Application-Aware NoC Design for Efficient SDRAM Access

Wooyoung Jang, *Student Member, IEEE,* and David Z. Pan, *Senior Member, IEEE*

***Abstract*—In systems-on-chip (SoCs), a microprocessor demands guaranteed synchronous dynamic random access memory (SDRAM) latency whereas most of the other cores are served as a best-effort packet. However, a priority service for the guaranteed latency causes the SDRAM utilization and latency of an overall system to be degraded critically. In addition, the data size of SDRAM requested by various cores is not matched with an SDRAM access granularity such that the SDRAM utilization and latency are further deteriorated. In this paper, we propose an application-aware networks-on-chip (NoCs) design for an efficient SDRAM access, which can consider memory latency demands and memory access granularities in various applications. In order to provide short latency for priority memory requests with few penalties, memory request packets are scheduled by our guaranteed SDRAM service router that includes a hybrid flow controller of priority-first and priority-equal algorithms. In addition, our SDRAM access granularity matching NoC design further improves the memory performance by splitting a memory request packet to several short memory request packets and then controlling the short memory request packets with a partially open-page mode and an auto-precharge operation in a memory subsystem. Experimental results show that our cost-effective application-aware NoC design significantly improves, on average, memory latency for latency-sensitive cores up to 32.8%, overall memory latency up to 7.8%, and memory utilization up to 3.4%, compared to the state-of-the-art SDRAM-aware NoC design [4].**

***Index Terms*—Access granularity, flow control, memory, networks-on-chip, quality-of-service, router.**

I. Introduction

WITH THE advance of semiconductor technology, many cores are integrated on a single chip and interconnected by on-chip networks, called networks-on-chip (NoCs) [1], [2]. In the NoC based multi-core processors, a memory service becomes one of the most important issues since its performance becomes the performance bottleneck of an overall system. However, its improvement aided by a single memory subsystem is severely limited since diverse applications generate their specific memory requests with various latency constraints and various sizes of data. In addition, the performance of applications considerably depends on the resource sharing policies employed in an on-chip network, which is one of the most critical shared resources in multi-core processors. Therefore, memory-aware NoC exploration and design have attracted great attentions in multi-core processor designs [3].

Recently, the responsibility for memory performance has been shared not only with a memory subsystem but also with an on-chip network. In [4], multiple NoC routers instead of a single memory subsystem schedule memory request packets that access a synchronous dynamic random access memory (SDRAM) for the purpose of encouraging row-buffer hit and bank interleaving but preventing bank conflict, data contention and short turn-around bank interleaving (which are introduced in Section III-A). As a result, the memory request packets can reach a memory subsystem in the order friendlier with SDRAM operations. In addition, the cost of overall hardware significantly reduced since a memory subsystem does not require a complex memory scheduler and a number of reorder buffers. This approach mainly provides a best-effort memory service since each SDRAM-aware NoC router equally manages all memory request packets. However, since the latest real-time applications request a memory service with short latency, a priority memory service should be provided for cores sensitive to memory latency. Furthermore, different cores request various sizes of SDRAM data. In the state-of-the-art multimedia system, the length of memory request packets requested by a video encoder/decoder such as H.264 [5] gets shorter whereas the length of memory request packets requested by a video enhancer/format converter gets longer. The long best-effort packets cause a priority packet to be further delayed. If any long best-effort packet is already scheduled in a router, a priority packet may wait until the best-effort packet is completely transferred to the next router. On the contrary, the short packets cause SDRAM utilization to be severely deteriorated. Since most SDRAMs receive or transmit fixed-length data per request, SDRAM data unnecessarily acquired may be thrown away. Therefore, a NoC design should consider the access granularity of diverse applications for an efficient SDRAM access.

In this paper, we propose an application-aware NoC design to efficiently access shared SDRAMs. Our key motivations are twofold. First, some cores request a guaranteed SDRAM service to an on-chip network and a memory subsystem. For

Manuscript received August 20, 2010; revised December 13, 2010 and March 8, 2011; accepted May 23, 2011. Date of current version September 21, 2011. This paper was supported in part by Samsung Electronics. This paper was recommended by Associate Editor R. Marculescu.

W. Jang was with the Department of Electrical and Computer Engineering, University of Texas, Austin, TX 78712 USA. He is now with the System Large Scale Integration Division, Samsung Electronics, Yongin 446-711, Korea (e-mail: wooyoung.jang@samsung.com).

D. Z. Pan is with the Department of Electrical and Computer Engineering, University of Texas, Austin, TX 78712 USA (e-mail: dpan@ece.utexas.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2011.2160176

example, a demand request generated by a microprocessor is usually served as a priority packet since the microprocessor may halt until the demand request is served. However, the priority packet causes overall memory latency and utilization (defined as the number of clock cycles used for data transfer divided by the number of total clock cycles) to be severely degraded. Since an on-chip network first serves the priority packet without any consideration of SDRAM operations, there exits strong possibility to meet bank conflict, data contention and short turn-around bank interleaving in SDRAM, which all make memory performance deteriorated. Therefore, the priority memory service should be considered not only in a memory subsystem but also in an on-chip network. In addition, since long best-effort packets may interfere with the fast service for the priority packet, they should be split to several short packets and then served. Second, different cores request various-length SDRAM data whereas DDR I/II SDRAMs always generate fixed-length data. Even if DDR III SDRAM can generate variable-length data, it has few advantages due to column access strobe (CAS) to CAS delay time (t_{CCD}) [6]. If the length of data requested by cores is not either the same as the length of data served by SDRAM or a multiple of the length of data served by SDRAM, unnecessary data may be accessed and then thrown away. Therefore, the access granularity mismatch problem, which causes memory utilization and latency to be deteriorated, is considered in our application-aware NoC design. Based on these motivations, the major novelties and contributions of this paper include the following.

1) We propose a guaranteed SDRAM service (GSS) router. It provides an efficient priority service for cores sensitive to memory latency.
2) We propose an SDRAM access granularity matching (SAGM) NoC design. Since a packet is split to several short packets of which the size is equal to or less than SDRAM access granularity and then served by our GSS router and memory subsystem, unnecessary SDRAM data can be less accessed.
3) We show the hardware architecture of our GSS router and memory subsystem working with a partially open-page policy and an auto-precharge (AP) operation.
4) We show the GSS router significantly improves memory latency for a priority packet with few penalties of overall memory utilization and latency. In addition, the SAGM NoC design not only recovers the penalties but also further improves overall memory performance.

To the best of our knowledge, this is the first work that addresses a NoC design improving the quality of SDRAM service through application-aware manners. The rest of this paper is organized as follows. In the next section, we survey related works. In Section III, we review basic SDRAM operations and scheduling schemes and then introduce two problems of conventional application-unaware NoC designs. Section IV presents the detail description of the proposed application-aware NoC design. Experimental results are shown in Section V. Finally, Section VI concludes this paper.

II. Related Works

Since guaranteed throughput and bounded latency are essential for NoC designs, many researchers have developed various approaches [7]. Æthereal NoC proposed in [8] provided a guaranteed service combined with a best-effort service employing variants of time division multiplexing. In [9], Nostrum NoC was implemented with the service of guaranteed bandwidth and latency in addition to the existing service of best-effort. In [10], MANGO using clockless circuit techniques was implemented. It exploited virtual channels to provide connection-oriented service guarantees and connection-less best-effort routing. Kim *et al.* proposed router architecture which utilized adaptive routing while maintaining low latency [11]. BiNoC supporting a self-configuring bidirectional channel mechanism for better bandwidth utilization and lower packet delivery latency was proposed in [12]. Das *et al.* [13] proposed efficient prioritization policies and architectural extensions to NoC routers that improved the overall application-level throughput, while ensuring fairness in the network. The prioritization policies were application-aware, distinguishing applications based on the stall-time criticality of their packets. In [14], they also proposed router prioritization policies that exploited the available slack of interfering packets in order to accelerate performance-critical packets and thus improved overall system performance. In [15], Kim *et al.* proposed a memory-centric NoC to support efficient pipelined task execution and coherence and consistency schemes tailored for 1-to-N and M-to-1 data transactions in a task-level pipeline. However, these approaches are not optimized for SDRAM request packets that cause the most critical latency.

Recently, microprocessors, shared buses, and NoC routers considering SDRAM operations have been developed to support a guaranteed memory service. Jang *et al.* [4] proposed an SDRAM-aware NoC design where multiple routers scheduled memory requests instead of a single memory subsystem. In [16], a memory bus was implemented to source-synchronous code division multiple access. A low-cost memory controller was present in [17] to maximize the benefit of useful prefetches and to minimize harms caused by useless prefetches. Cost-effective on-chip memory request issue mechanisms were proposed in [18] using SDRAM bank-level parallelism (BLP)-aware prefetch issue and BLP-preserving multi-core request issue. In [19], a network interface architecture was proposed to cope with in-order delivery, resource utilization, and latency. A memory controller was integrated into this network interface to improve memory utilization and reduced both memory latency and network latency. However, they all do not provide an efficient priority memory service or an access granularity matching solution.

