

An Efficient Quality-Aware Memory Controller for Multimedia Platform SoC

Kun-Bin Lee, *Member, IEEE*, Tzu-Chieh Lin, and Chein-Wei Jen, *Member, IEEE*

Abstract—The ongoing advancements in VLSI technology allow system-on-a-chip (SoC) design to integrate heterogeneous control and computing functions into a single chip. On the other hand, the pressures of area and cost lead to the requirement for a single, shared off-chip DRAM memory subsystem. To satisfy different memory access requirements for latency and bandwidth of these heterogeneous functions to this kind of DRAM memory subsystem, a quality-aware memory controller is important. This paper presents an efficient multilayer, quality-aware memory controller that contains well-partitioned functionality layers to achieve high DRAM utilization while still meet different requirements for bandwidth and latency. Layer 0, also called memory interface socket, is a configurable, programmable, and high-efficient SDRAM controller for designers to rapidly integrate SDRAM subsystem into their designs. Together with Layer 1 quality-aware scheduler, the memory controller also has the capability to provide quality-of-service guarantees including minimum access latencies and fine-grained bandwidth allocation for heterogeneous control and computing functional units in SoC designs. Moreover, Layer 2 built-in address generator designed for multimedia processing units can effectively reduce the address bus traffic and therefore further increase the efficiency of on-chip communication. Experimental results of a digital set-top-box emulation system show that the access latency of the latency-sensitive data flows can be effectively reduced by 37%–65% and the memory bandwidths can be precisely allocated to bandwidth-sensitive data flows with a high degree of control.

Index Terms—Memory controller, multimedia application, quality-aware scheduler (QAS).

I. INTRODUCTION

MULTIMEDIA processing technologies have been widely applied in many systems. These technologies have not only provided existing applications like desktop video/audio but also spawned brand new industries and services like digital video recording, video-on-demand services, high-definition TV, etc. The confluence of hardware and software technologies has given computers the ability to process dynamic media (video, animation, music, etc.) where before they could handle only static media data (text, images, and so on). To support complex multimedia applications, architectures of multimedia systems must provide high computing power and high data bandwidth. Furthermore, a multimedia operation system should support real-time scheduling and fast interrupt processing [1].

Manuscript received October 2, 2003; revised January 31, 2004. This work was supported by the National Science Council of the Republic of China under Contract NSC-91-2215-E-009-033.

K.-B. Lee and T. C. Lin are with the MediaTek Corporation, Science-Based Industrial Park, Hsinchu 300, Taiwan, R.O.C. (e-mail: kblee@twins.ee.nctu.edu.tw; tclin@twins.ee.nctu.edu.tw).

C.-W. Jen is with the Department of Electronics Engineering, National Chiao Tung University, Hsinchu 300, Taiwan, R.O.C. (e-mail: cwjen@twins.ee.nctu.edu.tw).

Digital Object Identifier 10.1109/TCSVT.2005.846412

Previous researches have shown that data transfer and storage of multidimensional array signals dominate the performance and power consumption in system-on-a-chip (SoC) designs. [2], [3]. In particular, the data transfer to off-chip memory is especially important due to the scarce resource of off-chip bandwidth. In fact, although the tremendous progress in VLSI technology provides an ever-increasing number of transistors and routing resource on a single chip, and hence allows integrating heterogeneous control and computing functions to realize SoCs, the improvement on off-chip communication is limited due to the number of available input/output (I/O) pins and the physical design issues of these pins. As many recent studies have shown, the off-chip memory system is one of the primary performance bottlenecks in current systems. For example, Hennessy and Patterson showed that while micro-processor performance improved 35% per year until 1986, and 55% per year since 1987, the access time to DRAM has been improving about 5% per year [4]. Rattner illustrated that whereas a Pentium Pro required 70 instruction cycles for a DRAM access, a Pentium 4 running at 2 GHz takes 500 to 600 cycles [5]. Even the performance of DRAM is ever-improved, the system overheads like turnaround time and request queuing account for a significant portion of inefficiencies in memory access. Schumann reported that 30%–60% of primary memory latency is attributable to system overhead rather than to latency of DRAM components in Alpha workstations [6]. For multicore SoC designs, the performance of memory subsystem is even more important, due to the share of memory bus with different access requirements of these heterogeneous cores.

Recognizing the importance of high performance off-chip DRAM communication as a key to a successful system design, several SDRAM controllers and schedulers have been proposed to make the most efficient use of the off-chip DRAM memory subsystem. For single-processor environments, several approaches have been presented to improve memory bandwidth utilization. McKee's stream memory controller (SMC) reorders memory references among streams [7], whereas Rixner's memory bank controller for each DRAM chip reorders both memory references among streams and within a single stream [8]. Several problems come with reordering the DRAM accesses within a single stream. First, the reordering efficiency is quite sensitive to the number of accesses visible to the access scheduler during each clock cycle. Hence, a large amount of register files are needed to hold the arriving DRAM accesses, which inevitably increase the area of the design. Second, since the DRAM accesses may be completed out of order, extra circuits are required to reorder the read data back to the original order to maintain data consistency, which in turns might lead to a reduction in the efficiency of the reorder scheme on the memory bandwidth improvement. Furthermore, these extra

circuits also increase the area overhead. Addressing to the aforementioned problem of out-of-order return of read data, Ayukawa's access-sequence control scheme [9] allocates an access ID to each DRAM access. Therefore, the processor can identify the original order of the read data according to the access ID. This solution, however, is only suitable for specific bus protocols and processing units (PUs) that have the capability to assign and identify access IDs. Instead to reorder memory accesses, Takizawa presented a memory arbiter to increase the bandwidth utilization by reducing bank conflicts (i.e., row miss) and read/write turnarounds for the multicore environment [10]. The arbiter lowers the priority of a DRAM access if the access is addressed to the same bank as the previous granted access, or the access direction (read or write) is different from that of the previous granted one. Hence, the possibility of bank conflicts and read/write turnarounds can be diminished.

The above-mentioned SDRAM controller designs only address to the improvement of the overall bandwidth utilization. However, PUs in a heterogeneous system usually require different services of memory access bandwidth and latency. Therefore, these SDRAM controllers need to cooperate with DRAM schedulers to provide proper DRAM services for these PUs. In our observation, traditional DRAM scheduler designs have one or more of the following limitations:

- 1) the unawareness of DRAM status leading to low scheduling efficiency on both DRAM bandwidth utilization and access latency;
- 2) the lack of control over the bandwidth allocation for different PUs (e.g., fixed-priority scheduler) leading to starvation of low-priority PUs in some situations;
- 3) the significant access latencies due to the fair scheduling policies (e.g., round-robin scheduler) leading to unbearable long access latencies for high-priority PUs.

Knowing the above limitations of the conventional DRAM scheduler designs, Sonics' MemMax memory scheduler provides quality-of-service (QoS) guarantees of a single, shared off-chip DRAM memory subsystem for multiple heterogeneous PUs by using a tiered filtering system [11]. However, MemMax is deeply coupled with Sonics' SiliconBackplane μ Network that is time-shared with these PUs. Access ownership of SiliconBackplane μ Network is determined by a two-level arbitration scheme: the first level of arbitration is based on a time-division multiplexing (TDM) mechanism, while a second, lower priority access scheme is based on a round-robin token passing mechanism [12]. Due to the inherent latency limitations of the TDM on-chip communication mechanism [13], MemMax cannot effectively provide short latency services and only has better improvement on bandwidth utilization. Similarly, a three-level memory arbitration scheme proposed in [14] can be used for systems where both continuous high-throughput and random low-latency requests present. However, this three-level memory arbitration scheme does not take the status of SDRAM into account. In addition, the first-come-first-serve scheme used for the arbitration of the continuous streams in [14] also lowers the memory bandwidth utilization.

The aforementioned critical issues in integrating heterogeneous control and computing functions into a single chip and the awareness of none of available designs has the capability to

solve these issues motivate us to explore an efficient solution of off-chip SDRAM memory controller for multimedia platform SoCs. The goal of this memory controller is to provide not only the high utilization of DRAM bandwidth but also the QoS guarantees, which are defined as:

- fair distribution of bandwidth over bandwidth-sensitive processing units;
- shortest possible transaction latency for latency-sensitive processing units.

In addition, most multimedia processing units have regular address patterns. Having built-in address generator (BAGs) in the memory controller can reduce the address bus traffic and therefore increase the efficiency of on-chip communication. On the other hand, because not every system needs the same requirement of memory usage, a well-partitioned architecture of a memory controller can let system designers choose and integrate the required functionality of the memory controller into their systems.

In this paper, we present a multilayered, quality-aware memory controller to provide high DRAM bandwidth utilization and QoS guarantees for multimedia SoCs. The contribution of this paper includes the following.

