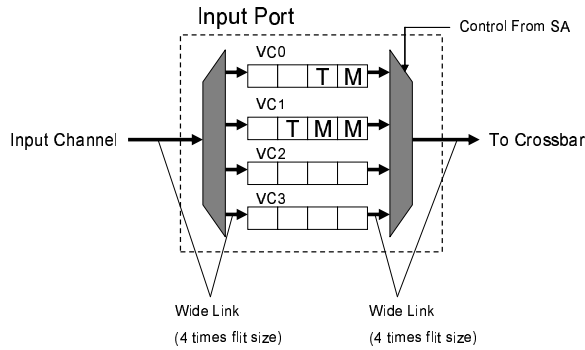


Figure 4: VC_0 and VC_1 can only send one flit each using their own sub-channels. VC_2 and VC_3 waste their sub-channels because they do not have any flits. These wasted sub-channels cannot be used by VC_0 and VC_1 .

3.3 Channel-stealing

To further improve the utilization of wide channels, we propose channel-stealing, which is built upon fair-sharing. Different from fair-sharing, if a VC finally has no flit to be sent, its sub-channel will be stolen by other VCs. Here the stealing occurs in two ways. One is stealing from VCs belonging to the same input port, and the other is stealing from VCs of different input ports which have the same output direction. Channel-stealing explores the channel resources thoroughly. It optimizes the arbitration of output channels by using the buffer occupancy information from each VC and finally increases the network throughput. In Figure 4, VC_2 and VC_3 have no flits to be sent. VC_0 and VC_1 can steal the sub-channel assigned to VC_2 and VC_3 , and send more than one flit. There are two options: Either VC_0 and VC_1 send two flits each or VC_0 sends one flit while VC_1 sends three flits. Choosing from the two options depends on the scheduling policy. In this study, we recruit Round-Robin as our scheduling policy, which fairly allocates the extra free sub-channels to VCs with more

flits.

4 APCR Router Microarchitecture Design

To support the three channel regulation schemes in Section 3, a generic NOC router microarchitecture has to be enhanced. In this section, we first briefly explain a generic NOC router microarchitecture and the functionality of each pipeline stage. Then we propose our APCR router microarchitecture design and analyze each main component.

4.1 A Generic NOC Router

Figure 5 (a) shows a generic NOC router architecture [14] for a 2-D mesh network. In most implementations, there are 5 ports: four from the four cardinal directions (NORTH, EAST, SOUTH and WEST) and one from local Processing Element (PE). The main building blocks are input buffer, route computation logic, VC allocator, switch allocator, and crossbar. To achieve high performance, routers process packets with four pipeline stages, which are routing computation (RC), VC allocation (VA), switch allocation (SA), and switch traversal (ST). First, the RC stage directs a packet to a proper output port by looking up a destination address. Next, the VA stage allocates one available VC of the downstream router determined by RC. The SA stage arbitrates input and output ports of the crossbar, and successfully granted flits traverse the crossbar at the ST stage. Due to the stringent area budget of a chip, routers use flit level buffering in a wormhole-switching network as opposed to packet level buffering. Additionally, a buffer is managed with credit-based flow control, where downstream routers provide back-pressure to upstream routers to prevent buffer overflow. Considering that only the head flit needs routing computation and middle flits always have to stall at the RC stage, low-latency router designs parallelize the RC, VA and SA using lookahead routing [15] and speculative switch allocation [16]. The functionality of lookahead routing is the same as a normal RC stage, calculating the output ports. However, instead of calculating routing information for the current router, lookahead routing does the same for the downstream router and stores the routing information in the head flit. In this way the RC and VA stages can be overlapped because the VC allocator does not need to wait for the output of RC logic. Speculative switch allocation predicts the winner of the VA stage and performs SA based on the prediction. If the packet fails to allocate a VC, the pipeline stalls and both the VA and SA will be repeated in the next cycle. These two modifications lead to two-stage and even single-stage [17] routers, which parallelize the various stages of operation. In this paper, we use a two-stage router as the baseline router.

4.2 APCR Router Microarchitecture

Figure 5 (b) shows the microarchitecture of an APCR router. The differences from a generic router are shaded. The VC allocator is the same as a generic router for the three regulation schemes. Also no modification will

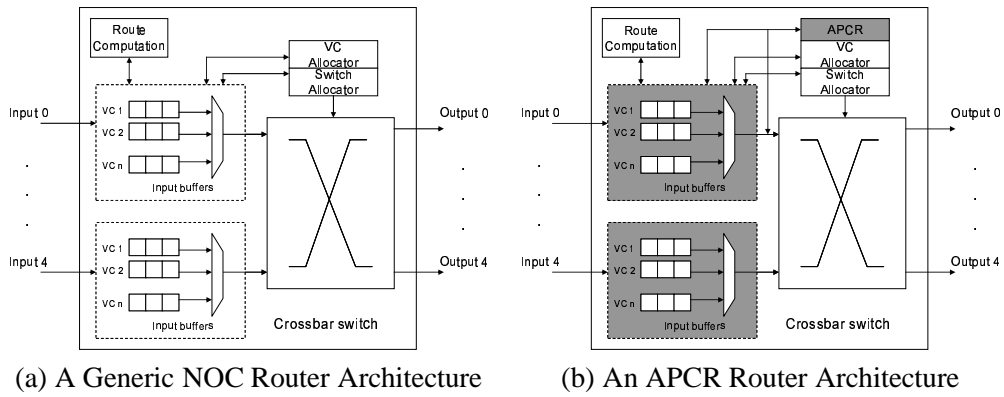


Figure 5: Generic Router and APCR Router Architecture.

be applied to the crossbar. Even with wide channels, the number of input and output ports will be the same (5×5 crossbar for a 2-D mesh topology). The APCR component works together with switch allocator and input buffers. Firstly, according to the characteristics of different regulation schemes, APCR decides the number of flits each VC can send. For example, monopolizing allows a VC to send multiple flits but selects only one VC from the same input port, while in fair-sharing a VC can only send one flit but no competition exists among VCs of the same input port. Secondly, APCR controls the input buffer operations according to the allocation results. In the following we describe the design of each main component in an APCR router. Detailed power and area evaluation will be covered in Section 6.