III. Problem Description and Our Basic Idea

A. SDRAM Operation and Scheduling

SDRAM has a 3-D structure, i.e., a bank, a row, and a column. Basic commands to access SDRAM are row access strobe (RAS), CAS, and precharge (PRE). A bank becomes active by a RAS command and idle by a PRE command. A

CAS command can be executed to read or write data only after a bank is activated.

SDRAM consists of independent multiple banks (commonly 4 or 8 banks) whereas address and data pin/wire resources serialize accesses to different banks. The benefit of this architecture is that pin/wire resources can be saved and commands to different banks can be pipelined, i.e., while data are transferred to or from any bank, the rest of banks become idle and active for the latter request. Based on this principle, a memory subsystem schedules SDRAM requests to obtain successive SDRAM data.

SDRAM conditions improving memory performance are row-buffer hit and bank interleaving. First, the row-buffer hit condition is satisfied if an SDRAM request accesses any data already filled in a row buffer. As a result, since activation and deactivation operations are not executed, successive data can be received or transferred. Second, the bank interleaving condition is satisfied if an SDRAM request accesses a different bank from the previous SDRAM request. While the previous SDRAM request is served from any bank, the current SDRAM request is ready to access a different bank. The bank interleaving is the most useful in a highly parallel system where SDRAM requests generated by any core interfere with SDRAM requests generated by different cores.

On the contrary, there exist SDRAM conditions that worsen memory performance: bank conflict, data contention, and short turn-around bank interleaving. First, the bank conflict condition is met if an SDRAM request accesses the same bank as the previous SDRAM request with a different row address (RA). Since a bank activated by the former SDRAM request should get idle and then active for the latter SDRAM request again, a lot of clock cycles are required to complete these operations. Therefore, the bank conflict is the most critical factor in SDRAM performance. Second, the data contention is met if a write SDRAM request follows a read SDRAM request or a read SDRAM request follows a write SDRAM request. Data pins/wires are bidirectional in most SDRAMs whereas control and address pins/wires are unidirectional. As a result, input data may be collided with output data if there is no interval between a read access and a write access. Finally, the short turn-around bank interleaving mainly happens in particular, in high performance SDRAM such as DDR III SDRAM, where the useful bank interleaving may frequently achieve little improvement [20]. That is because a bank interleaved does not get sufficient time to be idle or active again due to long deactivation or activation delay time after the bank is accessed by the previous SDRAM request with a different RA.

B. Priority SDRAM Service in NoC

A microprocessor including a general processor, a cache, and a prefetcher commonly generates a demand request and a prefetch request. The demand request should be served as soon as possible since the microprocessor may stall until it receives a service of the demand request. On the contrary, the prefetch request does not need to be served with such a priority since it may be useless or not promptly used by the microprocessor. Memory requests of multimedia processors and peripherals

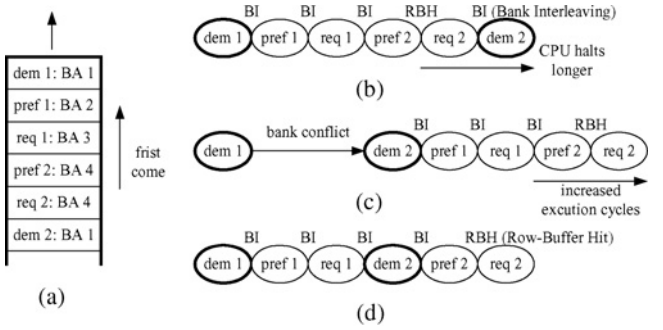


Fig. 1. Examples of scheduling memory requests. (a) Input buffer of router or memory subsystem. (b) Priority-equal scheduling (best-effort service). (c) Priority-first scheduling. (d) Our approach: hybrid of priority-equal and priority-first scheduling.

are commonly handled similarly to the prefetch request in the latest video/graphics systems.

Most of the conventional memory scheduler or NoC router takes two different approaches as to how to treat a priority request with respect to others. Figs. 1(b) and (c) show the operation of three different memory schedulers when two demand memory requests, two prefetch memory requests, and two memory requests by specific video processors are filled in their input buffer as shown in Fig. 1(a). In the figure, BA means a bank address and all requests are read operations. In addition, the RAs of all requests are different except prefetch 2 and request 2.

A memory scheduler providing a best-effort service as shown in Fig. 1(b) regards a priority memory request to have the same priority as others and then schedules all memory requests to avoid bank conflict, data contention, and short turn-around bank interleaving and to encourage row-buffer hit and bank interleaving. As a result, all memory requests are successively executed with no bank conflict whereas the execution of demand 2 is considerably delayed, which may cause the microprocessor generating demand 2 to halt for a long time. On the contrary, in Fig. 2(c), the demand requests are executed with a priority. This approach makes the demand requests executed early. However, since demand 2 accesses the same bank as demand 1 access with a different RA, bank conflict happens. It causes any data not to be delivered while the row buffer of bank 1 becomes deactivated and then is filled with the data of demand 2. Consequently, since total execution time of six requests is longer, memory utilization gets deteriorated. Therefore, a memory scheduler providing a priority service without the loss of memory utilization is required.

Fig. 1(d) is the most desirable scheduler that achieves the same memory utilization as the best-effort scheduler and the same memory latency for the demand requests as the priority-first scheduler. In order to achieve this performance, we propose a hybrid flow control algorithm that gets the advantage of the priority-equal and priority-first scheduler, which is fully described in Section IV-B. There may be strong possibility to meet bank conflict, data contention, and short turn-around bank interleaving if a demand request is separately considered on an on-chip network and in a memory subsystem. Therefore,

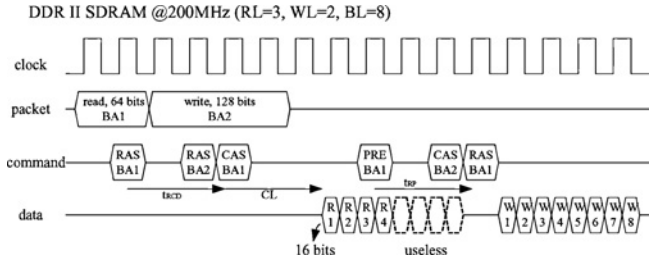


Fig. 2. Example of access granularity mismatch.

this scheduling is performed by multiple NoC routers which are similar to [4].

Moreover, we consider a long best-effort packet interfering with the fast service of priority packets. In the advanced video/graphics system, the length of a packet is longer and longer to provide a high-quality image. For example, the length of a packet generated by an industrial video enhancer/format converter reaches 64 burst lengths (BLs), which means that it takes at least 64 clock cycles to transfer the packet to the next router [21]. It is usually served as a best-effort packet. Let a router employing winner-take-all bandwidth allocation [22] schedule the long best-effort packet to any channel and then a priority packet reach in this router. If the router allocates the priority packet to the same channel as the long best-effort packet, the priority packet must wait until the long best-effort packet is completely delivered. In order to solve this problem, we split all packets to several short packets and then served. As a result, the priority packet can get more opportunities to be allocated to the channel. In the video/graphics system with 64-BL packets, if the best-effort packet is split to several packets with 4 BLs, a priority packet will wait for the maximum 4 clock cycles and then get the next competition. In the proposed application-aware NoC design, the length of a packet split is determined by an SDRAM access granularity introduced in the next subsection.

C. SDRAM Access Granularity Mismatch

SDRAMs transfer/receive fixed-length data (= the number of data bit $\times$ BL) per CAS command, called SDRAM access granularity. DDR I SDRAM has a BL 2, BL 4 and BL 8 mode and DDR II/III SDRAM has a BL 4 and BL 8 mode. In addition, since DDR III SDRAM has a selectable BL 4 or BL 8 on-the-fly (OTF) mode, it can deliver data with 4 or 8 BLs, depending on address 12 pin without any BL mode change. For example, if SDRAM with 16-bit data bus is set to a BL 8 mode via mode register set (MRS), it always generates 16 bytes per CAS command as shown in Fig. 2. On the contrary, any cores may request data with various lengths to SDRAMs. For example, an MPEG-1/2 and H.264 [5] encoder/decoder requests 8 or 16 bytes and 4, 8, or 16 bytes for motion estimation/compensation to SDRAM, respectively. If the MPEG-1/2 or H.264 encoder/decoder requests just 8 bytes as shown in Fig. 2, the rest of data unnecessarily accessed are thrown away, which seriously degrades memory performance.

