An SDRAM-Aware Router for Networks-on-Chip

Wooyoung Jang and David Z. Pan
Department of Electrical and Computer Engineering
University of Texas at Austin
wyjang@cerc.utexas.edu, dpan@ece.utexas.edu

ABSTRACT
In this paper, we present an NoC (Networks-on-Chip) router with an SDRAM-aware flow control. Based on a priority-based arbitration, it schedules packets to improve memory utilization and reduce memory latency. Moreover, our multi-scheduling scheme performed by the multiple SDRAM-aware routers helps to achieve better SDRAM performance and save the hardware cost of NoC platform. Experimental results show that our SDRAM-aware router improves memory latency by 18% and memory utilization by 4.9% on average with over 42% saving of gate count of the NoC platform with dual memory subsystem.

Categories and Subject Descriptors
C.2.1 [**COMPUTER-COMMUNICATION NETWORKS**]: Network Architecture and Design - Packet-switching networks.

General Terms
Algorithms, Performance, Design.

Keywords
Networks-on-Chip, router, flow control, memory

1. INTRODUCTION
An NoC (Networks-on-Chip) has been proposed as a scalable solution to complex on-chip interconnection problems [1, 2]. Recently, one inter-layer interconnection comes to extend the NoC into three dimensions by TSV (Through-Si-Vias) technology [3]. Especially, a 3D NoC embedded with multiple SDRAMs (Synchronous Dynamic Random Access Memory) for L2 or L3 caches on top of processing elements in different layers achieves higher memory bandwidth, shorter latency and more reliable electrical features than a conventional 2D NoC only interfacing with one or two off-chip SDRAMs [4].

A memory subsystem managing an SDRAM is one of the most important components in most 2D/3D NoC designs since the performance of the whole system is considerably sensitive to the performance of the memory subsystem. However, a memory subsystem frequently underperforms due to characteristic operation flows of an SDRAM [5] and dynamic accesses by various processing components. For example, DDR (Double Data Rate) II SDRAM utilization (defined as the number of clock cycles transferred valid data divided by the total number of clock cycles required to access data) becomes much worse by 55% in DTV (Digital Television) application [6]. Moreover, the corresponding number of memory subsystem is also equipped to manage the SDRAMs. Since the gate count of a single memory subsystem occupies over 35% of the entire 3×3 NoC platform consisting of a single memory subsystem, nine routers and twelve links from our experimental results, the NoC design cost highly increases depending on the number of memory subsystem. Therefore, considerable attention has shifted toward memory-aware NoC exploration to improve memory utilization and latency and save the design cost of the NoC platform.

In this paper, we propose an SDRAM-aware NoC router improving total SDRAM utilization and latency and decoupling the NoC design cost from the number of SDRAM. Our key ideas are two-fold. First, if an NoC router schedules packets to access SDRAM efficiently, the packets arrive at a memory subsystem into the order that is more friendly to SDRAM operations. At a result, the complexity of a memory subsystem considerably decreases while the memory performance is more improved. Since our SDRAM-aware router uses existing resources to schedule packets, e.g. input buffers for storing blocked packets or other flow-control mechanisms, the additional design cost is tiny. Instead, heavy reordering buffers and a scheduler are removed in a memory subsystem. Second, a multi-scheduling scheme performed by multiple SDRAM-aware routers outperforms a single-scheduling scheme performed by a single memory subsystem. The reason is that the performance of a single scheduling scheme is mainly limited by the depth of reordering buffers in a memory subsystem. However, our multi-scheduling scheme uses all input buffers in multiple routers to reorder packets. Based on these ideas, the major novelty and contribution of this paper include the following.

- We propose an NoC router with an SDRAM-aware flow control. It schedules a packet to access an SDRAM instead of a memory subsystem.
- We employ both a priority-based arbitration based on SDRAM-awareness and a multi-scheduling scheme that are performed by multiple NoC routers.
- We show that the NoC design with our SDRAM-aware router can achieve higher memory utilization, shorter memory latency and cheaper hardware cost than the conventional NoC design with an SDRAM-unaware router.

The rest of this paper is organized as follows. In the next section, we survey related works. In section 3, SDRAM operation principle and SDRAM scheduling are reviewed. In section 4, a problem in NoC design with a memory subsystem and our basic solution are present. Section 5 presents a detailed description of the SDRAM-aware router. Experimental results are shown in Section 6. Finally, section 7 concludes the paper.