4.2.1 Buffer Management

With the limitation of power and area, NOC routers rely on simple buffer structures. In the case of VC-based NOC routers, each VC recruits a specified number of FIFO buffers. Figure 6 (a) shows such a organization. Each input port has v virtual channels, each of which has dedicated k -flit FIFO buffer. Recent NOC routers have small input buffers to minimize their overhead. Hence v and k are smaller than in macro networks [17]. The necessity for low latency buffer operation dictates the use of a parallel FIFO as illustrated in Figure 6 (b). As opposed to a serial FIFO implementation [18], the parallel flavor eliminates the need for a flit to traverse all slots in a pipelined manner before exiting the buffer. This fine-grained control relies on read and write pointers to maintain the FIFO order. Figure 6 (c) shows a design of this FIFO buffer. When a read operation occurs, the flit pointed by the head is sent out and the head pointer is automatically moved to the next position. Similarly, when a write operation comes, the flit is written to the position pointed by the tail and the tail pointer is increased by one. When the head is equal to the tail after a read, the buffer is empty. When the tail is equal to the head after a write, the buffer becomes full.

The three regulation schemes (Monopolizing, Fair-sharing and Channel-stealing) require different buffer structures. Input buffers for Fair-sharing is the same as that of a generic router since every cycle only one flit

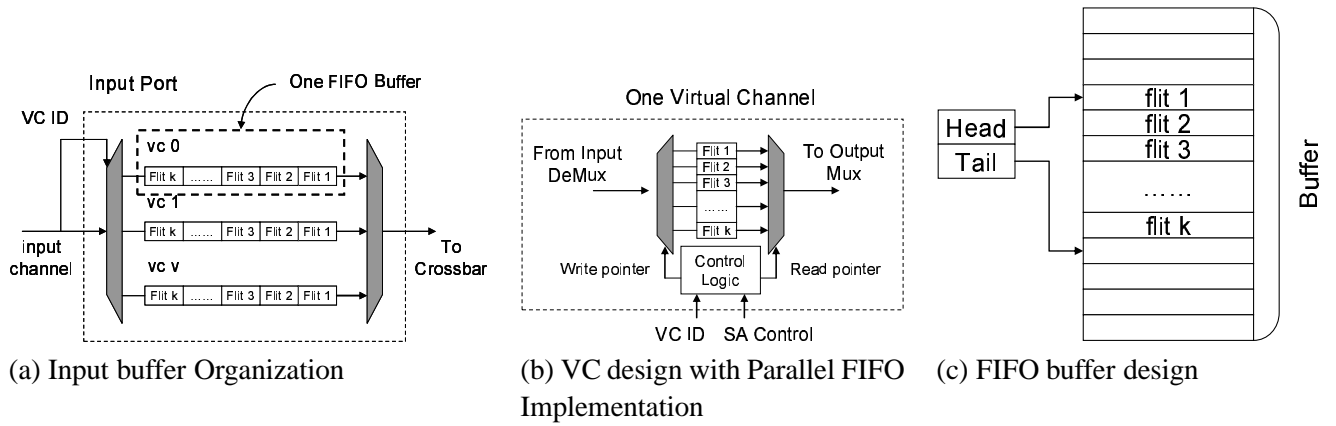


Figure 6: Generic NOC Router Input Buffer Architecture.

can be sent from a VC. Buffer designs for Monopolizing and Channel-stealing fall into the same category. In this section we mainly focus on the buffer design for Monopolizing and Channel-stealing. With a wide link and a small flit size, a VC can send multiple flits in a single cycle. Considering the high overhead of a multiple read or write ports buffer design [19], we do not add read or write ports to each VC. Every cycle the amount of data read from each VC is the same as the output channel bandwidth, which means the output width of SRAM/DRAM or registers² is the same as the width of output channels. This design is the same as a generic router which has the same flit size as the phit size. If the flit size is smaller than the phit size (for example, the flit size is one quarter of the phit size), then the data read from a VC in one cycle is not just one flit but multiple (four) flits. However, the flits read out from a VC are not guaranteed to be sent because according to the allocation of APCR, a VC may not use the whole bandwidth of an output channel. If multiple VCs share an output channel, each of them may not be able to send all the data (four flits) they have read. Only the guaranteed number of flits can be sent, and other read-out flits should be dropped. Meanwhile, only the flits belonging to the same packet can be sent out in the same cycle. To support the functionality of reading and dropping flits, input buffers design should be modified. The head and tail pointer are not enough. It is necessary to add a packet pointer which points the end position of a packet. Also the input buffer should be notified how many flits are effectively sent out. After each read operation, the head can be move to the correct position. This information is gathered from the APCR component after the SA stage. Figure 7 describes the whole procedure of the buffer management in an APCR router. VC_0 buffers multiple flits which belong to the same packet. Each link connected with VC_0 stands for a bunch of wires whose width is the same as the flit size. At step (1) multiple flits will be read out since the buffer output width is equal to the router output channel width, a multiple of the flit size. After the SA stage, the APCR component has the information of how many flits each VC should eventually send. At step (2) the allocation information will be used to set up the head pointer in each input buffer and control signals of each output MUX. At step (3) the guaranteed flits will be

²The buffers in an NOC router can be implemented as either SRAM/DRAM memory or registers [20, 21].

sent to downstream routers through crossbar and the remaining read-out flits are dropped. These dropped flits will be read out from the input buffers again in the next cycle. The buffer write recruits the same method as the buffer read. Obviously, these redundant operations will incur more buffer power consumption, which will be discussed in Section 6.

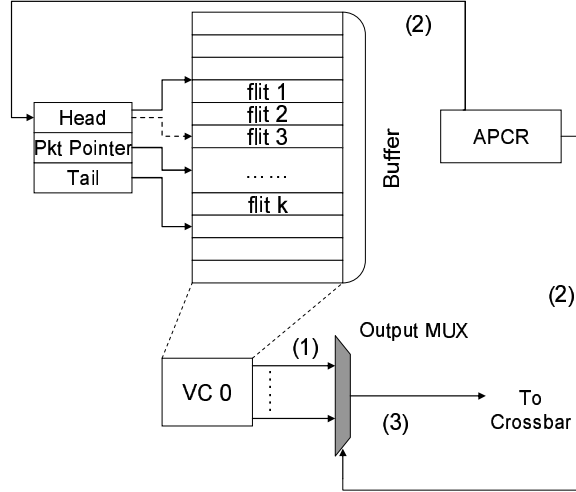


Figure 7: Three steps of buffer management: (1) VC_0 reads multiple flits out (the number of flits depends on the relationship between the flit size and phit size.). (2) APCR sets up the head of each input buffer and output MUX. (3) The guaranteed flits are sent through crossbar. The remaining flits will be dropped and read out again in the next cycle.