Simple solutions are to reduce the number of data bits or to use a short BL mode in DDR SDRAM. If the number of data bits is changed to 8 bits, there exists no wasteful data. However, the overall system interfacing with SDRAM with 8-bit data bus does not have sufficient memory bandwidth to feed all cores. If more SDRAMs are interfaced with the entire system in order to increase the memory bandwidth, additional memory subsystems and pins/wires that are the limited resources are required. On the contrary, when short BL modes such as BL 2 and BL 4 are used in DDR SDRAM, command bandwidth exceeds data bandwidth such that it is difficult to hide commands behind data input/output time. The reason is that BL 2 and BL 4 have just one and two spaces where commands can be executed, respectively, whereas three spaces per SDRAM access are always required to execute three commands such as RAS, CAS, and PRE, except for a row-buffer hit condition. If there exists no row-buffer hit condition, memory utilization cannot exceed 33.4% and 66.7% in BL 2 and BL 4.

For this SDRAM access granularity mismatch problem, we focus on the latter approach using a BL 4 mode. In order to overcome the shortage of a command execution space, we use an AP operation in a memory subsystem. When AP is executed with a CAS command, the row buffer of an accessed bank automatically becomes idle without a PRE command after finishing transferring or receiving SDRAM data. In addition, a packet is split to several short packets with the same BL as SDRAM or less BL of SDRAM and then served by an on-chip router and a memory subsystem in our application-aware NoC design. As mentioned in Section III-B, splitting a packet to several short packets is also helpful to a priority service when a best-effort packet is too long. The detail approach is described in Section IV-C.

IV. Application-Aware NoC Design

Even with a perfect network routing algorithm and a perfect flow control algorithm mentioned in [7], a priority memory request may be significantly congested and delayed in a memory subsystem if it reaches the memory subsystem with the order unfriendly to SDRAM operations. In addition, if the length of data requested by cores is different from that of data served by SDRAM, data unnecessarily accessed are thrown away. In this regime, our attention shifts to an application-aware NoC design to improve not only the overall memory performance but also the quality of memory service.

A. Architecture of GSS Router

The proposed GSS router with p input/output ports consists of an input buffer, a routing logic, a flow controller, and an output scheduler as shown in Fig. 3. Typically, p is 5 and 7 for 2-D and 3-D mesh networks, respectively. The input buffers are managed by a wormhole flow control mechanism or a virtual-channel flow control mechanism. For our experiment, the wormhole flow control mechanism is implemented due to its simplicity and wide popularity [22]. The routing logic is responsible for determining the next router for each packet. Our GSS router can be implemented to either deterministic or adaptive routers according to a routing logic that guarantees both deadlock and livelock freeness. For our experiment, we

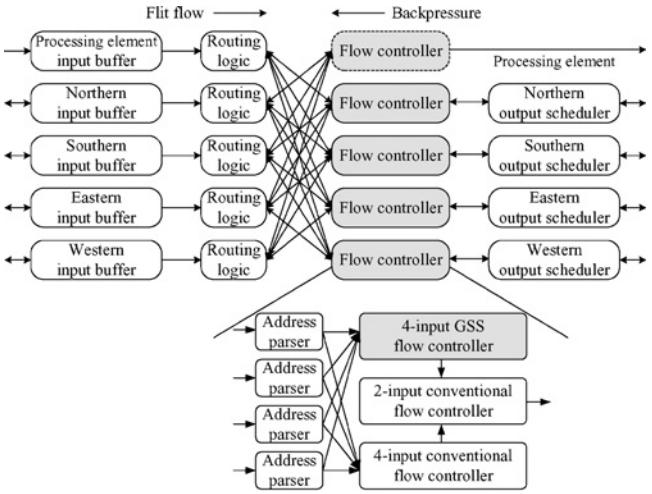


Fig. 3. Architecture of GSS NoC router for 2-D mesh.

implement XY routing that is a deterministic and minimal path routing algorithm such that it guarantees deadlock-free and livelock-free routing.

In this router, more than two different packets arriving on input buffers at the same time may desire the same channel toward a memory subsystem. In this situation, our GSS flow-control mechanism resolves this contention, allocating the channel to one packet and dealing with the others, blocked packets. In Fig. 3, our GSS flow controller is parallelly performed with the conventional flow controller. Each address parser sends an incoming memory request packet to our GSS flow controller and an incoming normal packet to the conventional flow controller. Our GSS flow controller schedules the memory request packets in order to prevent bank conflict, data contention, and short turn-around bank interleaving and provide a priority service at the same time. Then, the resulting memory request packet again competes with a normal packet by the conventional flow control mechanism. Hence, normal packets can reach their destination with no additional communication delay and interference. This parallel implementation can minimize an increase of timing critical path whereas its design cost is slightly expensive. In addition, our flow controllers adopt winner-take-all bandwidth allocation that allocates all of the bandwidth to just one packet until it is finished or blocked before serving the other packets.

An output scheduler either detects if an input buffer of the next router is available or expects when the input buffer is available. When the input buffer of the next router is full and a deterministic routing logic is implemented, an output scheduler makes the corresponding GSS flow controller stop scheduling packets. On the contrary, packets given multiple routing paths by an adaptive routing logic can be scheduled to other GSS flow controllers which are not busy.

In addition, we consider an ordering issue when a master core sends a read request to another slave core before the master core receives a read data from one slave core or when a master core requests another read data to a slave core in NoC employing an adaptive router before the master core receives one read data from the slave core. This ordering problem

Algorithm 1 GSS Flow Control

```
1:  if new packet h_k(n+1) comes in each router then
2:    for h_i(n+1) ∈ H(n+1) do
3:      t_i ← t_i+1;
4:      if h_k(n+1) is priory packet and its BA is equal to
          that of h_i(n+1) that is best-effort packet then
5:          h_i(n+1) is except from H(n+1);
6:      end if
7:    end for
8:    if h_k(n+1) is priority packet then
9:      t_k ← 2 to 5 (or 6); // PCT for Fig. 4(a) [or 4(b)]
10:   else
11:     t_k ← 1; // best-effort packet
12:   end if
13: end if
14: if h(n) finishes being delivered to the next router then
15:   for h_i(n+1) ∈ H(n+1) do
16:     T_i(t_i) in Fig. 4 ← h_i(n+1);
17:     T_i(0) in Fig. 4 ← h_i(n+1);
18:   end for
19:   if SP_PCT = ∅ then
20:     for h_i(n+1) ∈ H(n+1) do
21:       t_i ← t_i + 1;
22:     end for
23:     go to line 14;
24:   end if
25: end if
```

can be solved by various previous works including [23] or a following constraint: a master core can send a read request to a slave core only after the master core receives all requested data. The latter solution is employed in our implementation for simplicity. In addition, since our GSS flow control algorithm is performed with in-order buffers, the ordering problem does not happen in each GSS flow control.

B. GSS Flow Control Algorithm

In this section, we minutely present our flow control algorithm providing short latency for a priority memory request packet and similar overall memory utilization and latency. Let $h(n)$ be a packet, which is already allocated any channel by our GSS flow control at the nth arbitration. Let $h_i(n+1)$ be any packet i of all completing packets, $H(n+1)$, which may be allocated the same channel as $h(n)$ by our flow controller at the $(n+1)$th scheduling. The packets, $h(n)$ and $h_i(n+1)$, contain an address and a command to access 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). Thus, bank conflict, data contention, bank interleaving, and row-buffer hit conditions are defined as ($BA_n = BA_{n+1,i}$ and $RA_n \neq RA_{n+1,i}$), ($RW_n \neq RW_{n+1,i}$), ($BA_n \neq BA_{n+1,i}$), and ($BA_n = BA_{n+1,i}$ and $RA_n = RA_{n+1,i}$), respectively. Based on these notations and definitions, Algorithm 1 shows how our flow controller works for a guaranteed memory service, which consists of two parts.