DAC'09, July 26-31, 2009, San Francisco, California, USA

2. RELATED WORKS

A 3D NoC embedded with multiple SDRAMs provides more bandwidth, shorter latency and less number of timing-critical paths. However, the corresponding number of an expensive memory subsystem is also equipped to manage the SDRAMs. A memory subsystem usually consists of three parts, i.e., a buffer, a SDRAM scheduler and a SDRAM interface signal generator, where the depth of buffer and the scheduler for reordering memory accesses are a key factor of high memory utilization and short memory latency. Memory access scheduler proposed in [7] supports preemption and reordering to optimize bandwidth and average latency. The schedulers discussed in [8, 9] support preemption for high-priority requestors to decouple latency and data rate. PREDATOR proposes grouping memory accesses and a predictable arbiter for this group in [10]. Memory scheduler proposed in [11] adopts the adaptive history-based algorithm.

Flow control in an NoC router focuses on how network resources, e.g. channel bandwidth, buffer capacity and control state, are allocated to packets traversing a network. A decentralized control system and a predictive explicit-rate control are developed in [12, 13]. A predictive flow controller that controls the packet injection rate is proposed in [14]. To improve the overall execution time and link utilization, optimal link scheduler and shared buffer router architecture are proposed in [15]. An open-loop flow control scheme is proposed in [16] to reduce the conflicts of data transfers from multiple memory modules to the same masters. Our SDRAM-aware flow-controller schedules packets to improve the utilization and latency of SDRAM based on priority-based arbitration and a multi-scheduling scheme, which can work together with other flow controls and routing schemes.

3. PRELIMINARY

3.1 Basic SDRAM Operation

An SDRAM has a three dimensional structure, i.e., a bank, a row and a column as shown in Fig. 1. Basic commands to access the SDRAM are ACT (Activate), R/W (Read/Write) and PRE (Precharge), where the ACT is executed with a BA (Bank Address) and an RA (Row Address), the R/W is executed with a BA and a CA (Column Address) and the PRE is executed with a BA. In Fig. 1, when any idle bank is activated by ACT, one row data of the bank move to a row buffer of the bank. It takes t_{RCD} to complete the ACT. Timing parameters of a DDR II SDRAM used in this paper is shown in Table 1. Then, R/W is executed on the filled row buffer. After read latency called *CL* or write latency called t_{DQSS}, successive data go from or to the SDRAM. Finally, PRE is executed to deactivate the active row buffer in the bank. It also takes t_{RP} to become the idle state again.

3.2 Memory Scheduling

An SDRAM consists of independent multiple banks while address and data resources serialize access to different banks, as shown in Fig. 1. Its benefit is that pin/wire resources are saved and commands to a different bank are pipelined, i.e. while data is transferred to or from one bank, the rest of banks become idle and active for a later request. Nevertheless, the improvement of SDRAM utilization is still limited due to wasted cycles caused by the above characteristic operation flows of an SDRAM and dynamic accesses by various processing components. Main factors

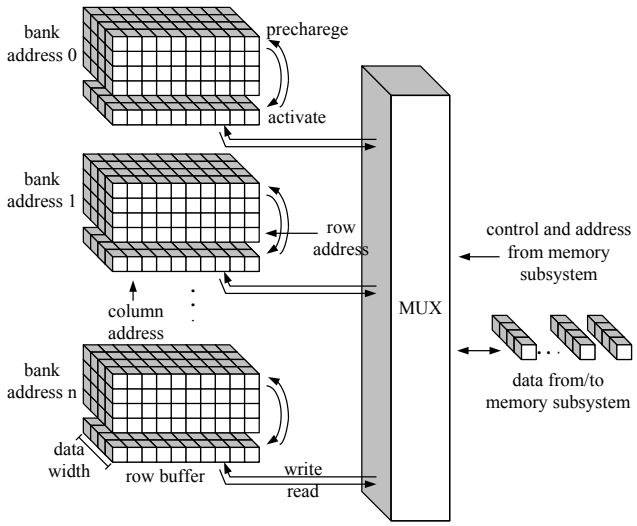


Figure 1: SDRAM architecture

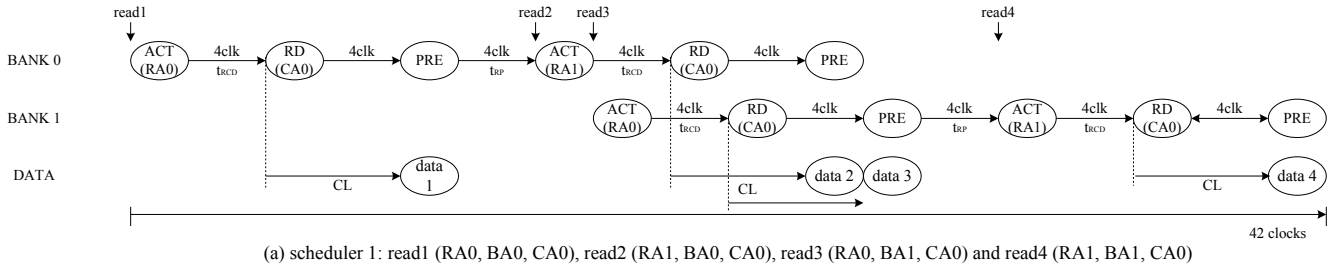
Table 1: Timing parameter of DDR II SDRAM @333MHz [5]