4.2.2 VC and Switch Allocation

The VC allocation in an APCR router is the same as in a generic router. However, the switch allocation is different, which brings the main overhead. Figure 8 (a) shows the two-stage arbitration used in a generic router³. The first stage is to select one VC from each input port. It needs $p v : 1$ arbiters. The second stage is for output ports, selecting one valid request from p input ports which have the same output direction. Hence, each output ports needs a $p : 1$ arbiter and the total number of the second stage arbiters adds up to p . Monopolizing has the same SA structure as a generic router, which does not incur any SA overhead. Switch allocation for fair-sharing is different as shown in Figure 8 (b). In fair-sharing, each VC of the same input port has its own reserved sub-channel, which guarantees one flit bandwidth. There is no competition among the VCs of the same input port. They share the wide channel between the input buffer and the crossbar, each occupying one flit bandwidth. This removes the first stage arbitration in a generic router. However, each output sub-channel needs a arbiter to decide the current winner because it has requests from VCs of different input ports. Considering VCs are bound with output sub-channels in fair-sharing⁴, the number of inputs of an output sub-channel arbiter is the same as the number of input ports, p . Assuming that the number of sub-channels of an output port is equivalent

³We assume the router has p input ports and each input port has v virtual channels.

⁴ VC_0 from any input port can only use sub-channel₀ of all the output ports.

to the number of VCs of an input port (v), $pv : 1$ arbiters should be provided. Channel-stealing needs even more complex SA structure because it provides the most flexible channel usage. Similar to fair-sharing, only a single stage arbitration is required, shown in Figure 8 (c). However, channel-stealing does not bind VCs with sub-channels, which means a VC can request any sub-channel in the same output direction. Hence, each output sub-channel needs a $pv : 1$ arbiter and the whole SA requires $pv : 1$ arbiters.

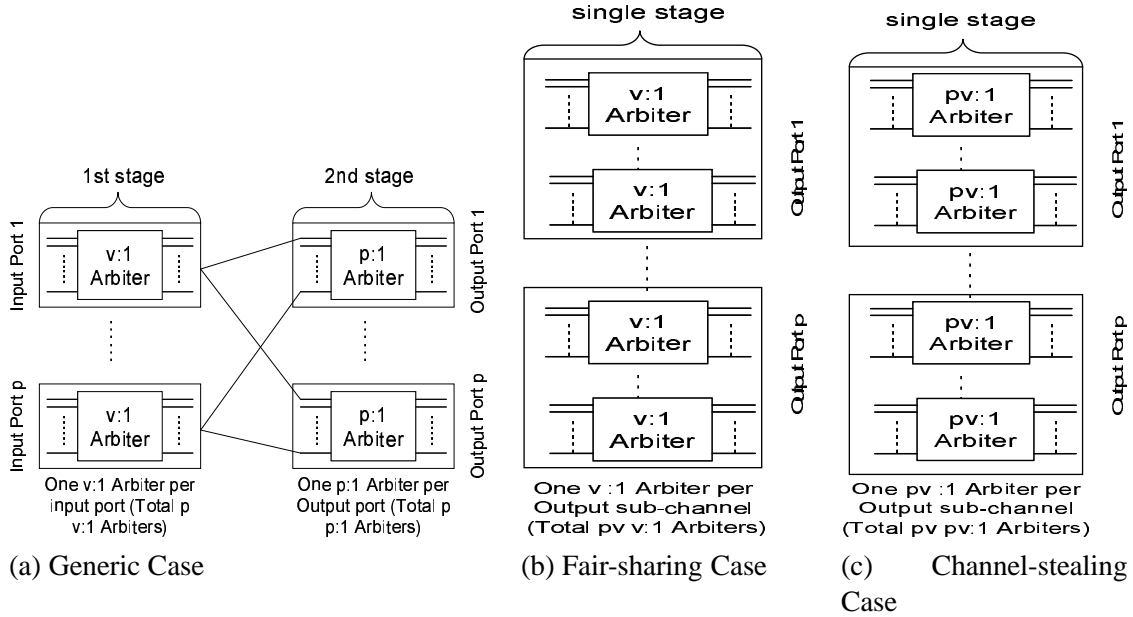


Figure 8: Switch Allocation

4.2.3 Credit-based Flow Control

With credit-based flow control, the upstream router keeps a record of the number of free buffers in each VC of downstream routers. Every cycle, when the upstream router forwards a flit, it will consume one credit until the credit counter becomes zero which means the VC of the corresponding downstream router is full. In the opposite way, when the downstream forwards a flit and frees the associated buffer, it sends a credit to the upstream, causing the credit counter to increase by one. In the generic NOC design, the flit size is the same as the phit size, which means every cycle the current router can forward at most one flit to its downstream router in each direction. This triggers a credit to be sent to the corresponding upstream router. However, in an APCR router the flit size is no more equivalent to the phit size (normally smaller), which causes the number of credits sent in each direction to be greater than one. To solve this problem, more wires are needed between two neighborhood routers to transmit the credit information. Considering the depth of a virtual channel is not big, the overhead of credit wires will be minor.

4.2.4 APCR’s Effect on the Router Pipeline

NOC router pipeline stage delays are quite imbalanced unlike the processor pipeline [22, 16]. Normally, VA stage is the bottleneck [23]. VA stage in an APCR router is the same as in a generic NOC router. However, SA stage is different. In the following timing analysis, we mainly focus on the SA stage. Table 1 shows the SA stage delays for three different regulation schemes. Results are obtained from HSPICE simulations using 45nm technology. Since monopolizing has the same SA as a generic router, they have the same SA delay. Fair-sharing has the lowest delay, because of two main reasons: firstly, fair-sharing only needs a single stage arbitration; secondly, binding VCs with output sub-channels makes the number of inputs of an output port arbiter small. Channel-stealing has the highest SA stage delay because of its complicated arbitration. However, compared to VA stage delay (328 ps), the SA delays of all the three schemes are still affordable. Additionally, Das et. al [23] have proposed a time stealing technology in the router pipeline design. Hence, the APCR component will not affect the pipeline depth or the router clock frequency.