First, a memory request packet (i) is given some tokens (t_i), depending on its input order and priority (lines 1–13). Let a

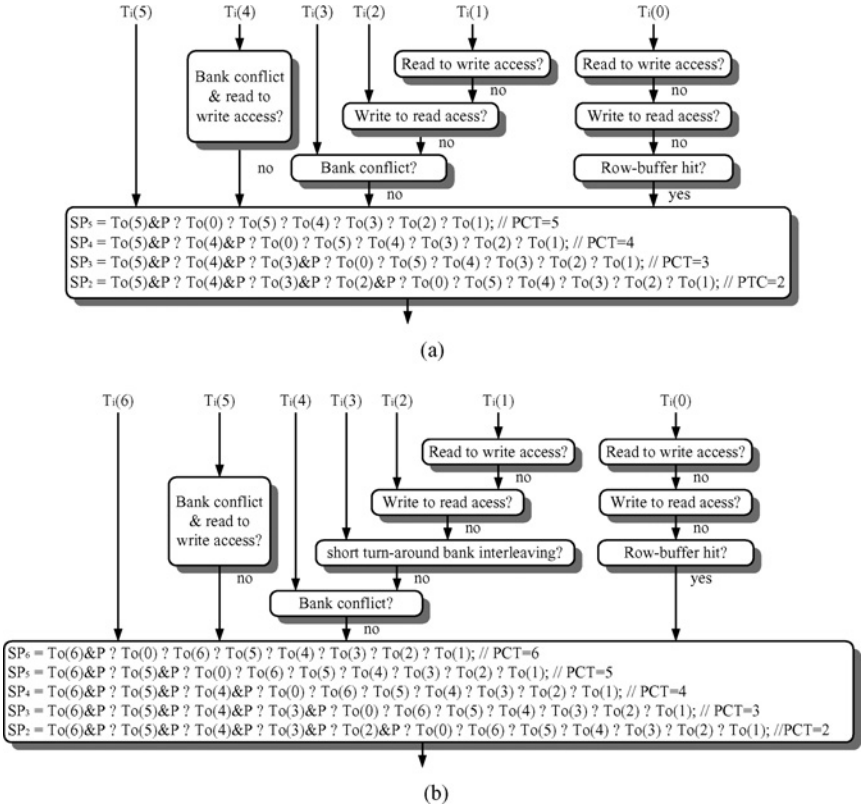


Fig. 4. Scheduling memory request packets for guaranteed SDRAM service considering (a) bank conflict and data contention, and (b) bank conflict, data contention and short turn-around bank interleaving.

new packet come in a router. All of the old packets are given to one additional token to avoid starvation (line 3). Then, if the new packet has a priority, old best-effort packets accessing the same bank as the priority packet are except from $H(n+1)$ (line 5). It means that old best-effort packets that access the same bank as any priority packet are not scheduled until the priority packet is scheduled. Then, the new packet gets an initial token. If it is a best-effort packet, one token is given (line 11). Otherwise, more than two tokens are given (line 9) to a priority packet by a user, called a priority control token (PCT). If a single token is given to the priority packet, it is equal to a priority-equal scheduler and if the maximum tokens are given to the priority packet, it is equal to a priority-first scheduler. Therefore, we can control the service speed of a priory packet by PCT.

Second, when $h(n)$ finishes being delivered, the rest of packets, $H(n+1)$ in the router are scheduled (lines 14–25). They all are input to Fig. 4, according to the number of tokens each packet has. That is, if any packet has 1, 2, 3, 4, 5, and 6 tokens, the packet is input to $T_{\mathrm{i}}(1)$, $T_{\mathrm{i}}(2)$, $T_{\mathrm{i}}(3)$, $T_{\mathrm{i}}(4)$, $T_{\mathrm{i}}(5)$, and $T_{\mathrm{i}}(6)$, respectively (line 16). All of the packets are also input to $T_{\mathrm{i}}(0)$ in line 17. As mentioned in Section III-A, a short turn-around bank interleaving problem is not critical for DDR SDRAM working at a low clock frequency since a short deactivation clock cycle and a reactivation clock cycle can be hidden behind the process of accessing a different bank. For such DDR SDRAMs, our flow controller just resolving bank conflict and data contention is shown in Fig. 4(a). On the contrary, since it takes a number of clock cycles to finish deactivation and reactivation in DDR SDRAM working at a high clock frequency, it is difficult for them to hide behind the process of accessing different banks. For example, in DDR III SDRAM working at an 800 MHz clock frequency, it takes 23 clock cycles to deactivate any bank after writing data [6]. Thus, until the written bank finishes being deactivated, a flow controller should make different banks accessed for 23 clock cycles to improve memory performance. Therefore, a flow controller working for such DDR SDRAMs should consider not only bank conflict and data contention but also short turn-around bank interleaving as shown in Fig. 4(b).

In order to check whether each bank finishes being idle, our flow controller has the same number of a counter as the bank of DDR SDRAM. After the last data are transferred to SDRAM, a counter corresponding to a bank written is set to $t_{WR} + t_{RP}$, where t_{WR} and t_{RP} are write recovery (WR) time and row precharge (RP) time, respectively [6]. On the contrary, after the last data are received from SDRAM, a counter corresponding to a bank read is set to t_{RP}. Then, the delay cycle stored in the counter is reduced by 1 every clock cycle. Thus, in Fig. 4(b), the short bank turn-around bank interleaving condition is defined as the counter corresponding to a bank accessed by $h_i(n+1)$ is greater than 0. If it is true, the bank is not ready to be activated again. Otherwise, the bank finishes being deactivated.

Finally, the packets are differently filtered in Fig. 4, depending on the number of token and the priority. If any packet has

a few tokens, which means an old packet or a priority packet, it is easy to pass this filter. After filtering all packets, if there is no packet passing the filter (line 19), all packets are given one additional token (line 21) and then go to the input of the filter again (line 23). Finally, if there are any packets passing the filter, one among the packets is output to SP_{PCT} (Scheduled Packet). If PCT is n in line 9, SP_n is used in Fig. 4 where $T_o(t_i)$ is the filtered output of $T_i(t_i)$. SP_n = A?B?C means A is chosen if A is not 0. If A is 0 and B is not 0, B is selected. Finally, if both A and B are 0 and C is not 0, C is chosen. In Fig. 4, a packet with a priority (P) and the most tokens is first selected. Next, a packet with $T_o(0)$ is selected. Lastly, a best-effort packet with the most tokens is selected. The reason that the packet with $T_o(0)$ is preferred to the best-effort packet with the most tokens is that there is strong possibility that $h(n)$ and $h_i(n+1)$ are split from the same packet. Why they are split from the same packet will be explained in the next section.

C. NoC Design for SAGM

It is useful to split a long packet into several short packets since on-chip network resources can be efficiently reserved and an SDRAM access granularity mismatching problem can be easily solved. That is, the optimal length of packets can improve memory/network utilization/latency. We split a packet to several short packets, depending on an SDRAM access granularity. Since our GSS routers communicate through a famous open core protocol (OCP) [24] or an AMBA AXI/AHP [25] protocol, packets consist of body flits but not head and tail flits including routing information. Instead, more controls and address buses include the routing information. Therefore, even if a packet is split to several short packets in each core and then is injected on a network, network loads do not increase.

As mentioned in Section III-C, DDR I/II SDRAMs always transfer/receive fixed-length data per read/write command after any BL mode is set in MRS. Most of the memory subsystems prefer a BL 8 mode in DDR I/II SDRAM because a BL 2 mode and a BL 4 mode can cause command bandwidth to be severely limited. As DDR SDRAMs transfer/receive two burst data per clock cycle, data are transferred/received for one and two cycles in the BL2 and BL4 mode, respectively. However, without any row-buffer hit, SDRAM needs three commands such as RAS, CAS, and PRE to obtain the short data. Therefore, the commands are so congested that the execution of commands is delayed. As shown in Fig. 5, we assume that a PRE command for BA 1 and a CAS command for BA 2 are issued at the same time. In Fig. 5(a), the PRE command is performed earlier than the CAS command. Consequently, the data of the second packet are written with some delays. In Fig. 5(b), the CAS command is performed earlier than PRE command. Consequently, the bank 1 gets idle and active with some delays. Therefore, such command congestion should be solved when short BL modes are used.

Fortunately, SDRAMs can omit a PRE command if a CAS command is executed with AP. The AP is enabled to provide a self-timed row precharge that is initiated at the end of burst access. As a result, both the PRE command and the CAS command are not delayed due to AP, as shown in Fig. 5(c). Under this consideration, it is useful that the BL (granularity)

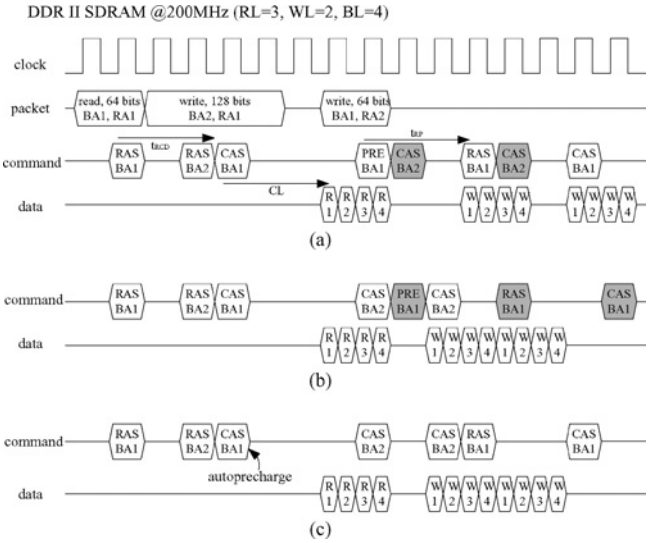


Fig. 5. SDRAM operations when BL is set to 4. (a) Delay of CAS commands (BA2). (b) Delay of PRE and RAS commands (BA1). (c) No delay of commands.

of packets is 2 and a BL mode in DDR I/II SDRAM is set to 4. Now that DDR III SDRAM has a selectable BL4 or BL8 OTF mode, it is useful that the BL of packets is 4 and a BL mode in DDR III SDRAM is set to 8. For example, if the BL of any packet is 9, it is split to five packets whose BLs are 2, 2, 2, 2 and 1 for DDR I/II SDRAM and it is split to three packets whose BLs are 4, 4, and 1 for DDR III SDRAM. It is efficient not only to match the access granularity but also to manage network resources. That is, a priority packet can be served faster in a winner-take-all bandwidth allocation policy. If the length of any best-effort packet is 9, a priority packet waits until all 9 bursts of the best-effort packet are transferred. If it is split like our approach, a priority packet wait until the maximum 2, 2, and 4 bursts of the best-effort packets are transferred in DDR I, II, and III SDRAM, respectively, and then get more opportunities to be allocated to a channel.