CL	CAS (Column Access Strobe) latency (4 clocks)
t_{RCD}	RAS (Row Access Strobe) to CAS delay (4 clocks)
t_{RP}	Row precharge time (4 clocks)
t_{WR}	Write recovery time (5 clocks)
t_{WTR}	Internal write to read command delay (3 clocks)
t_{DQSS}	DQS (Data Strobe) latching rising transitions to associated clock edges (1 clock)

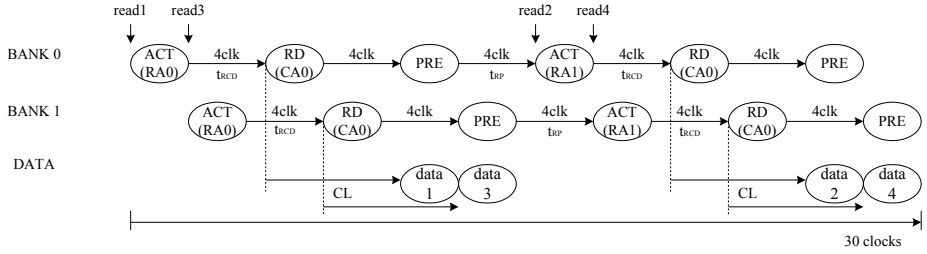
that deteriorate the SDRAM utilization are the following or their combinations:

- Continuous accesses of the same bank with different RAs.
- Write (read) access followed by any read (write) access.

The first factor called *bank conflict* is the most critical to SDRAM utilization since a bank activated by the former access should get idle and then active for the later access. In Fig. 2, there are two SDRAM schedulers that reorder four read accesses, i.e. read 1 (RA 0, BA 0, CA 0), read 2 (RA 1, BA 0, CA 0), read 3 (RA 0, BA 1, CA 0) and read 4 (RA 1, BA 1, CA 0). As shown in Fig. 2(a), let them scheduled in the order, read 1, read 2, read 3 and read 4 by scheduler 1. After completing read 1 as mentioned in section 3.1, read 2 cannot be immediately executed since the row buffer of bank 0 is already occupied by the data of RA 0. Therefore, PRE should be executed to release the filled row buffer of bank 0 and then ACT should be executed to fill the row buffer of bank 0 with the data of RA 1. On the other hand, read 3 can be pipelined, that is called *bank interleaving* since it has the different BA from the BA of read 2. At a result, the data 3 accessed by read 3 are generated with no loss of clock cycles as shown in Fig. 2(a). The last read 4 conflicts with read 3 since they have the same BAs, but the different RAs. On the other hand, scheduler 2 changes the order of four reads: read 1, read 3, read 2, and read 4 as shown in Fig. 2(b). This order does not cause any bank conflict such that all of the read accesses are pipelined. Consequently, the second SDRAM scheduler lets all SDRAM accesses completed faster than the first SDRAM scheduler. In this example, memory utilization of the first scheduler is 9.5% (= 4 data / 42 clock cycles) and that of the second scheduler is 13.3% (= 4 data / 30 clock cycles).



(a) scheduler 1: read1 (RA0, BA0, CA0), read2 (RA1, BA0, CA0), read3 (RA0, BA1, CA0) and read4 (RA1, BA1, CA0)



(b) scheduler 2: read1 (RA0, BA0, CA0), read3 (RA0, BA1, CA0), read2 (RA1, BA0, CA0) and read4 (RA1, BA1, CA0)

Figure 2: Examples of SDRAM operation depending on schedulers

The second factor is called *data contention*. Data pins or wires in an SDRAM are bidirectional while control and address pins or wires are unidirectional. Thus, input data can be collided with output data. To prevent it, at least one clock cycle should pass when a write access after a read access is accessed. A write access followed by a read access also waits for write recovery time (t_{WTR}) after the write access, which degrades the memory performance. Therefore, continuous read or write accesses are encouraged.