Table 1: Time analysis of SA stage delay from HSPICE simulation using 45nm technology.

Generic	Monopolizing	Fair-sharing	Channel-stealing
247 ps	247 ps	122 ps	333 ps

5 Generalized NOC Router Design

In this section, we explore the design space of NOC routers, which we refer to as Generalized NOC Router Design (GNRD).

Building on the characteristics of on-chip network, we define the GNRD as a five-tuple $\langle d, v, l, f, p \rangle$ where:

d - the depth of a virtual channel (defined as the number of flits)

v - the number of virtual channels per input port

l - packet length (defined as the number of flits)

f - flit size

p - phit size (inter-router channel width)

Firstly, different router designs can be specified with the GNRD. The area of input buffers is linear to parameter d , v and f . With fixed flit size, manipulating d and v results in different router buffer designs, such as DAMQ [19] and ViCharR [24]. As an extreme case, when either d or v is zero, the router becomes the recently proposed bufferless router [25]. The relationship between f and p is the key point of this paper. In a generic router, f is always equivalent to p , which prevents fine-grained flow control. However in an APCR router design, f is smaller than p . With an Adaptive Physical Channel Regulator a packet can transmit multiple flits in a single cycle, and the number of flits being transmitted every cycle dynamically changes according to the

network status, providing more flexible channel utilization and flow control.

Secondly, the router performance is related to the GNRD. In a wormhole-switching network, the sharing and multiplexing of physical links among different source-destination pairs result in increased packet transmission latency T , which is defined as the time elapsed between the head flit of the packet being injected at the source and the tail flit being ejected at the destination:

$$T = D/V + l/p + h \times t_{router} + t_c, \quad (1)$$

where D is the Manhattan distance between the source and the destination. V is the propagation velocity. l is the packet size and p is the channel width. h is the hop count while t_{router} is the router latency. t_c is the latency when a contention occurs. From Equation 1, we can see that the two parameters in the GNRD, l and p , directly affect the packet latency. The smaller l/p , the better is the performance achieved. It also give us this implication: If we define f to be the same as p , as p becomes bigger, f also gets bigger which results in a smaller l (We assume the total number of bits for a packet is fixed. l is total number of bits divided by f). A smaller l and bigger p will provide even smaller l/p , producing smaller T , which results in better performance. This conclusion is true when the workload of the network is low. In a high workload network, there are many packets injected from the network interface. A bigger p makes either d or v smaller if the total input buffer size is fixed. A smaller d or v will cause the contention delay (t_c) to increase. At this point, whether the packet latency (T) will increase or decrease is open to doubt. An APCR router provides a good trade-off between the two components related with the packet latency. On one hand, when the workload is low, an APCR router allows a packet to use the entire channel resources, equivalent to defining f equal to p . On the other hand, when the workload is high, an APCR router makes more packets share the output channels, which potentially releases the network congestion and reduces the contention delay t_c . While further analysis of the GNRD is beyond the scope of this paper, the experiment results in Section 6 include some sensitivity studies on several parameters in the GNRD.

6 Experimental Evaluation

We evaluate our APCR router design using both synthetic workloads and real applications, comparing it with a baseline router design, in which the flit size is always the same as the output channel width. We also examine the regulation schemes' sensitivity to a variety of network parameters.

6.1 Methodology

Our evaluation methodology contains two parts. Firstly, we use Simics [26], a full-system simulator, configured for UltraSPARCIII+ multiprocessors running Solaris 9 and GEMS [27] that models directory-based cache coherence protocols to obtain real workload traces. Table 2 shows the main parameters of our CMP system.

All the cache related delay and area parameters are determined by CACTI [28]. Secondly, we evaluate the performance with a cycle-accurate network simulator that models all router pipeline delays and wire latencies. Table 3 summarizes the configuration for our network simulator. We use Orion 2.0 [29] for power and area estimation. Orion 2.0 simulator uses a recent model [28] and estimates the area of transistors and gates using the analysis in [30]. For area modeling, Orion 2.0 provides value estimates for inverters and 2-input AND and NOR gates and adds an additional 10% to the total area to account for global white space. For power modeling, Orion 2.0 estimates dynamic and static power consumption for buffer, crossbar, arbiter, and link with 50% switching activity and 1V supply voltage in 45nm technology. We model a link as 512 parallel wires, which takes advantage of abundant metal resources provided by future multi-layer interconnect.

The workloads for our evaluation consist of synthetic workloads and real applications. Three different synthetic traffic patterns, namely Uniform Random (UR), Bit Complement (BC) and Transpose (TP), are used in our evaluation. Our synthetic workloads support different packet sizes. A one-flit packet (short packet) emulates a control message, and a five-flit packet (long packet) emulates a data message. The percentage of short packets is 60%⁵. The packet generation rate for each node is constant. The real application workloads considered in this paper are three programs (fft, lu, radix) from SPLASH-2 [32], four benchmarks (blacksholes, streamcluster, swaptions and freqmine) from the PARSEC suite [33], three programs (equake, fma3d and mgrid) from SPEComp2001 [34] and SPECjbb2000 [35]. We configure our network simulator to match the environment in which the traces are captured.

Table 2: CMP System Parameters.

Clock Frequency	4 GHz
Core Count	32
L1 I & D cache	1-way & 4-way, 32KB, 1 cycle
L2 cache	16-way, 16MB, 512KB per bank, 32 banks, 20 cycles
L1/L2 cache block	64B
Memory Latency	300 cycles
Coherence Protocol	Directory-based MSI

6.2 Performance

In this section, we first evaluate the average packet latency for the three schemes, compared with the baseline router design. The flit size in the three regulation schemes is only one quarter of the channel width, which means four flits can be transmitted in each cycle. Since the flit sizes in our design and the baseline design are different, we use *packet per node per cycle* as the metric of workloads to ensure a fair comparison. Each packet has the same number of bits. A packet which contains four flits in our design will only have one flit in the

⁵The percentage is taken from SIMICS with GEMS extension [31].