To implement this idea, we make a core generate short packets whose granularity is 2, 2, and 4 in DDR I, II, and III SDRAM, respectively, and the last packet has a tag to execute AP. Since the relation of packets split is row-buffer hit, there is not any loss of memory performance. As explained in Section IV-B, our GSS router prefers the row-buffer hit condition to the bank interleaving condition even if both do not cause any loss of memory performance. Therefore, if split best-effort packets do not meet any priority packet, they are scheduled successively. On the contrary, a priority packet is always scheduled without any interference.

Fig. 6 shows our memory subsystem. Since memory scheduling is performed in multiple GSS routers, our memory subsystem consists of an SDRAM controller, but not a complex memory scheduler and a number of buffers. Our SDRAM controller makes DDR SDRAMs work for a partially open-page mode. Each bank keeps an active state (open-page) after being accessed by a packet without any tag indicating the last packet split from a long packet. However, if a bank is accessed by a packet with a tag, the bank is deactivated (closed-page)

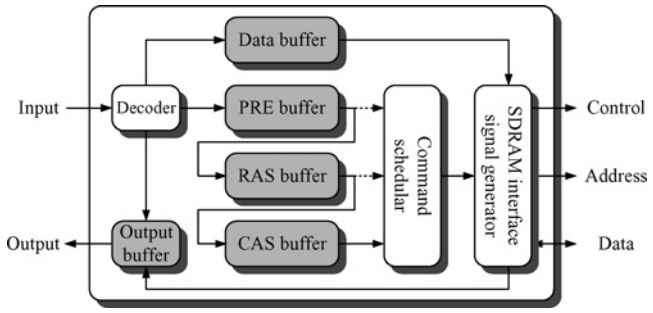


Fig. 6. SDRAM controller used in application-aware NoC design.

by AP. In addition, when a priority packet meets bank conflict relation with the previous best-effort packet, the bank is closed even if the previous best-effort packet has no tag. Our SDRAM controller works by this concept.

A memory request packet that is input to our SDRAM controller is decoded to extract SDRAM access information such as BA, RA, column address, the length of data, the type of a command, write data (if the command is a write request), and a master address. Then, the master address of read requests is stored in an output buffer and then used for building a memory service packet when requested data are received from SDRAM. The write data is stored in a data buffer and then used for generating an SDRAM interface signal for a write operation. The rest of SDRAM access information is stored in a PRE buffer. Then, the PRE buffer issues a PRE command only if a priority packet has any bank conflict relation with the previous best-effort packet without any tag. Since AP performing with a CAS command can be substituted for the PRE command, a number of PRE buffers are not required. The information stored in the PRE buffer is again stored to a RAS buffer. The RAS buffer issues a RAS command only if a packet does not have any row-buffer hit relation with the previous packet. The information stored in the RAS buffer is again stored in a CAS buffer. The CAS buffer always issues a read/write command. If a tag is attached to any information, its command is executed with AP. Next, all PRE, RAS, and CAS commands are scheduled by a command scheduler with a round-robin policy. Finally, an SDRAM interface signal generator builds SDRAM interface signals for each command and then sends them to SDRAM.

V. Experimental Results

Our application-aware NoC for an efficient SDRAM access is implemented in a Verilog hardware description language. We also implement the SDRAM-aware NoC design [4] and the conventional NoC design including a round-robin NoC router and a memory subsystem, called CONV. The memory subsystem interconnected to DDR SDRAM with 32-bits data bus employs the design concept from Sonics' MemMax [26] and Denali's Databahn [27]. MemMax offers a sophisticated thread-based pipeline and advanced arbitration schemes which prevent bank conflict and data contention conditions. Because there are no ordering requirements between threads, requests from different threads can be freely reordered. Different bandwidths and the qualities of service (QoSs) may be allocated to different threads to effectively support system data flow requirements. In MemMax, users can choose the depth of buffers, operation modes, and QoS settings that best suit various applications. Since MemMax supports OCP where request signals and data signals are separated, MemMax requires both a request buffer and a data buffer per thread. We use 4-thread MemMax where each thread requires a 32-flit request buffer and a 32-filt data buffer. The Databahn is an SDRAM controller that optimizes RAS, CAS, PRE and refresh operation. Since the Databahn employs command look-ahead to prepare pages in memory in advance of when commands execute, it can give class-leading performance even if the pattern of traffic is not known at design time. The MemMax and Databahn are employed in the conventional NoC design with a round-robin flow control based router. DDR I/II/III SDRAM modeled to Verilog files in [6] are interconnected to the SDRAM controller. The conventional NoC design and the SDRAM-aware NoC design set the DDR SDRAMs to a BL 8 mode via MRS. They are compared to our application-aware NoC design where DDR I/II SDRAM are set to a BL 4 mode and DDR III SDRAM is set to a selectable BL 4 or BL 8 OTF mode.

We use a Blu-ray model [5], a single digital television (DTV) model and a dual DTV model [21] as applications, which consist of 9, 9, and 16 cores, respectively. A memory subsystem is placed in a corner and the other cores are mapped to 3×3, 3×3, and 4×4 mesh network, respectively, by A3MAP [28] as shown in Fig. 7. These multimedia systems can work for various video sizes to measure memory performance in different DDR SDRAMs. For example, let dual DTV work for two video streams with 1920×1088 pixels, interfacing with 400 MHz DDR II SDRAM for real-time computing. If the dual DTV interfaces with 200 MHz DDR I SDRAM and 800 MHz DDR III SDRAM, it works for video streams with 1280×720 pixels and 2560×1600 pixels, respectively. All simulations run for one million cycles.

A. No Priority Memory Request

Our application-aware NoC design is first experimented when there is no priority packet. Since a demand packet generated by a microprocessor or a cache is not assigned to a priority packet, all packets receive a best-effort service. We implement the proposed application-aware NoC design to two versions. One is that only a GSS router is employed and the other is that both a GSS router and an SAGM design are employed in our NoC design, called GSS and GSS+SAGM, respectively.

Table I shows their memory performance, where the performance ratio is based on [4]. The GSS router achieves slightly better overall memory utilization and latency than [4] even if it is optimized for the latency of priority memory requests. On the contrary, the GSS router shows slightly worse latency of demand packets, compared to [4]. However, the latency of demand packets is not important since the demand packets are not assigned to a priority packet. Our NoC design employing both the GSS router and the SAGM design achieves not only higher memory utilization and shorter memory latency of

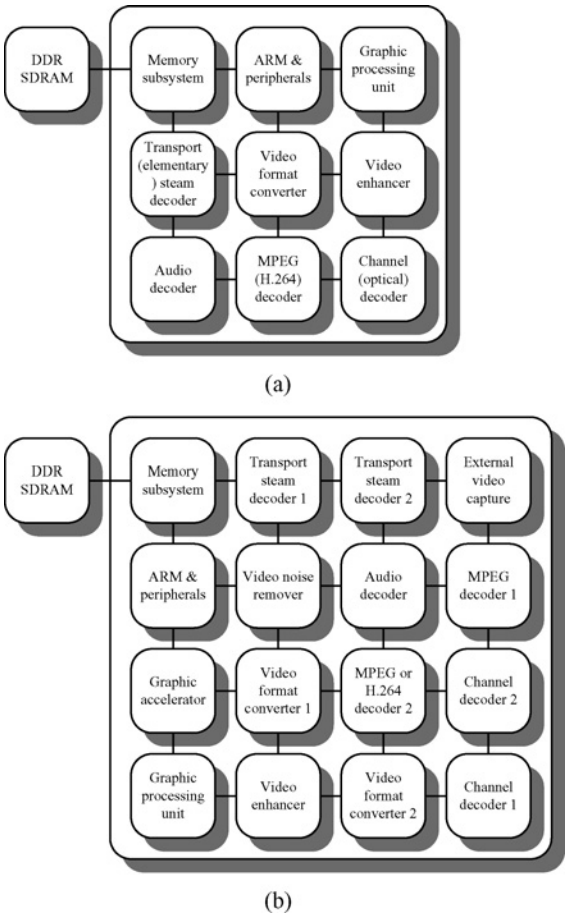


Fig. 7. Application mapping by A3MAP [28]. (a) Single DTV/Blue-ray model in 3×3 mesh network. (b) Dual DTV model in 4×4 mesh network.

overall requests, but also much shorter latency of the demand requests than [4] and the GSS router.