4. NOC DESIGN WITH SDRAM

4.1 Problem Description

The bank conflict and data contention frequently happen in the conventional NoC design due to the limited resources in the memory subsystem. Fig. 3 shows a simple example of bank conflict in the 2x3 NoC design. It includes one memory subsystem that schedules packets to avoid the bank conflict and data contention. In Fig. 3, *RxBy* means that an RA and a BA of packet are *x* and *y*, respectively and the arrow means a packet will move at the next cycle. We assume that the length of packet is 1, the memory subsystem includes two buffers to store two packets and its scheduler makes one of two stored packets executed every cycle (although actual execution time is longer than one cycle). Round-robin arbitration [17] as a flow control of NoC routers is adopted to assign a channel and a buffer of the next node to one packet among several competing packets. At cycle 0, three packets, R2B0, R2B1 and R3B0 get competition for advance to the router interconnecting with the memory subsystem and then R2B0 wins. R0B1 is executed in the memory subsystem. At cycle 1, R2B0 advances to the router interconnecting with the memory subsystem and then R3B1 also advances to the empty router by the advance of R2B0. Then three packets, R2B1, R3B0 and R3B1 also get competition such that R3B0 wins. R0B0 but not R1B1 is executed for avoiding bank conflict in the memory subsystem. At cycle 2, by round-robin arbitration R3B0 advances. Then, the rest of two packets get competition such that R2B1 wins. R1B1 is executed in the memory subsystem. At cycle 3, bank conflict happens in the memory subsystem since current execution is a bank 0 access and two buffers are also filled with bank 0 accesses, where all RAs are different. It is difficult to avoid bank conflict completely in the conventional NoC with the small depth of buffer.

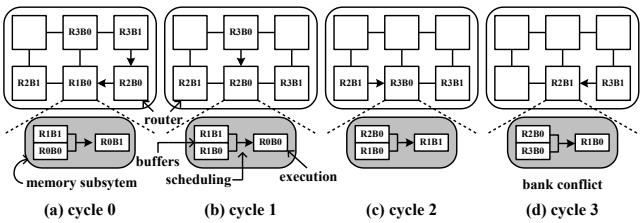


Figure 3: Conventional NoC with round-robin flow control

4.2 Basic Idea of SDRAM-Aware NoC Router

In our NoC design, the packet scheduling for an SDRAM access is performed by multiple SDRAM-aware routers. Consequently, the possibility of bank conflict becomes lower since packets arrive at the memory subsystem into the order that is friendly to the SDRAM operation. Fig. 4 shows how an NoC with our SDRAM-aware router works. At the first competition (cycle 0), the winner that advances to the router interconnecting with the memory subsystem is R2B1 that accesses bank 1 since the former access, R1B0 accessed bank 0. The rest of two packets may cause bank conflict since they have the same BAs but different RAs from the former access. At cycle 1, R2B1 advances and both competing packets, R2B0 and R2B1 can be a winner for advance but, R2B0 is chosen in this example. At cycle 2, R2B0 advances and R3B1 wins. Finally, R3B1 advances and R3B0 wins at cycle 3. Therefore, the proposed router prevents bank conflict better.

In addition, while it is desirable to use multiple SDRAMs for high performance, it is not desirable to use a corresponding number of memory subsystems. The reason is that the memory subsystem in

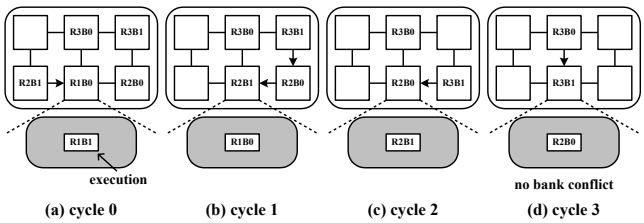


Figure 4: Novel NoC with SDRAM-aware flow control

Fig. 3 is too high in design cost due to a lot of buffers and a scheduler. Due to no buffer and no scheduler in the memory subsystem as shown in Fig. 4, our NoC design not only improves memory utilization and latency but also saves NoC design cost.

5. SDRAM-AWARE NOC ROUTER

Rather than a detail implementation, the proposed NoC router is about a novel paradigm for SDRAM-aware NoC exploration, which has a flow-control mechanism improve memory utilization and latency. Indeed, based on our idea, any deterministic and adaptive routing scheme can be combined to form our SDRAM-aware router. Another flow-control mechanism mentioned in section 2 can also be combined to avoid deadlock and livelock [17], to make traffic load balanced on a network [12-14] and to manage buffers and channel bandwidth [15].