- 1) We propose a layered architecture of the memory controller to efficiently decouple different functionality, including memory-specific control, quality-aware scheduling, and BAGs into different layers of the controller. Different combinations of these layers can produce memory controllers with different capabilities to meet distinct system requirements.
- 2) We propose a high efficient and flexible SDRAM memory interface socket (MIS) to take charge of SDRAM-specific control and make the best use of SDRAM bandwidth by support parallel access of each bank within SDRAM. In addition, MIS can respond to the requests from the SDRAM scheduler immediately to furthermore make the best use of SDRAM command and data bus. On the other hand, the flexibility of MIS is based on the configurable, shared-state finite-state machine (FSM) design that can easily be adjusted for different complex timing control latencies of SDRAM.
- 3) We propose a quality-aware scheduler (QAS) that not only improves the SDRAM bandwidth utilization by considering the SDRAM status and the relations of SDRAM accesses, but also provides QoS guarantees, i.e., minimum access latency and guaranteed bandwidth services based on the memory access requirements of different PUs.

The rest of this paper is organized as follows. Section II briefly introduces the basic characteristics of SDRAM and presents typical situations which efficient memory access should take into account. In Section III, the architecture of proposed multilayered quality-aware memory controller is explained in detail. Section IV shows the experiment results, including some constrained random experiments under different parameters and a simplified STB SoC to examine the QoS performance of several DRAM controllers. Finally, concluding remarks are made in Section V.

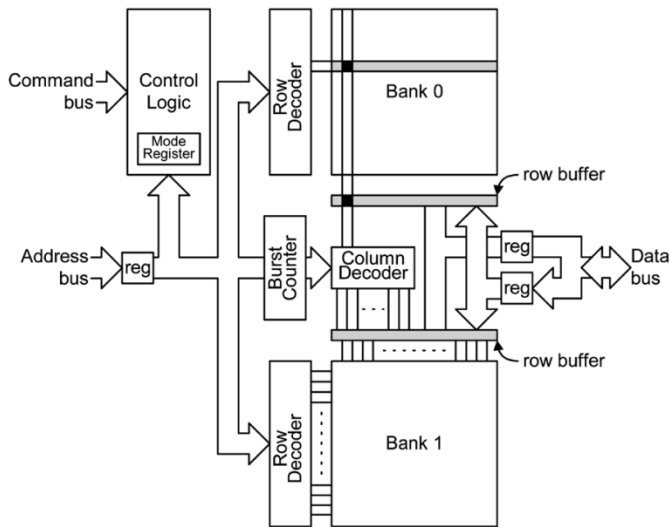


Fig. 1. Simplified architecture of a two-bank SDRAM.

II. SDRAM BASICS

Fig. 1 shows a simplified architecture of a two-bank SDRAM. All memory banks share the data and address bus, whereas each bank has its own row decoder, column decoder and row buffer. The mode register stores several SDRAM operation modes such as burst length, burst type, CAS (column address strobe) latency, etc. An m -bank SDRAM has a similar architecture. A complete SDRAM access may consist of several commands including row-activation, column-access (read/write) and precharge, as shown in Fig. 2. A row-activation command, together with the row and bank address, is used to open (or called activate) a specific row in a particular bank, and copy all data in the selected row into the selected bank's row buffer for the subsequent column accesses. After accepting this command, SDRAM needs a latency called t_{RCD} (ACTIVE to column access delay) to accomplish the command. No other commands can be issued to this bank during this latency. However, commands to other banks are permissible due to the independent parallel processing capability of each bank. Once a row of a particular bank has been opened, a column-access command can be issued to read/write data from/to the addressed word(s) within the row buffer. To issue either a read or write column-access command, DRAM address bus is also required to indicate the column address of the open row in that bank. For a write access, DRAM data bus is needed to transfer write data from the time the command is issued until the whole burst transfer is completed. As for a read access, DRAM data bus is used to transfer data after a latency called CAS, which is the time from the read column-access command is registered to the first read datum is available. The precharge command, together with the information on address bus, can be used to deactivate a single open row in a particular bank or all rows in all banks. While processing the precharge command, the addressed bank or banks are not allowed to accept any other commands during a time called t_{RP} (PRECHARGE command period.)

SDRAM bandwidth utilization and latency is lower and longer, respectively when more commands are required for a SDRAM access. The number of commands needed for a

complete SDRAM access deeply depends on the state of the bank addressed by the SDRAM access. Fig. 2(a) and (b) show a simplified bank state diagram and the access latencies due to different access statuses: bank miss, row miss, and row hit. In a bank miss status, an incoming access is addressed to a bank in the IDLE state, therefore it must first activate the target row and then issue the column access command. For a row miss status, the addressed bank is in ACTIVE state and the row address of its activated row is not identical to that of an incoming access. In this case, the incoming access has to first precharge the bank, then activate the target row, and finally issue column-access commands. As for a row hit status, the addressed bank is in ACTIVE state and the row address of its activated row is the same as that of the incoming access. Hence, column-access commands can be directly issued. The above discussion is only based on a simplified condition. A more complete discussion on various access latencies of a SDRAM access can be found in [15].

III. MULTILAYERED QUALITY-AWARE MEMORY CONTROLLER

Fig. 3 shows the configurations of different layers of the proposed memory controller. Layer 0 MIS is a configurable, programmable, and high-efficient SDRAM-specific controller for basic SDRAM operations, such as SDRAM initialization, refresh function, etc. Basically, MIS accepts access requests and translates them into proper command sequences according to the DRAM access status mentioned in the preceding section. We have designed a self-generating, tool-independent MIS silicon intellectual property (IP) to alleviate the burden for system designers [15]. This IP, called MIS-I in this paper, is featured with its parameterized and blockwisely design which is characterized by a rich set of choices of functionality, performance, interface and testbench. To improve SDRAM bandwidth utilization and access latency, an improved MIS-I, called MIS-II, is presented in this paper. Together with Layer 1 QAS, the memory controller also has the capability to provide QoS guarantees for heterogeneous control and computing PUs in multimedia SoC designs. Moreover, Layer 2 BAG designed for multimedia processing units can effectively reduce the address bus traffic and therefore further increase the efficiency of on-chip communication.

A. Configurable Shared-State FSM Design

Traditionally each control state of a FSM occupies one state of the FSM. However, this design style makes the FSM less flexible when there are many repeated control states in a FSM. For example, to complete a SDRAM read access with burst length of four in the row miss status, the timing diagram and a conventional FSM design for this access are shown in Figs. 2(b) and 4(a), respectively. These control latencies depend on both the specification provided by the SDRAM vendor and the clock frequency the memory is clocked. In addition, the unit of these latencies specified in the SDRAM datasheet is on the basis of real time (e.g., nanoseconds and microseconds), whereas the design of FSM that controls this sequence is clock cycle based. When the system clock is varied, these control latencies are varied in terms of clock cycle count. Conventional FSM design manually

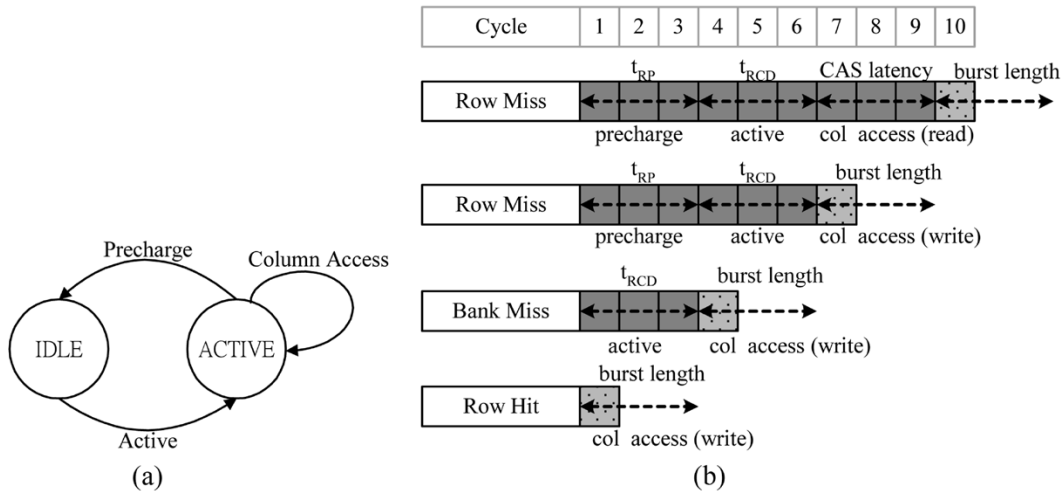


Fig. 2. (a) Simplified bank state diagram. (b) Access latencies of different access statuses.

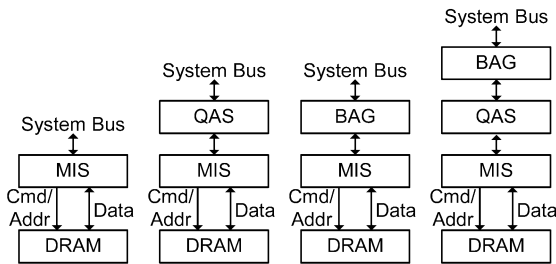


Fig. 3. Configurations of different layers of the proposed memory controller.