As shown in Table I, our application-aware NoC design with SAGM is the most useful for DDR II SDRAM where a read operation cannot be interrupted by any write and a write operation cannot be interrupted by any read and precharge operations. In DDR I SDRAM, a read operation can be interrupted by a burst stop command to support a short-burst data. However, since a write operation cannot be still interrupted, our SAGM design can improve memory performance in DDR I SDRAM. Now that DDR III SDRAM has a selectable BL4 or BL8 OTF mode, it looks perfect for the SAGM. However, in DDR III SDRAM, a CAS command can be performed only 4 clock cycles after the previous CAS command due to t_{CCD}. It makes DDR III SDRAM similarly work for a BL8 mode even if the BL mode is not set to 8. Therefore, our performance improvement in DDR III SDRSAM is less than that in DDR I/II SDRAM.

B. Priority Memory Request

We test our application-aware NoC design on priority packets. Since a demand packet generated by a microprocessor or a cache is assigned to a priority packet, it is served earlier than a best-effort packet. We also implement the conventional NoC design and the SDRAM-aware NoC with a priority-first service (PFS), called CONV+PFS and [4]+PFS, respectively.

Table II shows their memory performance, where the ratio is based on [4] in Table I. Our application-aware NoC design proves more merits when there exists a priority packet on NoC. [4]+PFS improves, on average, the latency of priority memory request packets up to 20.7%, compared to [4]. However, the memory utilization and latency of all packets are 8.3% and 23.3% worse than [4]. On the contrary, our GSS router improves, on average, the latency of priority memory request packets up to 23.7%, compared to [4]. The memory utilization and latency of all packets are just 1.7% and 2.9% worse than [4]. Compared to [4]+PFS, our GSS router improves, on average, 7.7% memory utilization, 16.5% latency of all packets, and 3.7% latency of priority packets. This result shows our GSS router has fewer penalties of memory performance than [4]+PFS to support a priority service.

Furthermore, GSS+SAGM further improves the memory performance since it accesses few SDRAM data unnecessary. GSS+SAGM achieves, on average, 4.7% higher memory utilization, 10.2% shorter memory latency of all packets, and 9.1% shorter memory latency of priority packets than GSS. Consequently, GSS+SAGM improves, on average, not only 32.7% latency of priority packets but also 3.4% memory utilization and 7.8% latency of all packets, compared to [4]. Compared to [4]+PFS, GSS+SAGM improves, on average, 12.7% memory utilization, 25.2% latency of all packets, and 15.2% latency of priority packets.

Fig. 8 shows the memory performance of our application-aware NoC design according to the number of GSS routers when a single DTV model (3×3), a Blue-ray model (3×3), and a dual DTV model (4×4) work with DDR I SDRAM at 200 MHz, DDR II SDRAM at 333 MHz, and DDR III SDRAM at 666 MHz, respectively. In the conventional NoC design, a router employing a priority-first and round-robin flow control algorithm is gradually replaced with our GSS router in the order where a router that is the closest to a memory subsystem is replaced first and where a router that is the farthest away from a memory subsystem is replaced last.

When any input buffer and any memory scheduler are not adopted in a memory subsystem and the conventional router is placed on a network, its memory utilization is just 69%, 56%, and 38% in a single DTV model, a Blue-ray model, and a dual DTV model, respectively, as shown in Fig. 8(a). However, whenever our GSS router is substituted for the conventional router, its memory utilization improves rapidly. As a result, when three GSS routers are substituted for three conventional routers, the memory utilization increases up to 77%, 73%, and 54% in a single DTV model, a Blue-ray model, and a dual DTV model, respectively. However, more than four GSS routers achieve little improvement of memory utilization since the solvable bank conflict and data contention are almost prevented by three GSS routers.

Fig. 8(b) shows the memory latency of all packets including both a priority packet and a best-effort packet. The memory latencies of all packets are initially 134 cycles, 157 cycles, and 332 cycles in a single DTV model, a Blue-ray model, and a dual DTV model, respectively. However, whenever the GSS

TABLE I
COMPARISON ON INDUSTRIAL BENCHMARKS WITHOUT PRIORITY MEMORY REQUEST

Application	Clock Speed	Memory Utilization				Memory Latency of All Packets (Cycle)				Memory Latency of Demand Packet (Cycle)			
		CONV	[4]	GSS	GSS+SAGM	CONV	[4]	GSS	GSS+ SAGM	CONV	[4]	GSS	GSS+SAGM
Blu-ray	133 MHz[a]	0.755	0.763	0.771	0.774	121	81	74	69	111	63	65	60
	266 MHz[b]	0.651	0.691	0.717	0.761	157	109	101	86	153	91	89	74
	533 MHz[c]	0.505	0.592	0.600	0.619	216	134	140	131	216	113	124	113
Single DTV	166 MHz[a]	0.717	0.737	0.766	0.776	144	101	86	71	140	80	74	61
	333 MHz[b]	0.625	0.673	0.715	0.756	173	120	108	91	171	96	94	77
	667 MHz[c]	0.463	0.554	0.577	0.596	244	154	143	140	248	126	127	119
Dual DTV	200 MHz[a]	0.696	0.707	0.708	0.712	154	104	89	80	128	73	67	57
	400 MHz[b]	0.555	0.627	0.627	0.682	246	149	141	115	196	107	104	85
	800 MHz[c]	0.426	0.559	0.531	0.547	364	191	195	184	266	133	144	128
Average		0.599	0.656	0.668	0.691	202	127	120	107	181	98	99	86
Ratio[d]		0.914	1.000	1.018	1.054	1.591	1.000	0.942	0.846	1.847	1.000	1.007	0.878

[a]DDR I SDRAM.
[b]DDR II SDRAM.
[c]DDR III SDRAM.
[d]Ratio is based on the SDRAM-aware NoC design [4].

TABLE II
COMPARISON ON INDUSTRIAL BENCHMARKS WITH PRIORITY MEMORY REQUEST

Application	Clock Speed	Memory Utilization				Memory Latency of All Packets (Cycle)				Memory Latency of Demand Packet (Cycle)			
		CONV+PFS	[4]+PFS	GSS	GSS+SAGM	CONV+PFS	[4]+PFS	GSS	GSS+SAGM	CONV+PFS	[4]+PFS	GSS	GSS+SAGM
Blu-ray	133 MHz[a]	0.729	0.742	0.77	0.774	141	106	77	72	97	59	42	38
	266 MHz[b]	0.612	0.621	0.699	0.745	176	134	112	96	123	73	72	60
	533 MHz[c]	0.454	0.517	0.561	0.608	248	166	151	138	179	88	98	90
Single DTV	166 MHz[a]	0.676	0.699	0.755	0.779	163	124	96	76	105	64	57	41
	333 MHz[b]	0.58	0.613	0.684	0.738	192	143	116	107	128	74	72	66
	667 MHz[c]	0.387	0.489	0.534	0.559	309	182	158	151	213	94	98	95
Dual DTV	200 MHz[a]	0.655	0.675	0.7	0.709	183	124	103	80	131	62	55	36
	400 MHz[b]	0.521	0.577	0.608	0.657	280	178	153	127	156	81	78	68
	800 MHz[c]	0.405	0.481	0.518	0.53	389	252	210	207	198	104	101	99
Average		0.558	0.602	0.648	0.678	231	157	131	117	148	78	75	66
Ratio[d]		0.85	0.917	0.987	1.034	1.821	1.233	1.029	0.922	1.508	0.793	0.763	0.672

[a]DDR I SDRAM.
[b]DDR II SDRAM.
[c]DDR III SDRAM.
[d]Ratio is based on the SDRAM-aware NoC design [4].

TABLE III
COMPARISON OF GSS + SAGM + STI TO GSS + SAGM ON INDUSTRIAL BENCHMARKS

Application	Clock Speed	Memory Utilization	Improvement	Memory Latency of All Packets	Improvement	Memory Latency of Priority Packet	Improvement
Blue-ray	533 MHz	0.674	10.9%	119 cycles	4%	79 cycles	12.2%
Single DTV	667 MHz	0.590	5.5%	140 cycles	7.3%	87 cycles	8.4%
Dual DTV	800 MHz	0.593	11.9%	161 cycles	22.2%	81 cycles	18.2%
Average		0.619	9.4%	140 cycles	11.2%	82 cycles	12.9%

router is substituted for the conventional router, the memory latency of all packets also improves rapidly. As a result, when three GSS routers are substituted for three conventional routers, the memory latency of all packets decreases up to 88 cycles, 98 cycles, and 191 cycles in a single DTV model, a Blue-ray model, and a dual DTV model, respectively.