5.1 Router Description

Our NoC router consists of input buffers, routing logics, SDRAM-aware flow controllers and an output schedulers as shown in Fig. 5. A packet is split into so-called flits (flow control digits) which are then routed and stored in a pipelined fashion. The input buffers are managed by wormhole flow control or virtual-channel flow control and a backpressure is used to inform the upstream nodes when they must stop transmitting flits because all of the downstream flit buffers are full. For our experiment, the wormhole flow control is implemented due to its simplicity and wide popularity [17] and an on/off flow control is adopted to avoid the loss of flits as the backpressure. Our SDRAM-aware router can be implemented to both deterministic and adaptive routers according to a routing logic that guarantees deadlock and livelock freeness. The approaches to solve deadlock are to use virtual channels [17] and deterministic dimension-ordered routings (e.g. XY routing, odd-even routing) [17]. We implement the XY routing that is a deterministic and minimal path routing algorithm, i.e. free deadlock and livelock. Simply to avoid another deadlock condition mentioned in [16], a master can send requests to another slave only after finishing requests for one slave.

In this router, over two flits arriving on different inputs buffers at the same time may both desire the same channel, where the final destination of flits is an SDRAM subsystem. In this case, the SDRAM-aware flow-control mechanism resolves this contention, allocating the channel to one packet and dealing with the others, blocked packets. Our SDRAM-aware flow control adopts winner-take-all bandwidth allocation that allocates all of the bandwidth to one packet until it is finished or blocked before serving the other packets [17]. Output scheduling detects if the buffers of the next node are available or expects when the buffers are available such that each SDRAM-aware flow control can choose the best path for low latency when multiple paths are given by the routing logic. In next section, a detail SDRAM-aware flow-control mechanism that uses a priority-based arbitration is present.

5.2 SDRAM-aware Flow Control

Our flow control allocates a channel to one of the competing flits whose destination is a memory subsystem controlling an SDRAM. Therefore, our flow-control mechanism performs an arbitration to determine which flit gets the channel it has requested. After the arbitration, the winning flit advances over this channel. Our arbitration algorithm also decides how to dispose of any flits that do not get their requested destination.

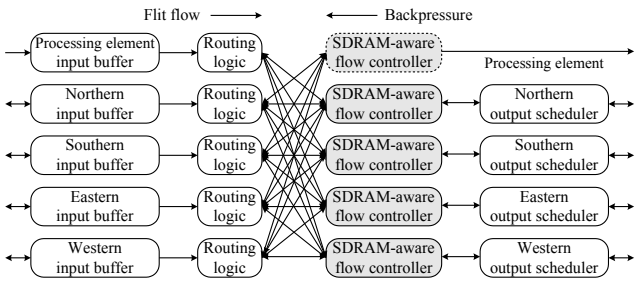


Figure 5: The proposed NoC router in mesh architecture

As shown in Algorithm 1, our arbitration is a priority-based algorithm, where the priority is composed from SDRAM awareness. The priority is assigned to all competing head flits which destination is a memory subsystem. Let $h(n)$ be the head flit of packet, which is already allocated a channel by the SDRAM-aware flow control at nth arbitration. Body and tail flits are assigned the same channel as their head flit. Let $h_i(n+1)$ be one of all competing head flits (I) that may be allocated the same channel as $h(n)$ at $(n+1)$th arbitration, where $i \in I$. The head flits, $h(n)$ and $h_i(n+1)$ contain addresses and command to access an SDRAM, denoted by (RA_n, BA_n, R/W_n) and ($RA_{n+1,i}$, $BA_{n+1,i}$, $R/W_{n+1,i}$), respectively, where the notations are (row address, bank address, read/write). At $(n+1)$th arbitration, all competing $h_i(n+1)$ are compared to $h(n)$ and then given a delay point from Table 2 (line 7) that is composed from DDR II SDRAM operating at 333MHz clock cycle [5] and that is changed according to the kind of SDRAM and operating speed.

Table 2 shows how many clock cycles are wasted when $h_i(n+1)$ is accessed in an SDRAM after $h(n)$. In Table 2, both case 1 and 10 have no loss of clock cycle since $h_i(n+1)$ gets the same R/W, BA and RA as $h(n)$, which means that data already stored in the row buffer of the same bank are accessed again by the same command. In case 3 and 12, data can be also accessed continuously since bank interleaving is completely performed as mentioned in section 3.2. Case 4 and 6 have one clock loss between the former read and the latter write access to avoid data contention. Case 7 and 9 waste 7 cycles to read data after writing data. Although bank interleaving is completely performed in case 7 and the continuous access of the same row buffer are performed in case 9, the later read command can be accepted t_{WRT} (internal write to read command delay) after finishing the former write. Then read data go out from an SDRAM after CL. In case 5, after reading data, its row buffer should be idle and then active since a write access with the same BAs but different RAs are executed. Then, data are written after write latency. Case 2 is already explained in section 3.2. Case 11 is a write-to-write access with the same BAs but the