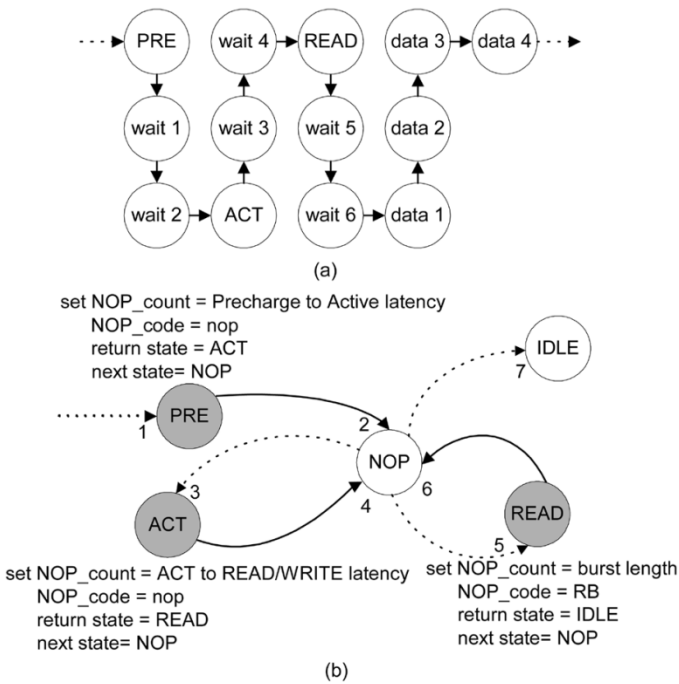


Fig. 4. FSM design in (a) traditional DRAM controller and (b) MIS-I.

calculates the control latencies from the relation between the timing constraints in datasheet and the clock frequency of the target system, and then fixes these latencies as states shown in Fig. 4(a). Changes in the control latencies or the numbers of access data force us to manually modify the design of FSM. For a

flexible and reusable design, these hard-coded states should be reduced or eliminated.

To make FSM more flexible, MIS unifies the repeated control states into a single control state of FSM as indicated in Fig. 4(b). Numerous “wait” states are needed to handle DRAM command latencies. In MIS-I, these states are all mapped to one no-operation (NOP) state. Before entering the NOP state, several registers have to be set by the command states (gray ones). These registers are NOP_count (the cycle count which is needed to stay in the NOP state), NOP_code (operation mode of MIS while in NOP state), and return state. In addition to wait states, data transfer states can also be mapped to NOP state and NOP_count now becomes the programmed DRAM burst length. Since MIS combines all wait and data transfer states into a NOP state and loads the command latencies or burst length into NOP_count dynamically, it is very easy to parameterize the command latencies without redesigning the FSM. If control latencies are determined and fixed before synthesis, they become hardwired logic after synthesis. These control latencies and other related SDRAM timing constraints are automatically converted from absolute timing to cycle count through built-in mathematical equations in a Verilog script file. In contrast, if there is a requirement of change in memory or clock frequency after the whole system is designed, control and status registers are allocated for these latencies to enable the ability to program them dynamically.

B. MIS-II

In MIS-I, it is clear that when the first DRAM access request is handled by the single, shared FSM, the second one has to wait until the first access has been completed. This procedure works fine for successive accesses addressed to the same bank since DRAM cannot process them at the same time. However, for those accesses addressed to different banks, SDRAM bandwidth loss is unavoidable. Because all repeated states of MIS-I are merged into a single shared NOP state, the performance of MIS-I is constrained by its poor capability of parallel processing accesses addressed to different banks. Fig. 5(a) shows how MIS-I processes two row-miss accesses addressed to different banks. It is obvious that before the command latency of

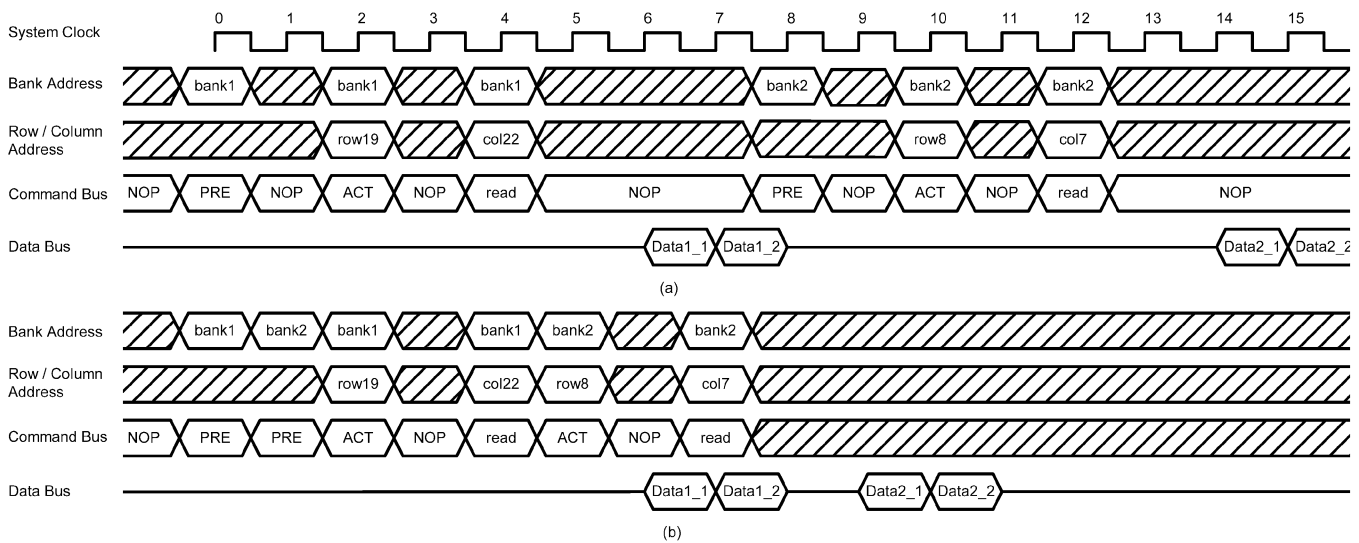


Fig. 5. Two row-miss accesses (in different banks) processed by (a) MIS-I and (b) MIS-II.

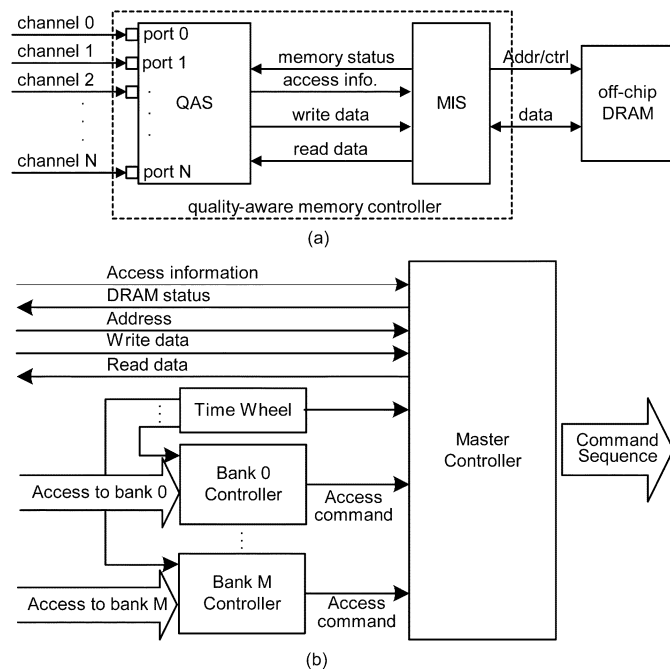


Fig. 6. (a) Quality-aware memory controller. (b) MIS-II architecture.

a registered command has been met, no any other commands can be issued. Hence, long access latencies and low bandwidth utilization are expectable. By using bank controllers and the master controller, MIS-II effectively fixes these problems [see Fig. 5(b)] and still inherits the advantages of MIS-I. As indicated in Fig. 6(b), each internal bank of SDRAM is allocated one bank controller to process accesses addressed to that bank. After accepting an access from the input port, the bank controller generates appropriate SDRAM commands according to the timing information provided by the time wheel. SDRAM commands from all bank controllers are collected by the master controller, which then issues the most proper command to SDRAM according to the information from time wheel and QAS. Master controller is also responsible for all of the other control procedures such as power-up, refresh, etc. To regulate the processing

and sequencing of all control within and among bank controllers and the master controller, the time wheel has to be carefully designed. Roughly speaking, MIS-I can be treated as MIS-II with only one bank controller.

To enhance the parallel processing of QAS, the access channels are connected as a star topology, which is widely used in practical SoC platforms, such as ARM's PrimeXsys platforms [16] and Palmchip's CoreFrame architecture [17]. Each channel has a dedicated bus connected to each port of QAS, while several PUs may share a channel. The share of a channel is basically based on the memory access characteristics of PUs, which will be illustrated in detail later.

Another mechanism that makes MIS-II more efficient than MIS-I is to separate burst transfer control from bank controllers. As mentioned earlier, data transfer states of MIS-I are merged into the shared NOP state. After all burst data have been transferred, the FSM returns to IDLE state and readies for accepting the next access. If the next access has already been pending, it still has to wait at least one clock cycle on state transition after the current access is completed. In addition, preparation cycles are sometimes needed for DRAM controllers to preprocess an access. These two types of delay result in unwanted bandwidth loss. In MIS-II design, the burst transfer control is handled by the master controller. After issuing a column-access command, the bank controller can return to IDLE state and accept the next access. The master controller will generate signals needed during the burst transfer cycles. Hence, the transition state and preparation cycles can be overlapped with burst transfer cycles and the bandwidth loss is mitigated. In brief, MIS-II is designed to respond to the DRAM access requests immediately to furthermore make the best use of SDRAM command and data bus. Therefore, SDRAM bandwidth utilization is raised whereas access latency is diminished.