Fig. 8(c) shows the memory latency of priority packets. The memory latencies of priority packets are 92 cycles, 122 cycles, and 146 cycles in a single DTV model, a Blue-ray model, and a dual DTV model, respectively, when any input buffer and memory scheduler are not adopted in a memory subsystem and the conventional router is placed on a network. However, when three GSS routers are substituted for three conventional routers, the memory latency of priority packets decreases up to 54 cycles, 63 cycles, and 95 cycles in a single DTV model, a Blue-ray model, and a dual DTV model, respectively. Therefore, three GSS routers placed around a memory subsystem show the most efficient result in terms of hardware cost and memory performance.

We also evaluate the improvement of memory performance when a short turn-around bank interleaving problem is considered in our application-aware NoC design, called GSS+SAGM+STI. For this experiment, we use three GSS routers employing Fig. 4(b) and execute a Blue-ray model,

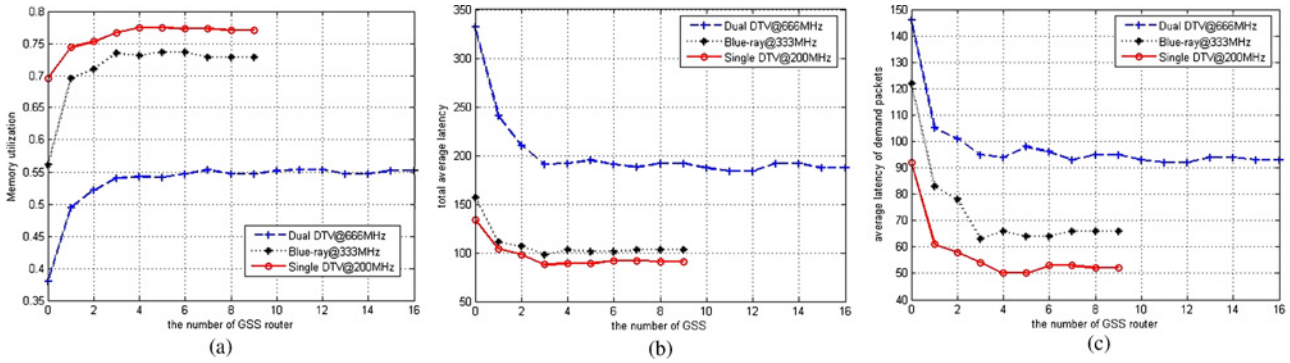


Fig. 8. Memory performance of our application-aware NoC design according to the number of GSS routers. (a) Average memory utilization. (b) Average latency for all packets. (c) Average latency for demand packets.

TABLE IV
GATE COUNT COMPARISON AT 400 MHZ CLOCK SPEED

Module	CONV		[4]		GSS+SAGM+STI	
	Gate Count	Ratio	Gate Count	Ratio	Gate Count	Ratio
Flow controller	3310	0.539	6732	1.097	6136	1
Router	56 683	0.904	62 949	1.003	62 721	1
Memory subsystem	489 898	3.283	158 874	1.065	149 245	1
3×3 NoC with memory subsystem	966 250	1.511	661 645	1.035	639 481	1

TABLE V
AVERAGE POWER CONSUMPTION COMPARISON

Application	Clock Speed	CONV		[4]		GSS+SAGM+STI	
		Power	Ratio	Power	Ratio	Power	Ratio
Single DTV	200 MHz	179.0 mW	1.550	116.0 mW	1.004	115.5 mW	1
Blue-ray	400 MHz	351.6 mW	1.550	227.8 mW	1.004	226.8 mW	1
Dual DTV	800 MHz	961.9 mW	1.328	726.0 mW	1.003	724.1 mW	1
Average		497.5 mW	1.399	356.6 mW	1.003	355.5 mW	1

a single DTV model, and a dual DTV model with DDR III SDRAM at 533 MHz, 667 MHz, and 800 MHz, respectively. The short turn-around bank interleaving problem is not critical in DDR SDRAM working at a low clock frequency. This is because a bank can be sufficiently deactivated and reactivated while any different bank is accessed. On the contrary, the short turn-around interleaving problem causes memory performance to be critically degraded in DDR SDRAM working at a high clock frequency. This is because the deactivation, activation, and WL/CL delay time are too long, compared to the length of data accessed. Table III shows that GSS+SAGM+STI achieves, on average, 9.4% higher memory utilization, 11.2% shorter memory latency of all packets, and 12.9% shorter memory latency of priority packets than GSS+SAGM.

CONV, [4], and the proposed NoC design are synthesized by Synopsys Design Vision with OSU PDK 45 nm CMOS standard cell library [29]. Table IV shows their gate count in case that they are optimized at 400 MHz clock speed. Our flow controller is 8.9% smaller than [4] even if it provides effective QoS and high throughput. This is because our GSS flow control mechanism for scheduling memory requests and avoiding starvation is optimized by event driven architecture. On the contrary, the gate count of our flow controller is 85.4% greater than that of a conventional flow controller due to the additional GSS flow control mechanisms. However, since the flow controllers are commonly tiny, the gate count increased or decreased by our GSS flow control mechanism has little impact on the area of whole NoC design. In addition, routers can be equipped with the minimum GSS flow controllers according to a routing policy. That is, any conventional flow controller through which a packet goes to a memory subsystem can be just substituted for the proposed flow controllers. Moreover, the GSS flow controllers can have fewer input ports. For example, if a memory subsystem is placed in the upper left corner on NoC as shown in Fig. 7, a router located in (2, 2) can have two 3-input GSS flow controllers. The GSS flow controllers schedule memory requests from processing element, south, and east inputs and are attached to a north and west output scheduler, respectively. As a result, the gate count of our router is just 10.7% greater than a conventional router and 0.4% less than [4], as shown in Table IV.

Our memory subsystem has great impact on the area of whole NoC design since it does not require any reordering buffers and any complex memory scheduler. Since memory requests are already scheduled by multiple routers with GSS flow controllers, the memory requests arrive at our memory subsystem with the order friendly with memory operations. In addition, our memory controller has fewer PRE buffers than a conventional memory controller and [4] due to effective AP operations. Thus, our memory subsystem is 69.5% and

6.1% smaller than a conventional memory subsystem and [4], respectively. Such a distinguished gate count decrease by removing reordering buffers and a memory scheduler in our memory subsystem far exceeds a gate count increase by GSS flow controllers in multiple routers. As a result, NoC with our memory subsystem and three routers with GSS flow controllers is 33.8% and 3.3% smaller than CONV and [4], respectively, as shown in Table IV.

We compute their power consumption by Synopsys Prime Time PX after gate-level simulation. Our application-aware NoC design consumes on average 28.5% and 0.3% less power than CONV and [4], respectively.

VI. Conclusion

In NoC, a microprocessor and a specific core that perform various applications request not only a best-effort memory service but also a priority memory service. In addition, they request memory data with various sizes which do not match an SDRAM access granularity. Therefore, we proposed an application-aware NoC design for an efficient SDRAM access. The proposed GSS router schedules a priority packet as fast as possible with the consideration of bank conflict, data contention, and short turn-around bank interleaving which all make memory performance severely degraded. Furthermore, the proposed SAGM NoC design splits a packet to several short packets, based on the BL of SDRAM and then serves them with a partially open-page mode and an AP operation in our memory subsystem. Experimental results showed our application-aware NoC design improved not only the memory utilization and latency of all packets but also the memory latency of priority packets in famous industrial multimedia systems, compared to the conventional NoC design and the state-of-art SDRAM-aware NoC design [4]. In conclusion, our application-aware NoC provides more opportunity for bandwidth-hungry system-on-chip designs with the high quality of a memory service.

References

[1] L. Benini and G. D. Micheli, "Network on chips: A new SoC paradigm," *Computer*, vol. 35, no. 1, pp. 70–78, Jan. 2002.

[2] W. J. Dally and B. Towles, "Route packets, not wires: On-chip interconnection networks," in *Proc. Des. Autom. Conf.*, 2001, pp. 684–689.

[3] N. Dutt, "Memory-aware NoC exploration and design," in *Proc. Des. Autom. Test Eur.*, 2008, pp. 1128–1129.

[4] W. Jang and D. Z. Pan, "An SDRAM-aware router for networks-on-chip," in *Proc. Des. Autom. Conf.*, 2009, pp. 800–805.

[5] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 560–576, Jul. 2003.