Algorithm 1 SDRAM-Aware Flow Control Algorithm

Input: $h(n)$, $h_i(n+1)$ and table 2
1: **for** each $h_i(n+1)$, $i \in I$ **do**
2: **if** $h_i(n+1)$ is a new packet entering to the router **then**
3: $w_i = 0$;
4: **else**
5: $w_i = w_i$ + waiting cycles from the previous arbitration(n);
6: **end if**
7: d_i = delay cycle between $h(n)$ and $h_i(n+1)$ from table 2;
8: $p_i = w_i - d_i$;
9: **end for**
10: $h_i(n+1)$ with maximum(p_i) is allocated to a channel;
Output: $h(n+1)$

Table 2: Execution delay between *h(n)* and $h_i(n+1)$

Relation of *h(n)* with RW_n, BA_n, and RA_n and $h_i(n+1)$ with $RW_{n+1,i}$, $BA_{n+1,i}$, and $RA_{n+1,i}$				delay cycle (d_i) @ 333MHz DDR II SDRAM	case
RW_n=R	$RW_{n+1,i}$=R	BA_n=$BA_{n+1,i}$	RA_n=$RA_{n+1,i}$	0 cycle	1
			RA_n≠$RA_{n+1,i}$	t_{RP} (4 cycles) + t_{RCD} (4 cycles) + CL (4 cycles) = 12 cycles	2
		BA_n≠$BA_{n+1,i}$		0 cycle	3
	$RW_{n+1,i}$=W	BA_n=$BA_{n+1,i}$	RA_n=$RA_{n+1,i}$	1 cycle (timing gap between input and output not to collide)	4
			RA_n≠$RA_{n+1,i}$	t_{RP} (4 cycles) + t_{RCD} (4 cycles) + t_{DQSS} (1 cycle) = 9 cycles	5
		BA_n≠$BA_{n+1,i}$		1 cycle (timing gap between input and output not to collide)	6
RW_n=W	$RW_{n+1,i}$=R	BA_n=$BA_{n+1,i}$	RA_n=$RA_{n+1,i}$	t_{WTR} (3 cycles) + CL(4 cycles) = 7 cycles	7
			RA_n≠$RA_{n+1,i}$	t_{WR} (5cycles) + t_{RP} (4 cycles) + t_{RCD} (4 cycles) + CL (4 cycles) = 17 cycles	8
		BA_n≠$BA_{n+1,i}$		t_{WTR} (3 cycles) + CL (4 cycles) = 7 cycles	9
	$RW_{n+1,i}$=W	BA_n=$BA_{n+1,i}$	RA_n=$RA_{n+1,i}$	0 cycle	10
			RA_n≠$RA_{n+1,i}$	t_{WR} (5cycles) + t_{RP} (4 cycles) + t_{RCD} (4 cycles) + t_{DQSS} (1 cycle) = 14 cycles	11
		BA_n≠$BA_{n+1,i}$		0 cycle	12

different RAs. After the first writing, write recovery time (t_{WR}) is needed. Then, its row buffer is released (t_{RP}), filled with new data (t_{RCD}), and finally wrote after write latency (t_{DQSS}). Case 8 is the worst case since a read access with the same BAs but different RAs happens after writing data. The write access needs t_{WR} and the row buffer should be released, filled with new data and finally read after read latency (CL).

Our priority-based arbitration guarantees the upper bound latency since the low priority of packet may last for a long time. For example, let a packet with case 11 lose a competition against a packet with case 10. Nevertheless, since the priority condition is not changed at the next arbitration, the defeated packet keeps losing the next competition if it meets another packet with case 10. Therefore, the packet should escape from this competition after several defeats. To solve this starvation problem, our flow control counts the number of clock cycles passed from the first competition to the current competition (line 5) for each defeated packet. Then, this waiting time is subtracted by the delay cycle obtained in Table 2 (line 8). By this operation, any packet delayed for 17 clocks does not get a lower priority than a new packet entered in the router.

Finally, the packet with the maximum p_i is allocated the channel (line 10) and the blocked packets wait for the next competition or get another competition at a different SDRAM-aware flow controller if multiple routing paths are allocated by a routing logic. Thus, if an adaptive router instead of a deterministic router is employed in a routing logic, the performance would be better.