Table 3: Baseline Network Configuration and Variations.

Characteristic	Baseline	Variations
Topology	8×8 2D Mesh	–
Routing	XY	–
Router Arch	Two-stage Speculative	APCR Router
Per-hop Latency	3 cycles: 2 cycle in router, 1 cycle to cross link	–
Virtual Channels/Port	4	–
Packet Length(flits)	one flit for control and five flits for data	–
Traffic Pattern	Uniform Random, Bit Complement, Transpose	SPEComp, SPECjbb, SPLASH-2 and PAR-SEC
Simulation Warm-up Cycles	10,000	–
Total Simulation Cycles	200,000	10,000,000 for real applications

baseline design. We also fix the total buffer size of each router. Since the flit sizes are different, the depth of input VCs in our design is different from that of the baseline design.

Standard Synthetic Workloads: Figure 9 shows the simulation results using three synthetic workloads. The results are consistent with our expectations. The trends observed in all the three traffic patterns are the same. When the packet injection rate is low, the performance of the four schemes only has minor differences. However, with high injection rates, an APCR router performs better than a baseline router. For example, when the injection rate is 0.1, monopolizing improves the performance by 67% for the Uniform Random traffic. Among the three regulation schemes, channel-stealing is the best. The main reason is that channel-stealing utilizes the output channels most efficiently. If there are flits ready in any VC and downstream routers have enough credits, the output channel can always be utilized. No channel usage restrictions exist in channel-stealing. In the baseline design, if a flit is transmitted through the channel, since the flit size is the same as the channel width, the wide channel is also fully used. However, a big flit size can affect the VC depth if we keep the router buffer size constant. When the packet injection rate is high, with shallow VC depth, contentions occur and the network easily can get saturated. An APCR router makes more packets share the same channel resources, which potentially releases the network congestion. The three schemes perform much better than the baseline when the packet injection rate is above 0.1. Also when the network congestion is released, the network throughput is improved. It is observed that channel-stealing increases the throughput by more than 100% in all the three traffic patterns. Figure 9 (d) shows the performance result with the buffer size of the APCR router halved. We can see that the APCR router can still outperforms the baseline router.

Real Applications: Figure 10 shows the average packet latency across real applications. Channel-stealing delivers the best performance while the baseline is the worst. Since the packet injection rate of each node in these real applications is very low (below 0.01), the latency improvement of the three schemes over the baseline

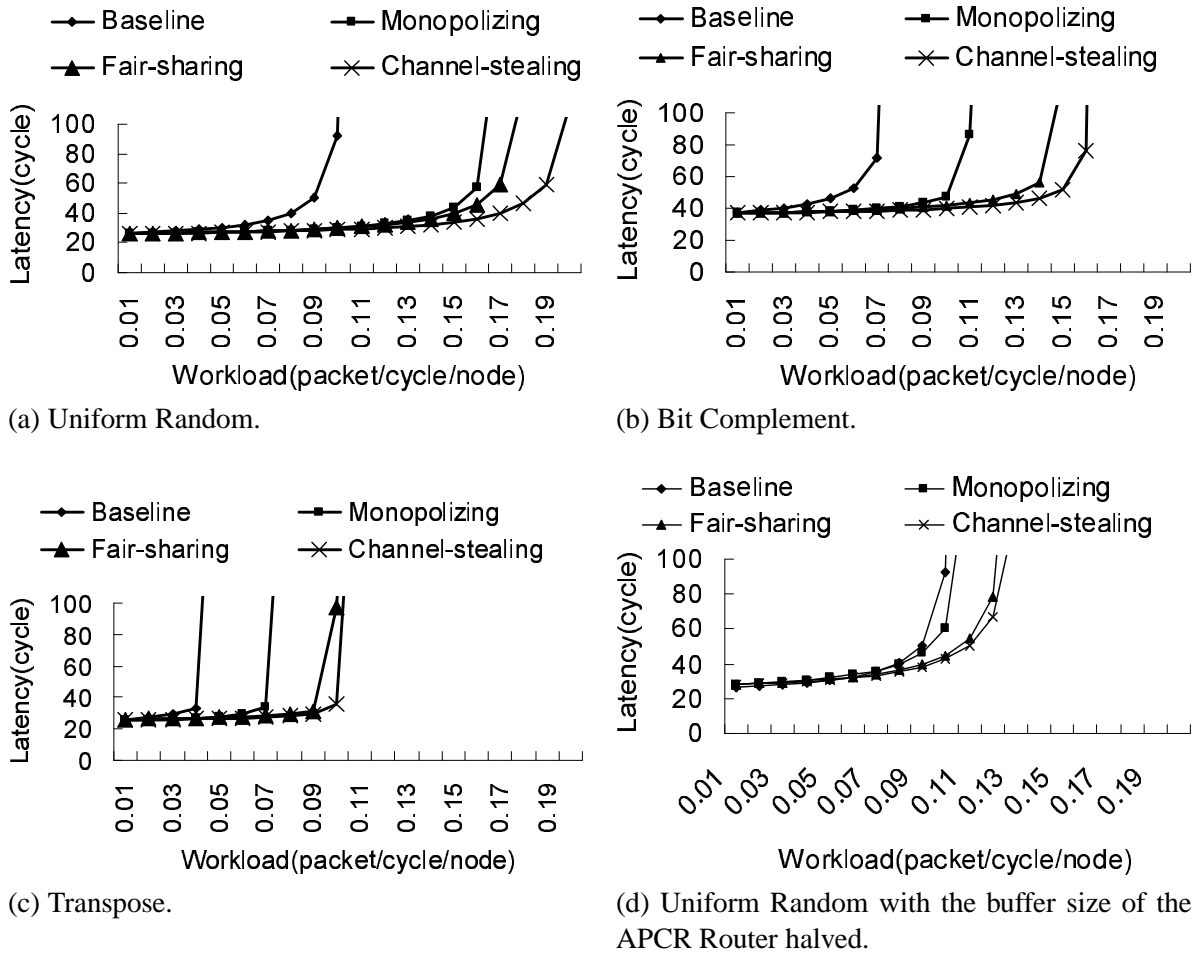


Figure 9: Packet Latency with Three Synthetic Traffic Patterns in an (8×8) Mesh Network.

is not obvious. Specjbb gets 9.3% improvement while fft gets 4.3% and lu gets 6.6%.