C. BAG

The efficiency of on-chip communication plays an important role on system performance. For PUs having regular access behaviors, (e.g., image processing unit, audio/video codec, etc.)

the addresses of their DRAM accesses can be obtained in advance or through a simple translation. BAG is designed to generate access addresses locally for some multimedia processing units. BAG can be connected either to QAS or MIS. Without transferring address information for every DRAM access, the address bus traffic of on-chip channel can be effectively reduced. Currently, the following address generators are supported in our design.

1) *One-Dimensional (1-D) (Linear) Address Generator*: By giving the start address and access length, the 1-D address generator can automatically generate addresses for PUs whose address mapping schemes are linear, e.g., audio codec, according to the programmed DRAM burst length.

2) *Two-Dimensional (2-D) Block-Based Address Generator*: The 2-D block based address generator is designed for block based processing units such as MPEG-2 motion compensation, DCT, etc. Tile-based mapping of behavioral array references to physical memory locations is used to minimize power consumption on address bus transitions [18] and improve DRAM utilization [10].

IV. QAS

Access conflicts of shared resources are an old problem in hardware design. Mechanisms such as semaphores and scheduling are conveniently applied to eliminate these conflicts. Two common used scheduling policies are round-robin and fixed-priority scheme. Round-robin policy fairly allocates DRAM bandwidth to all channels. However, lacking priority nature makes it hard to guarantee access latency for any channel. Fixed priority policy may solve the problem of access latency. However, lower-priority channels may suffer starvation due to the high access rates of higher priority channels. Thus, it is obvious that neither of these two scheduling policies can effectively provide QoS guarantees. This problem can be especially fatal to some applications such as multimedia system designs. For example, signal processing units such as video codec may require guaranteed bandwidth, whereas CPU may concern about the access latency more when waiting for a cache line fetch. To effectively solve this problem, we propose a QAS whose scheduling policy provides not only high DRAM utilization but also QoS guarantees. In the proposed QAS design, channels are put into three categories: latency-sensitive, bandwidth-sensitive, and don't-care.

1) *Latency-Sensitive Channel*: Latency-sensitive channels are for PUs that highly concern about latencies of DRAM accesses. Accesses issued through latency-sensitive channels are called latency-sensitive accesses. Normally latency-sensitive accesses will be granted with the highest priority. Even though in this case, access latencies may still be long due to DRAM's status. For example, if a latency-sensitive access is addressed to a DRAM bank that is currently busy in serving another access, it won't be granted until the bank returns to the standby status when using DRAM controllers with conventional schedulers, such as MemMax [11]. Even when the addressed bank is at standby status, the command or the data bus of DRAM may be occupied by other accesses having been granted to access to other banks. The situation is more severe when DRAM is set

at long burst mode. In order to reduce those latencies caused by the aforementioned conditions, QAS provides two services to shorten the DRAM access latencies of latency-sensitive accesses: preemptive and column-access-inhibition (CAI) services.

- **Preemptive service**

Preemptive service is used to issue latency-sensitive accesses as soon as possible by suspending the processed access from a bandwidth-sensitive or don't-care channel. This indicates that preemptive service may reduce the average bandwidth utilization. The problem is worse when the previous access has already activated a distinct row in the same bank. Therefore, preemptive service for latency-sensitive channels is only guaranteed within an allocated DRAM bandwidth. Access requests beyond this allocated bandwidth will change the channel to don't-care type. System designers should carefully assign the allocated bandwidth to prevent serious degradation of the overall DRAM performance. Naturally, latency-sensitive accesses that are being processed will be protected and won't be suspended by other latency-sensitive ones.

- **Column-access-inhibition service**

CAI service is used to preserve the data bus for latency-sensitive accesses by inhibiting issuing column-access commands from bandwidth-sensitive and don't-care channels, and therefore eliminates latencies resulted from data bus congestion. Again, the overall bandwidth utilization is diminished when CAI service is applied since the data bus is not optimally utilized. Hence, CAI service for each latency-sensitive channel is also only guaranteed within an allocated DRAM bandwidth. Access requests beyond this bandwidth will change the channel to don't-care type.

Besides the above two services, the bank controller which is processing latency-sensitive accesses also has the highest priority to use DRAM command bus to avoid possible latencies caused by waiting for the command bus. From the above discussion, one can see that optimizing the access latencies for some particular channels is often harmful to the overall bandwidth utilization. The trade-off between high bandwidth utilization and short access latencies for some particular channels should be thoroughly considered by the system designers.

2) *Bandwidth-Sensitive Channel*: Bandwidth-sensitive channels are for PUs that concern only about bandwidth. Since accesses through this type of channels are insensitive to latency, they are scheduled by DRAM status (i.e., access status and Read/Write turnaround) to achieve the highest bandwidth utilization. For example, a row-hit access will have higher priority than a row-miss access. If two accesses have the same bank status, QAS will grant them according to round-robin scheduling policy that can fairly favor all accesses in different channels. QAS also gives those accesses having the same access direction (read or write) as the previously granted one the higher priority to prevent bandwidth loss due to SDRAM data bus turnaround cycles. Accesses through bandwidth-sensitive channels may be suspended by preemptive service to shorten

access latency of latency-sensitive channels. The access requests within the allocated bandwidth will be fulfilled by QAS to provide guaranteed bandwidth for each channel. Access requests beyond the allocated bandwidth, however, will change the channel to don't-care type.

3) *Don't-Care Channel*: Don't-care channels are for PUs that care about neither latency nor bandwidth. Accesses through this type of channels won't be guaranteed any bandwidth or latency. They are processed only when extra bandwidth left over by the first two types of channels. The same as bandwidth-sensitive channels, accesses through don't-care channels are scheduled by DRAM status and may be suspended by preemptive service.

The bandwidth allocation for latency-sensitive and bandwidth-sensitive channels is on the basis of service cycle. Service cycles are cycles in which data are transferred to/from DRAM. As shown in Fig. 7, a service period is a union of N service cycles. System integrators can configure the number of service cycles for different channel types in one service period by using the scheme originally proposed for bus management in [19]. By assigning service cycles for latency- and bandwidth-sensitive channels, users can have fine-grained control over the bandwidth allocated to these types of PUs.

Fig. 8 shows the pseudocode of the proposed quality-aware scheduling. At the beginning of a service period, all channel services mentioned above are enabled. These services won't be disabled until running out of the allocated bandwidths (service cycles actually) in a service period. At each cycle, QAS checks requests from all channels C . Requests from latency-sensitive channels C_{LS_list} are served first. When there are multiple requests from C_{LS_list} , they are scheduled by round-robin scheduling policy to decide the final unique request C_{winner} . If there is already a latency-sensitive access which is being served, C_{winner} stays in pending status. Otherwise, C_{winner} is served with preemptive and CAI services. When there is no request from C_{LS_list} , requests from bandwidth-sensitive channels C_{BS_list} are served. When there are multiple requests from C_{BS_list} , they are scheduled by DRAM status. Round-robin scheduling policy is then applied to those requests with the same DRAM status to decide the final unique request C_{winner} . C_{winner} from C_{BS_list} is, however, served without preemptive and CAI services. Finally, requests from don't-care channels are accomplished only when no requests from C_{LS_list} and C_{BS_list} , and the final unique request C_{winner} of these requests is decided in a similar manner for C_{BS_list} .

V. EXPERIMENT RESULT

In this section, we present the experimental framework used to evaluate several SDRAM controllers. We will describe the system test-bed and the use of each component in this test-bed. First, several constrained random experiments are conducted under different configuration parameters to measure the average SDRAM bandwidth utilization and the access latencies for the highest priority access channels or latency-sensitive channels of the considered SDRAM controllers. Then, a simplified STB SoC is simulated to examine the QoS performance of each SDRAM controller.

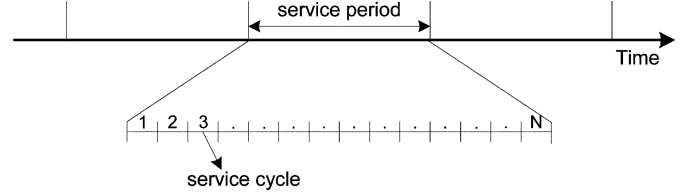


Fig. 7. Definition of service cycle and service period.

```

check_channel_request(C);

if (LS_channel_assert) { //latency sensitive channel
  for (Ci in CLS_list)
    if (check_allocated_BW(Ci) == NULL)
      change_channel_type(donot_care, Ci);
  Cwinner = round_robin(CLS_list);
  if (preemptive_service_running or CAI_service_running)
    MIS_service(pending, Cwinner);
  else
    MIS_service(preemptive, CAI, Cwinner);
}
else if (BS_channel_assert) { //bandwidth-sensitive channel
  for (Ci in CBS_list)
    if (check_allocated_BW(Ci) == NULL)
      change_channel_type(donot_care, Ci);
  sorted_by_DRAM_status(CBS_list);
  Cwinner = round_robin(CBS_list);
  MIS_service(normal, Cwinner);
}
else if (donot_care_channel_assert) {
  sorted_by_DRAM_status(Cdonot_care_list);
  Cwinner = round_robin(Cdonot_care_list);
  MIS_service(donot_care, Cwinner);
}

if (end_of_a_service_period)
  reset_all_channel_setting();

```

Fig. 8. QAS.