5.3 Hardware Implementation

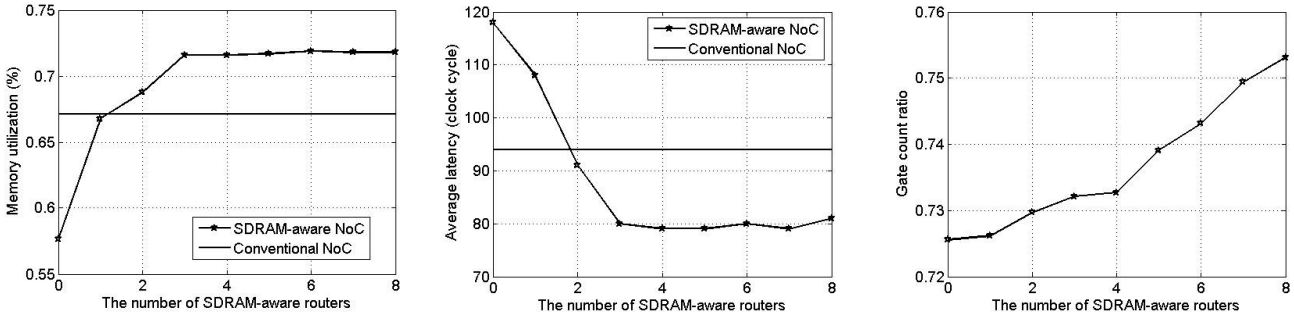
Simple logics are added in the SDRAM-aware router to compute the delay (d_i) and the waiting time (w_i) while the buffers and schedulers of memory subsystem are removed as shown in Fig. 4. The buffers in the memory subsystem are used to store several packets and then to reorder the packets such that an SDRAM is accessed as fast as possible. However, since the massive size of packet is generated in a GPU (Graphics Processing Unit) and a high-definition video system, the size of buffers also becomes larger and larger. Our NoC embedded with SDRAMs does not require the buffers in the memory subsystem since the scheduling is performed in multiple NoC routers. Furthermore, the size of input buffer in the router does not increase since the maximum four input buffers per router in a mesh network substitute for the buffers of memory subsystem by wormhole flow control and our multi-scheduling scheme gets the similar effect to the increase of input buffers. Consequently, the distinguished decrease of buffers in the memory subsystem exceeds the increase by the complexity of arbiter in multiple routers such that total chip area decreases. In addition, each router can get a different scheduler for a faster SDRAM access or for another purpose, which is one of the most important benefits of a multi-scheduling scheme.

6. EXPERIMENTAL RESULTS

An NoC with our SDRAM-aware router is implemented in Verilog HDL (Hardware Description Language). A memory subsystem operates with a DDR II SDRAM at 333MHz [5] and is implemented by the design concept from Sonics' MemMax [18] and Denali's Databahn [19], where MemMax is an SDRAM scheduler with four 32-flit buffers and Databahn is an SDRAM signal generator. Both are included in a conventional NoC design with a round-robin flow control based router. This is compared to our NoC design that includes multiple SDRAM-aware routers and a SDRAM signal generator instead of a full memory subsystem.

6.1 DTV Application

Our SDRAM-aware NoC router is applied to a Samsung DTV system that consists of 9 subsystems, i.e., an ARM9, an MPEG (Moving Picture Experts Group) decoder, a DNIE (Digital Natural Image Engine), a GPU, an audio decoder, a TS (Transport Stream) decoder, an AV (Audio and Video) format converter, a channel decoder and a memory subsystem, which are mapped to 3×3 mesh network. A conventional NoC router with round-robin flow control is gradually replaced by our SDRAM-aware router where the router that is the closest to a memory subsystem is replaced first and the router that is the farthest away from memory subsystem is replaced last. Fig. 6 shows the results according to the number of SDRAM-aware router placed to the order. In Fig. 6(a), the memory utilization of our NoC design starts 57% in case of no buffer, no memory scheduler and no SDRAM-aware router. When three SDRAM-aware routers are placed, its memory utilization increases by 72% (4.8% higher than the conventional NoC design). The reason why it is no longer improved in the case that over three SDRAM-aware routers are employed is that the bank conflicts and data contentions are almost removed. In Fig. 6(b), service latency is also shortened by 79 cycles (33% shorter than the conventional NoC design) since the high memory utilization lets a packet accessed faster with short waiting time and our flow control mechanism manages the upper bound latency. Our SDRAM-aware NoC design and the conventional NoC design are also synthesized by DesignVision from Synopsys with TSMC130LV library. The gate count of SDRAM-aware NoC design is 24.8% smaller as shown in Fig. 6(c) even if all



(a) Memory utilization (b) Average latency (c) Gate count ratio

Figure 6: Comparison in DTV application according to the number of SDRAM-aware routers

conventional NoC routers are replaced by our SDRAM-aware router. The reason is that large buffers and a memory scheduler in a memory subsystem are removed while the gate count increase caused by our flow-control mechanism is minimal. Thus, considering both the performance and the hardware cost, it is the best choice to replace three conventional routers to the SDRAM-aware routers in our DTV application. Our SDRAM-aware NoC router is also applied into dual DTV model [20] containing two memory subsystems. As the result, over 42% gate count is saved.