6.3 Power Analysis

Power consumption is one of the main concerns in NOC router design. Compared with the baseline router design, an APCR router incorporates certain modifications into the operations of input buffers and SA arbiters. In this section, we use Orion 2.0 to estimate the power consumption of an APCR router. Figure 11 (a) shows the total power consumption with different packet injection rates in the Uniform Random (UR) traffic. We observe that all the three schemes (Monopolizing, Fair-sharing and Channel-stealing) consume less power than a baseline router before the network saturation point (0.1). Monopolizing saves 33%, fair-sharing saves 43% and channel-stealing saves 35%. Among the three schemes Fair-sharing has the lowest power consumption.

Figure 11 (b) presents the detailed power values for each component. Compared with the baseline router, monopolizing and channel-stealing exhibit higher power consumption in input buffers. The reason behind this can be understood from the buffer operations of the two schemes. To avoid using multiple read and write ports, the buffer in monopolizing and channel-stealing still have the same port width as the output channel even

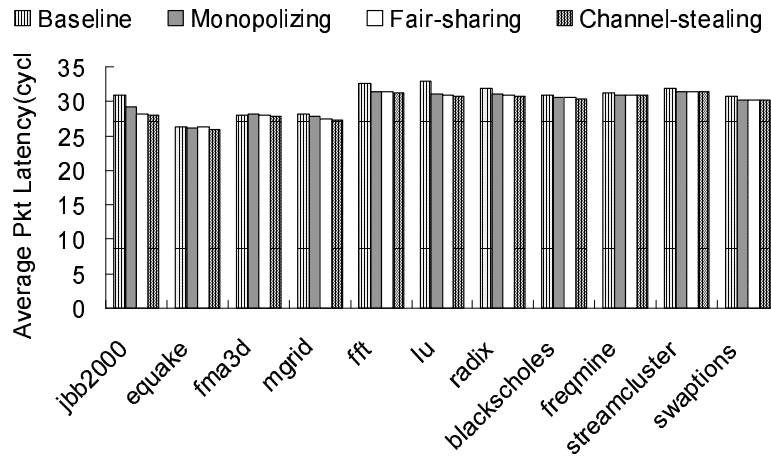
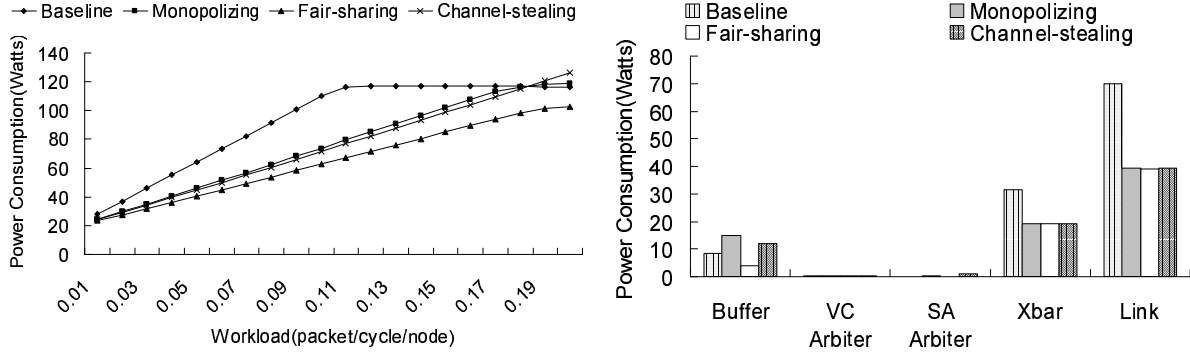


Figure 10: Average Latency Across Benchmarks.

though the flit size is smaller than the phit size. This design introduces multiple redundant read operations for the same flit. For example, let's assume the output channel width is 512 bits and the flit size is 128 bits. Then the width of the input buffer read port is also 512 bits. At the ST stage, even though the current VC can only send one flit, four flits are read out, three of which will be dropped since they cannot traverse the crossbar in the current cycle. In the worst case, if four flits are stored in a VC but the arbitration happens such that only one flit can be actually sent every cycle, then the first flit will be read out once, the second flit twice, the third flit three times and the fourth flit four times. However, in a baseline router flits are read from input buffers only when they can be sent. These redundant buffer operations in monopolizing and channel-stealing cause the high buffer power consumption. The buffer design in fair-sharing does not have this problem because each VC sends at most one flit every cycle, which is the same as a baseline router. However, fair-sharing has less power consumption than a baseline router. The main reason is that a small flit size achieves finer granularity packet definition, which is more power efficient. For example, in a baseline router, if the channel is 512 bits, then the flit size should be 512 bits. The control message in a cache coherence protocol is usually no more than 128 bits. Since a flit is the basic unit, this 128-bit control message should use at least one flit. In a 512-bit flit, only the first 128 bits are valuable data and the other 384 bits are useless, thereby wasting the network power. An APCR router can define a relatively small flit size, such as 128 and 256 bits. With a 128-bit flit, all the information in a flit is useful and with a 256-bit flit, only 128 bits are wasted. Compared with wasting 384 bits, an APCR router is more power efficient. This benefit can also be observed from the crossbar and link power in Figure 11 (b). If we only use one portion of the wide channel, the underused portions will be idle thus consuming only static power. Simulation results show that the three schemes save crossbar power by 40% and link power by 44%. As an APCR router recruits a more complicated SA arbiter design, it leads to higher power consumption in SA arbiters. From Figure 11 (b) we can see that compared with other components the power consumption due to SA arbiters is negligible. This also explains why the total power consumption of an APCR router is still lower

than a baseline router.



(a) Total Power Consumption with Uniform Random Traffic. (b) Power Consumption of Each Component.

Figure 11: Power Analysis of APCR Router.