A. Experimental Setup

Fig. 9 shows the test-bed used to evaluate the considered SDRAM controllers. Each access initiator generates different SDRAM access behaviors for each channel connected to the SDRAM controller according to three control parameters: process_period indicates how many clock cycles an access initiator needs to process the read data; access_num indicates how many accesses an initiator issues every process_period; access_behaviors specifies different access patterns: constrained random, 1-D, 2-D block base, 2-D interlace, and 2-D block base with unpredictable start address. The simulation coordinator is responsible for generating all control signals needed in the experiments and dumping the simulation results for further analysis. Two types of on-chip bus (OCB) models are used in the experiments: a single shared bus and multiple dedicated buses. The single shared bus model is used to connect a single-channel SDRAM controller and the access initiators. In this model, a bus arbiter is included to grant accesses from access initiators with one clock cycle bus handover. The applied arbitration policies are round-robin and fixed-priority. Multiple dedicated buses are used to connect a multichannel SDRAM controller and the access initiators. Each channel can be allocated to one or more access initiators. Seven different

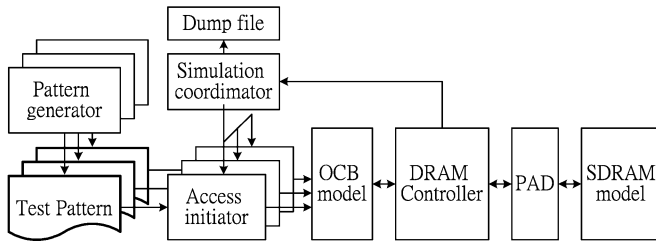


Fig. 9. Test-bed for SDRAM controller performance evaluation.

TABLE I
SDRAM CONTROLLERS USED IN THE EXPERIMENTS

Controller	Descriptions
SIG-RR-MIS-I	Single-channel MIS-I controller, the OCB arbiter uses round-robin arbitration policy
SIG-FP-MIS-I	Single-channel MIS-I controller, the OCB arbiter uses fixed-priority arbitration policy
SIG-RR-MIS-II	Single-channel MIS-II controller, the OCB arbiter uses round-robin arbitration policy
SIG-FP-MIS-II	Single-channel MIS-II controller, the OCB arbiter uses fixed-priority arbitration policy
MUL-RR-MIS-II	Multi-channel MIS-II controller with round-robin DRAM scheduler
MUL-FP-MIS-II	Multi-channel MIS-II controller with fixed-priority DRAM scheduler
QA-MIS-II	Multi-channel MIS-II controller with quality-aware DRAM scheduler

TABLE II
KEY PARAMETERS OF SDRAM MODEL

Parameters	Values	Parameters	Values
Clock rate	100 MHz	t_{RCD}	2 cycles
Data bus width	16 bit	t_{RP}	2 cycles
Num. of bank	4	t_{RAS}	5 cycles
Burst length	1, 2, 4, 8	CAS latency	2 cycles

SDRAM controllers listed in Table I will be compared in the following experiments. The SDRAM model used in this experiment is Micron's mt48lc8m16a2 SDR-SDRAM [20]. Some key parameters of this SDRAM are listed in Table II, whereas the complete specification and Verilog simulation model can be found in [20].

B. Performance Evaluation of Constrained Random Access Streams

Performance evaluation of constrained random access streams looks at the performance variations of SDRAM controllers when some control parameters are changed. The default control parameters are listed in Table III, whereas some of them may vary in different experiments. In each experiment, the bandwidth utilization provided by each SDRAM controller will be observed. Besides, to measure the shortest access latency (denoted as *min_latency*) provided by each SDRAM controller, we assume that access from *initiator 0* will be granted with the highest priority in fixed-priority controllers. For the same reason, we set *initiator 0* as latency-sensitive channel in quality-aware controllers. For round-robin controllers, no particular settings are needed due to the fair scheduling.

1) *Effect of Available Bank*: For the SDRAM containing multiple banks, the capability of SDRAM controllers to handle parallel bank access can severely affect the DRAM bandwidth

TABLE III
CONTROL PARAMETERS IN CONSTRAINED RANDOM EXPERIMENTS

Parameters	Values
No. of access initiators	7
No. of bank used	4
Burst length	4
Process period	60
access_no	3
Bandwidth requested by each initiator	32.4 MB/s

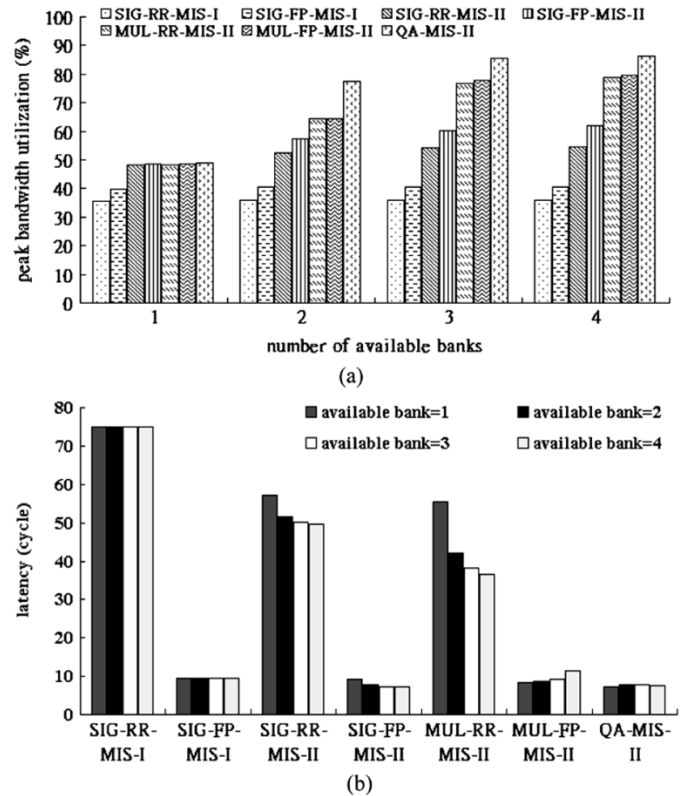


Fig. 10. Effects of number of available bank on (a) bandwidth utilization and (b) min_latency.

utilization. Fig. 10(a) illustrates the bandwidth provided by each SDRAM controller when the number of available banks varies from one to four. Some key points are listed as followed.

- For single-channel MIS-I controllers, increasing the DRAM banks obviously contributes nothing to bandwidth utilization. It is mainly because that MIS-I does not support parallel bank access. Furthermore, FP-MIS-I controller has better performance than RR-MIS-I controller has. This is evident since fixed-priority policy allows high-priority access initiators to occupy the MIS-I longer and, hence, avoids wasted cycles due to frequent bus handover.
- For one-bank SDRAM, since every SDRAM access cannot be issued until the previous one is completed, bus handover cycles shorter than the burst length have totally no influence on the bandwidth utilization. Hence, all MIS-II controllers provide the same bandwidth. However, the hidden bus handover and scheduling cycles can still make MIS-II controllers provide better bandwidth utilization than MIS-I controllers do.

- The bandwidth utilization of MIS-II controllers increases when the number of available banks increases. Compared to single-channel MIS-II controllers, multichannel controllers have more improvement on bandwidth due to the support of high-efficient parallel bank access.
- Among all controllers, QA-MIS-II provides the highest bandwidth utilization. This is mainly because high-efficient scheduling for bandwidth-sensitive accesses.

The effect of bank number on min_latency of each SDRAM controller is shown in Fig. 10(b). We make the following observations from this figure.

- min_latency of single-channel MIS-I controllers are insensitive to the number of available bank since no parallel bank access is supported. In contrast, min_latency of single-channel MIS-II controllers are reduced when the number of available banks is increased.
- min_latency of round-robin controllers are longer compared to fixed-priority controllers. This is because round-robin controllers fairly grant accesses from each channel. Therefore, accesses from *initiator 0* have to wait for its round for a long time since the number of access initiators in this experiment is seven. The performance variation due to the number of access initiators will be discussed later.
- For the MUL-FP-MIS-II controller, increasing the available banks makes the access latencies longer. This is because multichannel controllers allow accesses addressed to a free bank to be processed as soon as possible to optimally utilize the DRAM bandwidth. Hence, low-priority accesses and high-priority ones may be processed at the same time. This condition unavoidably makes the command and data bus congestion more serious, which in turn leads to longer min_latency. This problem is effectively eliminated by the preemptive and CAI services used in QA-MIS-II controller.

2) *Effect of Burst Length*: Generally speaking, increasing the burst length comes with higher bandwidth utilization since each column-access command can transfer more data. Therefore bandwidth loss due to bus handover and turnaround is reduced. Fig. 11(a) depicts the bandwidth utilizations when the burst length varies from one to eight. As we can see, single-channel MIS-II controllers provide almost the same bandwidth utilization as the multichannel controllers can when the burst length was programmed to eight. It is reasonable since the burst transfer cycle is long enough for single-channel controllers to hide all bus handover cycles and command latencies. Again, the quality-aware controller provides the highest bandwidth utilization due to the high-efficient scheduling for bandwidth-sensitive accesses.