6.2 Experiments with Synthetic Benchmark

We evaluate the improvement of memory utilization and latency using 10 randomly generated applications based on industrial IPs (Intellectual Properties). The IPs mapped into 3×3 to 6×6 mesh topology generate 4 to 32 flits per packet at dynamic intervals. Each simulation runs for one million cycles. Table 3 shows our SDRAM-aware NoC improves memory utilization by 4.9% and latency by 18% on average compared to the conventional NoC. Especially, the improvement is higher in 6x6 NoC than in 3x3 NoC since packets passing through more SDRAM-aware routers are scheduled better for SDRAM operations. Thus, the improvement would be greater in a more complex NoC.

Table 3: Comparison in synthetic benchmarks

Content	router	3x3 NoC	6x6 NoC	average
memory utilization	SDRAM-aware NoC	63.7%	61.2%	64.6
	conventional NoC	59.4%	53.2%	59.7
	improvement	3.3%	8.0%	4.9%
memory latency	SDRAM-aware NoC	59 cycles	71 cycles	68 cycles
	conventional NoC	65cycles	99 cycles	83 cycles
	improvement	9.2%	28.3%	18%

7. CONCLUSION

We propose a NoC router including an SDRAM-aware flow-control mechanism. The proposed scheme allocates one of the competing packets toward an SDRAM, to a channel in our multiple routers based on SDRAM-awareness. Thus, the memory utilization and latency are more improved and the expensive buffers and scheduler of memory subsystem are removed. Furthermore, our SDRAM-aware router achieves better performance when it is employed in a more complex NoC design or when its routing scheme is adaptive. Our basic idea can be also extended to all SoCs with a flow controller and an SDRAM.

8. REFERENCES

[1] Luca Benini and Giovanni De Micheli, "Network on chips: a new SoC paradigm," Computer, 2002, 35, pp. 70-78.

[2] W. J. Dally and B. Towles, "Route Packets, Not wires: On-Chip Interconnection Networks," In Proc. Design Automation Conf., 2001.

[3] A. W. Topol, et al., "Three-dimensional integrated circuits," IBM J. Research and Development, vol. 4, 2006.

[4] F. Li, C. Nicopoulos, T. Richardson, Y. Xie, N. Vijaykrishnan, and M. Kandemir, "Design and Management of 3D Chip Multiprocessors Using Network-in-Memory," In Proc. International Symposium on Computer Architecture (ISCA), 2006.

[5] "Samsung DDR II SDRAM. Device operations & timing diagram," http://www.samsung.com/global/business/semiconductor.

[6] L. A. Vervoort, Philip Yeung, and Anil Reddy, "Addressing memory bandwidth needs for next-generation digital TVs with cost-effective, high-performance Consumer DRAM Solutions," http://www.rambus.com.

[7] S. Rixner, W. J. Dally, U. J. Kapasi, P. Mattson, and J. D. Owens, "Memory access scheduling," In Proc. ISCA, 2000.

[8] S. Heithecker and R. Ernst, "Traffic shaping for an FPGA based SDRAM controller with complex QoS requirements," In Proc. Design Automation Conference, 2005.

[9] W. D. Weber, "Efficient shared DRAM subsystem for SoCs," Sonics Inc., 2001, White paper.

[10] B. Akesson, K. Goossens and M. Ringhofer, "Predator: a predictable SDRAM memory controller," In Proc. CODES+ISSS, 2007.

[11] I. Hur and C. Lin, "Memory scheduling for modern microprocessors," ACM Trans. on Computer Systems, vol. 25, no. 4, Dec. 2007.

[12] F. Paganini, J. Doyle, and S. Low, "Scalable laws for stable network congestion control," In Proc. IEEE Conference on Decision and Control, 2001.

[13] Dongyu Qiu and Ness B. Shroff, "A Predictive flow control scheme for efficient network utilization and QoS," IEEE Trans. on Networking, vol. 12, no. 1, Feb. 2004.

[14] U. Y. Ogras and R. Marculescu, "Prediction-based flow control for network-on-chip traffic," In Proc. Design Automation Conf., 2006.

[15] W. C. Kwon, S. M. Hong, S. Yoo, B. Min, K. M. Choi, and S. K. Eo, "An open-loop flow control scheme based on the accurate global information of on-chip communication," In Proc. DATE, 2008.

[16] W. C. Kwon, S. Yoo, S. M. Hong, B. Min, K. M. Choi, and S. K. Eo, "A practical approach of memory access parallelization to exploit multiple off-chip DDR memories," In Proc. DAC, 2008.

[17] W. J. Dally and B. Towles, "Principles and practices of interconnection networks," Morgan Kaufmann, 2004.

[18] "Sonics MemMax," http://www.sonicsinc.com.

[19] "Denali Databahn: DRAM Memory Controller IP," http://www.denali.com.