6.4 Area Analysis

Table 4 shows the area of each router component. The difference only comes from the SA arbiters. The arbiters used in fair-sharing consume two times the area of the arbiters in a baseline router while for channel-stealing it is four times. However, compared with other components, the area used by arbiters is negligible. That’s why we cannot see much area difference when we consider the total router area (The last row of Table 4).

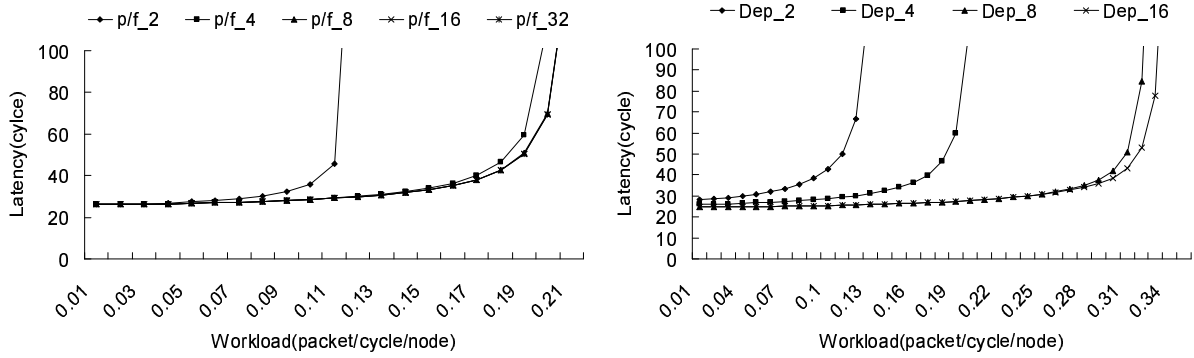
Table 4: Router Area.

	Baseline (μm^2)	Monopolizing (μm^2)	Fair-sharing (μm^2)	Channel-stealing (μm^2)
Buffer	94195.7	94195.7	94195.7	94195.7
XBar	1.36273E+07	1.36273E+07	1.36273E+07	1.36273E+07
VA Arbiter	36659.9	36659.9	36659.9	36659.9
SA Arbiter	954.437	954.437	1908.87	36659.9
Total	1.37591E+07	1.37591E+07	1.37601E+07	1.37948E+07

6.5 Sensitivity to Network Design Points

According to the GNRD, different parameters come out different router designs. Here, we present variations that provide insight into the performance of an APCR router in different environments. Observing that channel-stealing delivers the best performance, we use channel-stealing for our experiments in our sensitivity study. The key idea of an APCR router is that the flit size is smaller than the phit size. The effect of the ratio between the phit size and flit size on the performance is shown in Figure 12 (a). “p/f₂” means the phit size is two times the flit size. In this situation, the output channel is divided into two sub-channels. We can see that from “p/f₂” to “p/f₄” the network throughput is almost doubled. However, as the value of “p/f” keeps increasing, there is no significant improvement. The main reason is that the depth of VCs is fixed as four. Even though the router has

many free sub-channels, like 16 or 32, the maximum flits stored in a VC is four. Hence, VCs cannot provide enough flits to completely utilize the sub-channel resources.



(a) Impact of the Number of Sub-channels on Packet Latency. (b) Impact of the Depths of VC on Packet Latency.

Figure 12: Sensitivity Studies with Different Network Design Points.

To further study how the VC depth affects the performance of an APCR router, we simulate with different VC depths. Each output channel is divided into four sub-channels, which also means the flit size is one quarter of the phit size. The number of VCs is also fixed as four. Figure 12 (b) summarizes our experiments. Firstly, we observe that increasing the depth of VCs improves the network throughput. For example, changing the depth from 2 to 4 causes the network throughput to be improved by more than 53%. From 4 to 8, we get around 60% improvement. However, if we further increase the depth of VCs from 8 to 16, the improvement is insignificant. One reason for this can be deduced from the packet size. In our experiment, the maximum packet size is five. VCs, whose depth is 8 or 16, can always hold one whole packet. The number of credits provided by the downstream router can also be greater than five. However, as the rule of regulation schemes, flits belonging to different packets cannot be transmitted in the same cycle. The maximum number of flits a VC can send is no more than five. Another reason is from the number of sub-channels. Since there are only four sub-channels, the output channel can only process four flits in a cycle. Thus, even with a big VC depth, the network throughput cannot be improve much.

7 Conclusions

Abundant wires inside a chip infer that it is important to organize this huge wiring capabilities. In this paper, we propose an Adaptive Physical Channel Regulator (APCR) for NOC routers to efficiently allocate the huge wiring resources. By defining a small flit, an APCR router allows flits from different packets or flows to share the same output channel in a single cycle. The three regulation schemes (Monopolizing, Fair-sharing and Channel-stealing) intelligently allocate the output channel resources considering not only the availability of physical channels but the occupancy of input buffers. An APCR router allows a virtual channel to transmit

multiple flits every cycle. The number of flits being transmitted every cycle dynamically changes according to the network status. We also introduce Generalized NOC Router Design (GNRD) – a framework for exploring the design space of NOC routers. Our simulation results using a detailed cycle-accurate simulator show that an APCR router improves the network throughput by over 100%, compared with a baseline router design with the same buffer size. An APCR router can still outperform a baseline router when the buffer size is halved. Furthermore, an APCR router saves the router power consumption by over 33% with a little area overhead.