Fig. 11(b) illustrates min_latency of *initiator 0*. For single-channel controllers, increasing burst length results in longer access latency since the time to wait the completion of the previous accesses is longer. As for multichannel controllers, increasing the burst length can be taken as increasing the data bus congestion. Hence, the access latencies is also longer. On the other hand, min_latency of QA-MIS-II is reduced by 46%

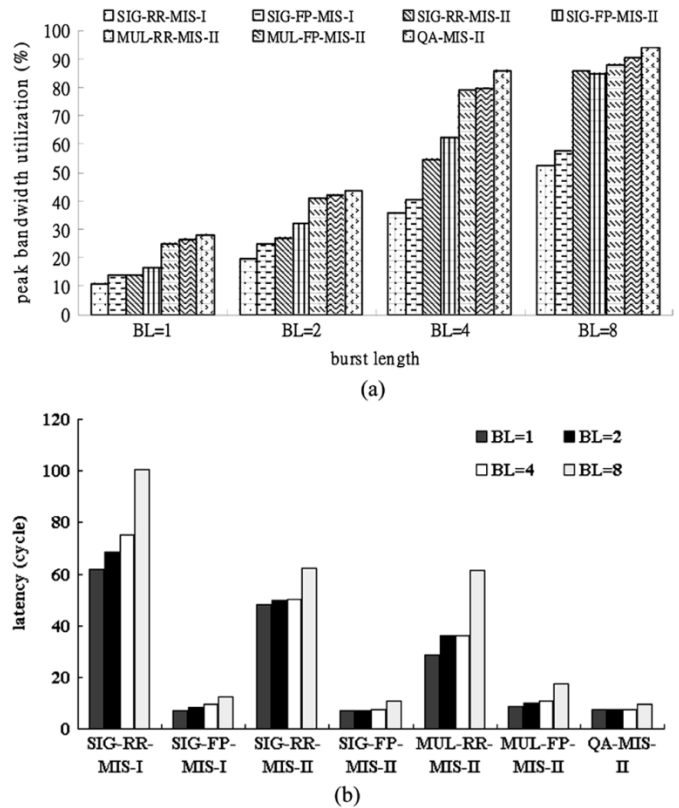


Fig. 11. Effects of burst length on (a) bandwidth utilization and (b) min_latency.

and 30% compared to MUL-FP-MIS-II when the burst length is programmed to eight and four, respectively. These improvements are due to that SDRAM data bus congestion problem happened to multichannel controllers are again eliminated by the preemptive and CAI services.

3) *Effect of Number of Access Initiators*: As mentioned earlier, single-channel MIS-II controllers can only support limited capacity for parallel bank access since only one access is visible to the controller at each clock cycle. In contrast, multichannel controllers do not have this problem since these controllers can see accesses from every channel at the same time. In the following discussion, we examine the performance variation of each SDRAM controller when the number of access initiators in the system increases from one to seven. Note that the SDRAM bandwidth requirement for each initiator is fixed to 32.4 MB/s.

Fig. 12(a) and (b) show the effects of the number of access initiators on bandwidth utilization and min_latency, respectively. When the number of initiators is less than two, the total bandwidth requirement can be fulfilled by all controllers. The performance difference starts to be obvious when more than three initiators are concurrently issuing DRAM accesses. Single-channel MIS-I and MIS-II controllers encounter their performance bound when the number of initiators is three and four, respectively. In contrast, multichannel controllers still have some available bandwidth for initiators more than four and the bandwidth utilization even goes higher with more access initiators. Fig. 12(b) shows min_latency of each SDRAM controller. It is clear that when the number of the initiators increases, the access latencies become longer. The congestion

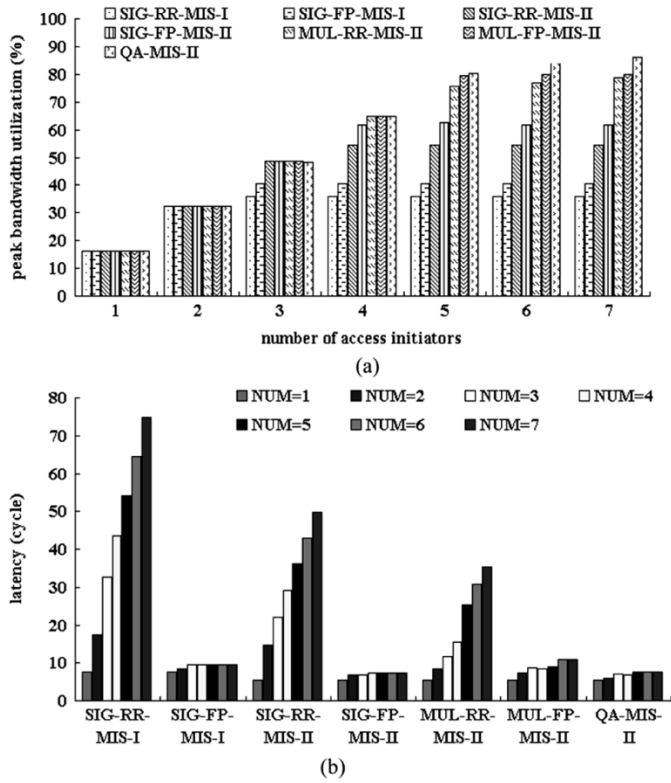


Fig. 12. Effects of channel number on (a) bandwidth utilization and (b) min_latency.

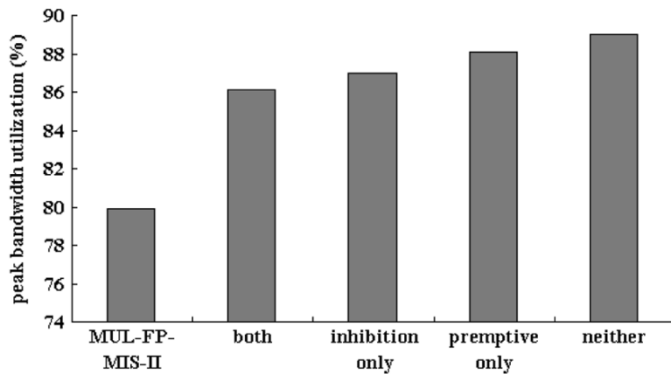


Fig. 13. Bandwidth comparison between MUL-FP-MIS-II and QA-MIS-II with different services.

problem is again solved by the preemptive and CAI services in QA-MIS-II. Compared to MUL-FP-MIS-II, min_latency of QA-MIS-II can be reduced up to 30%.

4) *Effect of Preemptive and CAI Service:* In this subsection, we take a look at how preemptive and CAI services affect the overall system performance. Fig. 13 compares the MUL-FP-MIS-II and the QA-MIS-II with different services. Compared to MUL-FP-MIS-II, any configuration of QA-MIS-II can provide higher bandwidth utilization due to the high-efficient scheduling for accesses from bandwidth-sensitive channels. When both the preemptive and CAI services are enabled, the bandwidth degradation is most severe compared to other configurations. The second severe degradation of bandwidth occurs when only CAI service is enabled since this service preserves the DRAM data

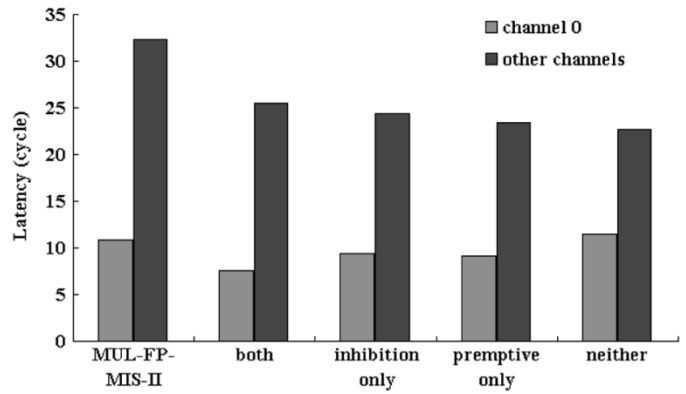


Fig. 14. Latency comparison between MUL-FP-MIS-II and QA-MIS-II with different services.

bus for latency-sensitive channels to prevent data bus congestion problem and therefore results in low bus utilization. When only preemptive service is enabled, the bandwidth degradation is minor. This is because preemptive service can make accesses from *initiator 0* be processed as soon as possible and then free QAS to grant other requests from nonlatency-sensitive channels. This higher efficient scheduling is helpful to increase bandwidth utilization and hence can neutralize the bandwidth loss resulted from preemptive service.

Fig. 14 illustrates min_latency of MUL-FP-MIS-II and QA-MIS-II with different services. It is obvious that optimizing the access latency of a particular channel is harmful to that of other channels. Note when both preemptive and CAI services are disabled, min_latency of QA-MUL-II is slightly longer than that of MUL-FP-MIS-II. This is because the higher bandwidth utilization of QA-MIS-II also makes the data bus congestion more severe. The access latency of other channels of MUL-FP-MIS-II is higher than that of QA-MIS-II because it is dominated by access latencies of low-priority channels.