[6] DDR I, II, and III. *Device Operations and Timing Diagram* [Online]. Available: http://www.samsung.com/global/business/semiconductor

[7] R. Marculescu, U. Y. Ogras, L.-S. Peh, N. E. Jerger, and Y. Hoskote, "Outstanding research problems in NoC design: System, microarchitecture, and circuit perspectives," *IEEE Trans. Comput.-Aided Des. Integr. Curcuits Syst.*, vol. 28, no. 1, pp. 3–21, Jan. 2009.

[8] K. Goossens, J. Dielissen, and A. Rădulescu, "Æthereal network on chip: Concepts, architectures and implementations," *IEEE Des. Test Comput.*, vol. 22, no. 5, pp. 414–421, Sep. 2005.

[9] M. Millberg, E. Nilsson, R. Thid, and A. Jantsch, "Guaranteed bandwidth using looped containers in temporally disjoint networks within the nostrum network on chip," in *Proc. Des. Autom. Test Eur.*, 2004, pp. 890–895.

[10] T. Bjerregaard and J. Sparsø, "A router architecture for connection-oriented service guarantees in the MANGO clockless network-on-chip," in *Proc. Des. Autom. Test Eur.*, 2005, pp. 1226–1231.

[11] J. Kim, D. Park, T. Theocharides, N. Vijaykrishnan, and C. R. Das, "A low latency router supporting adaptivity for on-chip interconnects," in *Proc. Des. Autom. Conf.*, 2005, pp. 559–564.

[12] Y.-C. Lan, S.-H. Lo, Y.-C. Lin, Y.-H. Hu, and S.-J. Chen, "BiNoC: A bidirectional NoC architecture with dynamic self-reconfigurable channel," in *Proc. Int. Symp. Netw. Chip*, 2009, pp. 266–275.

[13] R. Das, O. Mutlu, T. Moscibroda, and C. R. Das, "Application-aware prioritization mechanisms for on-chip networks," in *Proc. Int. Symp. Microarchitecture*, 2009, pp. 280–291.

[14] R. Das, O. Mutlu, T. Moscibroda, and C. R. Das, "Aergia: Exploiting packet latency slack in on-chip networks," in *Proc. Int. Symp. Comput. Architecture*, 2010, pp. 1–11.

[15] D. Kim, K. Kim, J.-Y. Kim, S. Lee, S.-J. Lee, and H.-J. Yoo, "81.6 GOPS object recognition processor based on a memory-centric NoC," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 17, no. 3, pp. 370–383, Mar. 2009.

[16] J. Kim, B.-C. Lai, M.-C. F. Chang, and I. Verbauwhede, "A cost-effective latency-aware memory bus for symmetric multiprocessor systems," *IEEE Trans. Comput.*, vol. 57, no. 12, pp. 1714–1719, Dec. 2008.

[17] C. J. Lee, O. Mutlu, V. Narasiman, and Y. N. Patt, "Prefetch-aware DRAM controller," in *Proc. Int. Symp. Microarchitecture*, 2008, pp. 200–209.

[18] C. J. Lee, V. Narasiman, O. Mutlu, and Y. N. Patt, "Improving memory bank-level parallelism in the presence of prefetching," in *Proc. Int. Symp. Microarchitecture*, 2009, pp. 327–336.

[19] M. Daneshtalab, M. Ebrahimi, P. Liljeberg, J. Plosila, and H. Tenhunen, "A low-latency and memory-efficient on-chip network," in *Proc. Int. Symp. Netw. Chip*, 2010, pp. 99–106.

[20] W. Jang and D. Z. Pan, "An SDRAM-aware router for networks-on-chip," *IEEE Trans. Comput.-Aided Des. Integr. Curcuits Syst.*, vol. 29, no. 10, pp. 1572–1585, Oct. 2010.

[21] EE Times. (2004, Jul.). *Samsung Unveils HDTV System-on-Chip* [Online]. Available: http://www.eetimes.com/electronics-news/4049612/Samsung-unveils-HDTV-system-on-chip

[22] W. J. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. San Francisco, CA: Morgan Kaufmann, 2004.

[23] 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. Des. Autom. Conf.*, 2008, pp. 447–452.

[24] OCP-IP. *Open Core Protocol Specification, Release 2.0* [Online]. Available: http://www.ocpip.org

[25] ARM. *AMBA Open Specifications* [Online]. Available: http://www.arm.com

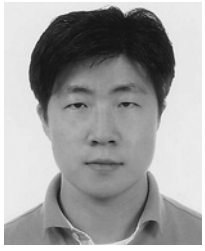
[26] Sonics, Inc. *MemMax Scheduler* [Online]. Available: http://www.sonicsinc.com

[27] Denali Software, Inc. *Databahn DRAM Memory Controller IP* [Online]. Available: http://www.denali.com

[28] W. Jang and D. Z. Pan, "A3MAP: Architecture-aware analytic mapping for networks-on-chip," in *Proc. Asian South Pacific Des. Autom. Conf.*, 2010, pp. 523–528.

[29] J. E. Stine, I. Castellanos, M. Wood, J. Henson, F. Love, W. R. Davis, P. D. Franzon, M. Bucher, S. Basavarajaiah, J. Oh, and R. Jenkal, "Free PDK: An open-source variation-aware design kit," in *Proc. IEEE Int. Conf. Microelectron. Syst. Educ.*, 2007, pp. 173–174.

[30] W. Jang and D. Z. Pan, "Application-aware NoC design for efficient SDRAM access," in *Proc. Des. Autom. Conference*, 2010, pp. 453–456.



Wooyoung Jang (S'08) received the B.E. degree in radio science and technology from Kyung Hee University, Suwon, Korea, in 1998, the M.S. degree in electrical and computer engineering from Yonsei University, Seoul, Korea, in 2000, and the Ph.D. degree in electrical and computer engineering from the University of Texas, Austin, in 2011.

Since 2000, he has been with the System Large Scale Integration Division, Samsung Electronics, Yongin, Korea, as a Senior Engineer. His current research interests include computer architecture and nanometer physical design for on-chip communication.

Dr. Jang was the recipient of the SK Telecom Scholarship from 1994 to 1997, the Samsung Outstanding Achievement Award in 2005, and the Samsung Scholarship from 2006 to 2011.

David Z. Pan (S'97–M'00–SM'06) received the Ph.D. degree (with honors) in computer science from the University of California, Los Angeles (UCLA), in 2000.

From 2000 to 2003, he was a Research Staff Member with the IBM T. J. Watson Research Center, Yorktown Heights, NY. He is currently an Associate Professor (with tenure) with the Department of Electrical and Computer Engineering, University of Texas, Austin. He has published over 140 technical papers in refereed journals and conferences, and is the holder of six U.S. patents. His current research interests include nanometer physical design, design for manufacturing, vertical integration of technology/computer-aided design (CAD)/architecture, and design/CAD for emerging technologies.

Dr. Pan served as an Associate Editor for four premier IEEE journals, including the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS I, and the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS II from 2006 to 2007. He also served as an Area Editor for the *Journal of Computer Science and Technology*, an Associate Editor for the IEEE CIRCUITS AND SYSTEMS (CAS) SOCIETY NEWSLETTER, and a Guest Editor of the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN Special Section on the International Symposium on Physical Design in 2007 and 2008. He is a member of the Design Technology Working Group of the International Technology Roadmap for Semiconductors. He has served as the Chair of the IEEE CANDE Committee and the ACM/SIGDA Technical Committee on Physical Design. He was the General Chair of ISPD 2008 and the Steering Committee Chair of ISPD 2009. He served in the technical program committees of major VLSI/CAD conferences, including ASPDAC (Subcommittee Chair), DAC (Subcommittee Chair), DATE, ICCAD (Subcommittee Chair), ISPD (Program Chair), ISQED (Topic Chair), ISCAS (CAD Track Chair), among many others. He has served as a Technical Advisory Board Member of Pyxis Technology, Inc. He received a number of awards for his research contributions and professional services, including the ACM/SIGDA Outstanding New Faculty Award in 2005, the NSF CAREER Award in 2007, the UCLA Engineering Distinguished Young Alumnus Award in 2009, the SRC Inventor Recognition Award three times in 2000 and 2008, the IBM Faculty Award four times in 2004–2006 and 2010, the DATE Best IP Award in 2009, the ASPDAC Best Paper Award in 2010, the ISPD Best Paper Award in 2011, the SRC Techcon Best Paper in Session Award twice in 1998 and 2007, the ICICDT Best Student Paper Award in 2009, the Dimitris Chorafas Foundation Research Award in 2000, the eASIC Placement Contest Grand Prize in 2009, the ISPD Routing Contest Award in 2007, and the ACM Recognition of Service Award in 2007 and 2008. He was an IEEE CAS Society Distinguished Lecturer from 2008 to 2009. He is a member of ACM/SIGDA, SPIE, and ASEE.