References

- [1] J. D. Owens, W. J. Dally, R. Ho, D. N. Jayasimha, S. W. Keckler, and L.-S. Peh, “Research Challenges for On-Chip Interconnection Networks,” *IEEE Micro*, vol. 27, no. 5, pp. 96–108, 2007.
- [2] R. Ho, K. Mai, and M. Horowitz, “The Future of Wires,” in *Proceedings of the IEEE*, pp. 490–504, 2001.
- [3] Y. Hoskote, S. Vangal, A. Singh, N. Borkar, and S. Borkar, “A 5-GHz Mesh Interconnect for a Teraflops Processor,” *IEEE Micro*, vol. 27, no. 5, pp. 51–61, 2007.
- [4] M. B. Taylor, W. Lee, S. P. Amarasinghe, and A. Agarwal, “Scalar Operand Networks: On-Chip Interconnect for ILP in Partitioned Architecture,” in *Proceedings of HPCA*, pp. 341–353, 2003.
- [5] J. Kim, J. Balfour, and W. J. Dally, “Flattened Butterfly Topology for On-Chip Networks,” in *Proceedings of MICRO*, pp. 172–182, 2007.
- [6] B. Grot, J. Hestness, S. W. Keckler, and O. Mutlu, “Express Cube Topologies for On-Chip Interconnects,” in *HPCA*, pp. 163–174, 2009.
- [7] W. J. Dally and B. Towles, “Route Packets, Not Wires: On-Chip Interconnection Networks,” in *Proceedings of DAC*, pp. 684–689, 2001.
- [8] R. D. Mullins, A. West, and S. W. Moore, “The Design and Implementation of a Low-Latency On-Chip Network,” in *ASP-DAC*, pp. 164–169, 2006.
- [9] J. D. Balfour and W. J. Dally, “Design Tradeoffs for Tiled CMP On-Chip Networks,” in *ICS*, pp. 187–198, 2006.
- [10] R. Das, S. Eachempati, A. K. Mishra, N. Vijaykrishnan, and C. R. Das, “Design and Evaluation of a Hierarchical On-Chip Interconnect for Next-Generation CMPs,” in *HPCA*, pp. 175–186, 2009.
- [11] A. Leroy, P. Marchal, A. Shickova, F. Catthoor, F. Robert, and D. Verkest, “Spatial Division Multiplexing: A Novel Approach for Guaranteed Throughput on NoCs,” in *CODES+ISSS*, pp. 81–86, 2005.
- [12] C. G. Requena, M. E. Gómez, P. López, and J. Duato, “Exploiting Wiring Resources on Interconnection Network: Increasing Path Diversity,” in *PDP*, pp. 20–29, 2008.
- [13] C. G. Requena, M. E. Gómez, P. J. L. Rodríguez, and J. Duato, “An Efficient Switching Technique for NoCs with Reduced Buffer Requirements,” in *ICPADS*, pp. 713–720, 2008.

- [14] W. J. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. Morgan Kaufmann, 2003.
- [15] M. Galles, “Scalable Pipelined Interconnect for Distributed Endpoint Routing: The SGI SPIDER chip,” in *Proceedings of Hot Interconnect 4*, pp. 141–146, 2009.
- [16] L.-S. Peh and W. J. Dally, “A Delay Model and Speculative Architecture for Pipelined Routers,” in *Proceedings of HPCA*, pp. 255–266, 2001.
- [17] R. D. Mullins, A. West, and S. W. Moore, “Low-Latency Virtual-Channel Routers for On-Chip Networks,” in *Proceedings of ISCA*, pp. 188–197, 2004.
- [18] A. Yakovlev, A. Koelmans, and L. Lavagno, “High-Level Modeling and Design of Asynchronous Interface Logic,” *IEEE Design & Test of Computers*, vol. 12, no. 1, pp. 32–40, 1995.
- [19] Y. Tamir and G. L. Frazier, “High-Performance Multi-Queue Buffers for VLSI Communication Switches,” in *Proceedings of ISCA*, pp. 343–354, 1988.
- [20] J. Hu and R. Marculescu, “Energy- and Performance-aware Mapping for Regular NoC Architectures,” *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 24, no. 4, pp. 551–562, 2005.
- [21] H. Wang, L.-S. Peh, and S. Malik, “A Technology-Aware and Energy-Oriented Topology Exploration for On-Chip Networks,” in *DATE*, pp. 1238–1243, 2005.
- [22] R. Das, A. K. Mishra, C. Nicopoulos, D. Park, V. Narayanan, R. Iyer, M. S. Yousif, and C. R. Das, “Performance and Power Optimization Through Data Compression in Network-on-Chip Architectures,” in *HPCA*, pp. 215–225, 2008.
- [23] A. K. Mishra, R. Das, S. Eachempati, R. Iyer, N. Vijaykrishnan, and C. R. Das, “A Case for Dynamic Frequency Tuning in On-Chip Networks,” in *MICRO*, pp. 292–303, 2009.
- [24] C. 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 MICRO*, pp. 333–346, 2006.
- [25] T. Moscibroda and O. Mutlu, “A Case for Bufferless Routing in On-Chip Networks,” in *ISCA*, pp. 196–207, 2009.
- [26] P. S. Magnusson, M. Christensson, J. Eskilson, D. Forsgren, G. Hållberg, J. Högberg, F. Larsson, A. Moestedt, and B. Werner, “Simics: A Full System Simulation Platform,” *IEEE Computer*, vol. 35, no. 2, pp. 50–58, 2002.
- [27] M. M. K. Martin, D. J. Sorin, B. M. Beckmann, M. R. Marty, M. Xu, A. R. Alameldeen, K. E. Moore, M. D. Hill, and D. A. Wood, “Multifacet’s General Execution-driven Multiprocessor Simulator (GEMS) Toolset,” *SIGARCH Computer Architecture News*, vol. 33, no. 4, pp. 92–99, 2005.
- [28] S. Thoziyoor, N. Muralimanohar, J. H. Ahn, and N. P. Jouppi, “CACTI 5.1,” tech. rep., HP Laboratories, 2008.

- [29] A. B. Kahng, B. Li, L.-S. Peh, and K. Samadi, "ORION 2.0: A Fast and Accurate NoC Power and Area Model for Early-Stage Design Space Exploration," in *DATE*, pp. 423–428, 2009.
- [30] H. Yoshida, K. De, and V. Boppana, "Accurate Pre-layout Estimation of Standard Cell Characteristics," in *DAC*, pp. 208–211, 2004.
- [31] B. Cuesta, A. Robles, and J. Duato, "An Effective Starvation Avoidance Mechanism to Enhance the Token Coherence Protocol," in *PDP*, pp. 47–54, 2007.
- [32] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta, "The SPLASH-2 Programs: Characterization and Methodological Considerations," in *ISCA*, pp. 24–36, 1995.
- [33] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The PARSEC Benchmark Suite: Characterization and Architectural Implications," in *PACT*, pp. 72–81, 2008.
- [34] "Speccomp 2001 benchmark suite." <http://www.spec.org/omp/>.
- [35] "Specjbb 2000 benchmark." <http://www.spec.org/jbb2000/>.