5) *Summary:* The above experiments clearly indicate that multichannel SDRAM controllers can provide much higher bandwidth than single-channel controllers when high-efficient parallel bank access is support. Furthermore, these experiments also show that maximizing the bandwidth utilization can be harmful to access latency for some particular initiators because of the possible command and data bus congestions. This problem can be effectively solved by preemptive and CAI service provided by QA-MIS-II. Hence, only QA-MIS-II can successfully provide both high bandwidth utilization and short access latency services.

C. Performance Evaluation of STB Emulation Environment

In this section, we simulate several events that may occur in a digital STB chip, which is a good example of multimedia SoC design. The basic hardware components of a STB SoC may include a CPU, audio/video codec, network devices, etc. Various demands on DRAM service cause it difficult to design a memory controller that fulfills different requirements of these components. Some new features supported by digital STB, such as the interactive television (ITV) service, make the situation even worse due to the drastic change in the requirement of DRAM bandwidth. ITV service allows users to not just sit in front of the

TABLE IV
CONFIGURATION OF PUs IN THE STB SYSTEM

PU	Fixed-priority	Quality-aware
CPU	1 st priority	Latency-sensitive
Transport stream	2 nd priority	Bandwidth-sensitive
DSP	3 rd priority	Bandwidth-sensitive
OSD	4 th priority	Bandwidth-sensitive
Video decoder	5 th priority	Bandwidth-sensitive
Display	6 th priority	Bandwidth-sensitive
Wireless LAN	7 th priority	Don't-care

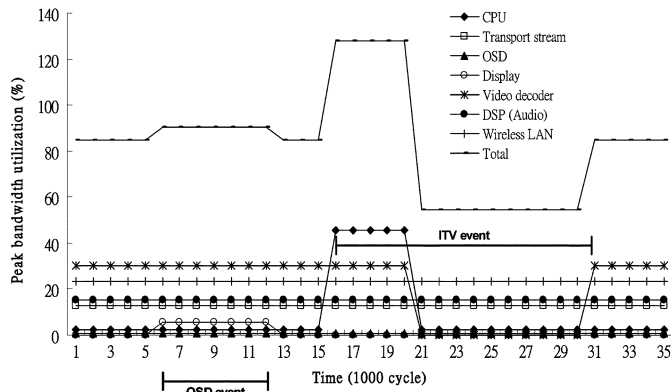


Fig. 15. Bandwidth requirement of each PU in the STB SoC.

TV but interact with the broadcasting programs, such as online betting, home shopping, etc. Therefore, the DRAM bandwidth required by CPU might vary when CPU performs different application programs.

1) *System Setup*: In this experiment, seven PUs with configuration listed in Table IV share a unified off-chip SDRAM resource. To assure low-latency access for CPU's cache line fetch, we give CPU the highest priority in fixed-priority controllers. For the same reason, CPU is taken as latency-sensitive PU in the quality-aware controller. Besides, because users are less sensitive to the efficiency of download speed, wireless LAN controller is set as the lowest priority and don't-care PU in fixed-priority and quality-aware controllers, respectively. Since round-robin controllers fairly schedule accesses from all PUs, no special configuration is needed. As shown in Fig. 15, we simulate some events that may occur when the STB operates. Most of the time the user just watches TV programs provided by the broadcasters. Therefore, the total bandwidth requirement of the STB system is rather steady. While watching the program, the user also downloads some files through wireless LAN, e.g., video clips or MP3 files. Hence, wireless LAN controller also needs SDRAM bandwidth for the downloaded data. Two events that cause variation of SDRAM bandwidth requirement are on-screen display (OSD) and ITV events. The OSD event has happened during cycle 5000–10000. It is activated by the user to setup the functionality of the STB. The total bandwidth requirement slightly increases during this OSD event. ITV event happens when the user executes some ITV applications. For example, while watching a basketball game, the user wants to browse the player files by asking the broadcaster to provide the service. After the ITV application has been activated, CPU requests a large amount of bandwidth instantly to process the application and the system bandwidth

requirement hence exceeds the peak bandwidth that could be provided by the applied DRAM. The user then paused the TV program temporarily during cycle 21000–31000 to browse the information offered by the broadcaster.

2) *Simulation Results*: To examine the QoS performance of each SDRAM controller, we look at the fulfillment of bandwidth allocation first. The latency of CPU is assessed later. For round-robin controllers, the simulation results of SIG-RR-MIS-I and SIG-RR-MIS-II are shown in Fig. 16(a) and (b), respectively. As we can see, the limited bandwidth provided by these two SDRAM controllers obviously cannot fulfill the requirement of STB SoC. Fig. 16(c) shows the simulation result of MUL-RR-MIS-II. Although the overall bandwidth utilization is much better in MUL-RR-MIS-II, the bandwidth allocation for each PU is unacceptable. As mentioned above, round-robin controllers evenly allocate total bandwidth to each PU. Thus, both OSD and ITV events may result in quality degradation of the broadcasting program. Take the ITV event for example, CPU requests a large bandwidth instantly and some bandwidth for the video decoder and the display unit is allocated to CPU. Therefore, the guaranteed bandwidth requirements of these two PUs are ruined.

As for fixed-priority controllers shown in Fig. 16(d) and (e), the single-channel fixed-priority controllers can provide higher bandwidth utilization than single-channel round-robin controllers can. This is because they allow high priority PUs to access DRAM uninterruptedly and hence can avoid bandwidth loss due to bus handover and frequent row reopening. However, the provided bandwidth is still not enough. For example, no any bandwidth is allocated to wireless LAN controller during the normal operation period. The bandwidth provided by MUL-FP-MIS-II is much higher. However, as indicated in Fig. 16(f), the bandwidth allocation problem still exists. During the ITV event, CPU takes a large portion of bandwidth. This severely degrades the quality of the broadcasting program and is therefore unacceptable.

Fig. 16(g) illustrates the simulation result of quality-aware controller. As we can see, the bandwidths allocated for all bandwidth-sensitive PUs are well guaranteed. When ITV event happens, the bandwidth for wireless LAN controller is taken first. The quality of the broadcasting program remains unchanged when the user is still watching the program. After the TV program has been paused, CPU takes the spared bandwidth released by the video decoder for the ITV program. In addition, accesses from the wireless LAN controller can be served as much as possible during the period.

As shown in Fig. 17, CPU access latency is measured separately when CPU operates during the normal operation period and the ITV event. First we take a look at the normal operation period. As expected, round-robin controllers have the longest access latencies compared to other controllers due to the fair scheduling, which apparently cannot fit CPU's low latency requirement. By granting accesses from CPU with the highest priority, fixed-priority controllers can effectively reduce the access latencies. Among these fixed-priority controllers, MUL-FP-MIS-II has the longest CPU access latency due to that its high bandwidth utilization causes serious SDRAM command and data bus congestion. The congestion problem

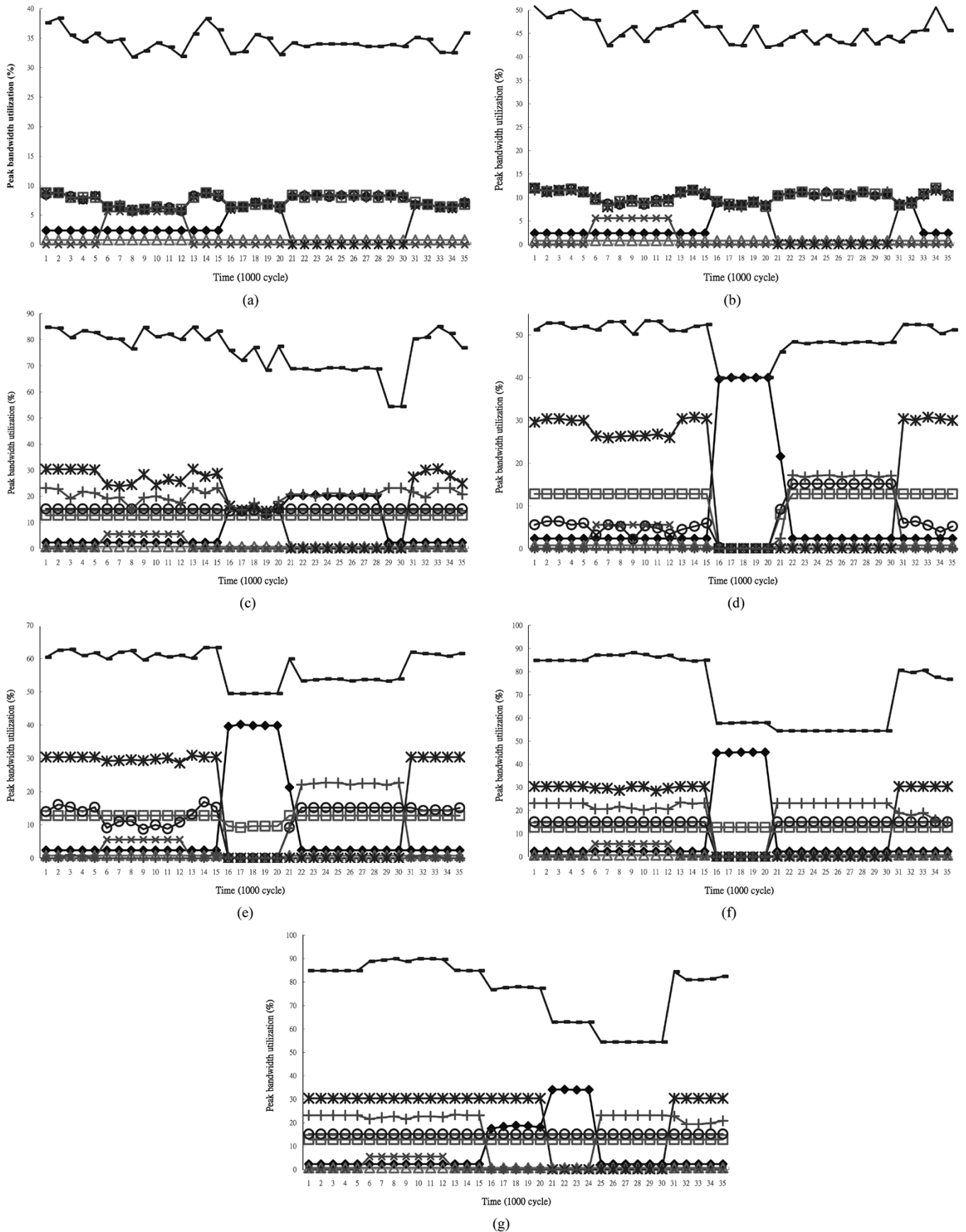


Fig. 16. Bandwidth utilization of (a) SIG-RR-MIS-I, (b) SIG-RR-MIS-II, (c) MUL-RR-MIS-II, (d) SIG-FP-MIS-I, (e) SIG-FP-MIS-II, (f) MUL-FP-MIS-II, and (g) QA-MIS-II.

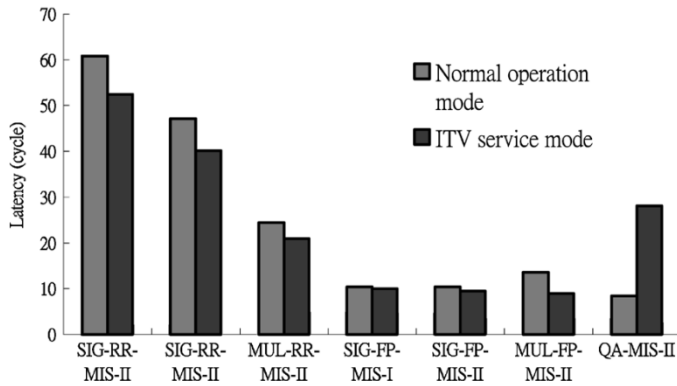


Fig. 17. CPU access latency in different operation modes of each controller



Ch 0 AB	Address buffer of CPU channel
Ch 1 AG	Address generator of transport stream
Ch 2 AG	Address generator of DSP
Ch 3 AG	Address generator of OSD
Ch 4 AG	Address generator of Video decoder
Ch 5 AG	Address generator of Display
Ch 6 AG	Address generator of wireless LAN
QAS	Quality aware scheduler
MIS	Memory interface socket II
Technology:	TSMC 0.35 μ m 1P4M CMOS
Gate count:	17,365
Die area:	1.05 \times 1.05 mm ²
Supply voltage:	3.3 V
Clock rate:	100 MHz

Fig. 18. Layout and specification of the prototyping chip.

is effectively eliminated in QA-MIS-II. With preemptive and CAI services, CPU access latency of QA-MIS-II can be further reduced by about 19% and 37% compared to SIG-FP-MIS-II and MUL-FP-MIS-II, respectively.

During the ITV event, CPU access latencies of round-robin controllers are lower than those latencies of round-robin controllers during normal operation mode, since the video decoder is paused in cycle 21000–30000. Similarly, fixed-priority controllers also have shorter CPU access latency during the ITV event. This is because the high request rate of CPU blocks other PUs to access DRAM and hence preserves most DRAM resources for CPU. In contrast, the latency of QA-MIS-II is longer

than that of fixed-priority controllers since the bandwidth requirement of the ITV applications is taken as don't-care type. This is acceptable because users are often less sensitive to the execution speed of ITV applications.

The design written in Verilog RTL is synthesized by Synopsys Design Analyzer in worst case using Avanti! cell library, TSMC 0.35- μ m 1P4M CMOS technology. The layout and the specification of the prototyping chip are shown in Fig. 18.

VI. CONCLUSION

In this paper, we have presented a multilayer, quality-aware SDRAM controller for multimedia platform SoCs. The layered architecture is motivated by the awareness of that not every system needs the same requirement of memory usage. Therefore, we well partition the functionality of a memory controller into proper layers such that designers have the flexibility to adopt the best fitting layers for various applications. By appropriately categorizing channels into three types, QAS is able to provide the best DRAM services including short access latency and guaranteed bandwidth for each type of channels. DRAM bandwidth utilization is improved by the support of parallel access of each bank within SDRAM and the ability to issue every DRAM command at the earliest time available. The configurability of MIS, based on the shared-state FSM design, can alleviate the burden for system designers by rapid integration of SDRAM subsystem. Some recently developed systems, especially those for portable applications, have power management with the ability to control the system clock frequency in adjusting system performance to just fit to the requirement. Programmability of DRAM control latencies enables the power management to dynamically lower the clock frequency of MIS.

The results of STB experiment show that the access latency of the latency-sensitive channel can be reduced by 37% and 65% compared to conventional multichannel fixed-priority and round-robin controllers, respectively. Furthermore, the memory bandwidths can be precisely allocated to bandwidth-sensitive channels with a high degree of control and no bandwidth-sensitive channel suffers starvation in all simulated STB events. In summary, the proposed memory controller for multimedia SoCs can achieve high DRAM utilization while still meets different memory access requirements of bandwidth and latency.

REFERENCES

- [1] B. Furht, "Multimedia systems: An overview," *IEEE Multimedia*, vol. 1, no. 1, pp. 47–59, Spring 1994.
- [2] F. Cathoor *et al.*, *Custom Memory Management Methodology: Exploration of Memory Organization for Embedded Multimedia System Design*. Norwell, MA: Kluwer, 1998.
- [3] P. R. Panda, N. Dutt, and A. Nicolau, *Memory Issues in Embedded Systems-on-Chip: Optimizations and Exploration*. Boston, MA: Kluwer Academic Publishers, 1999.
- [4] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*, 3rd ed. San Francisco, CA: Morgan Kaufmann, 2002.
- [5] A. Cataldo. (2001, Oct.) MPU Designers target memory to battle bottlenecks. [Online]. Available: <http://www.siliconstrategies.com/story/OEG20011019S0125>
- [6] R. C. Schumann, "Design of the 21 174 memory controller for DIGITAL personal workstations," *Dig. Tech. J.*, vol. 9, no. 2, pp. 57–70, 1997.

- [7] J. Carter *et al.*, "Impulse: Building a smarter memory controller," in *Proc. HPCA*, Jan. 1999, pp. 70–79.
- [8] S. Rixner *et al.*, "Memory access scheduling," in *Proc. ISCA*, Vancouver, BC, Canada, Jun. 2000, pp. 128–138.
- [9] K. Ayukawa, T. Watanabe, and S. Narita, "An access-sequence control scheme to enhance random-access performance of embedded DRAM's," *IEEE J. Solid-State Circuits*, vol. 33, no. 5, pp. 800–806, May 1998.
- [10] T. Takizawa and M. Hirasawa, "An efficient memory arbitration algorithm for a single chip MPEG2 AV decoder," *IEEE Trans. Cons. Electron.*, vol. 47, no. 3, pp. 660–665, Aug. 2001.
- [11] Efficient shared dram subsystems for SOCs [Online]. Available: http://www.sonicsinc.com/sonics/products/memmax/productinfo/docs/DRAM_Scheduler.pdf
- [12] SOCCreator guide design flow [Online]. Available: <http://www.socworks.com/socworks/support/documentation/html/SOCCreator-Guide-Design-Flow.html>
- [13] K. Lahiri, A. Raghunathan, and G. Lakshminarayana, "LOTTERYBUS: A new high-performance communication architecture for system-on-chip designs," in *Proc. Design Automation Conf.*, Jun. 2001, pp. 15–20.
- [14] F. J. Harmsze, A. H. Timmer, and J. L. van Meerbergen, "Memory arbitration and cache management in stream-based systems," in *Proc. DATE*, Paris, France, Mar. 2000, pp. 257–262.
- [15] K.-B. Lee and C.-W. Jen, "Design and verification for configurable memory controller—Memory interface socket soft IP," *J. Chin. Inst. Elect. Eng.*, vol. 8, no. 4, pp. 309–323, 2001.
- [16] PrimeXsys platforms [Online]. Available: <http://www.arm.com/armtech/PrimeXsys?OpenDocument>
- [17] B. Cordan, "An efficient bus architecture for system-on-chip design," *Proc. IEEE Custom Integrated Circuits*, pp. 623–626, May 1999.
- [18] P. R. Panda and N. D. Dutt, "Low-power memory mapping through reducing address bus activity," *IEEE Trans. Very Large Scale (VLSI) Integr. Syst.*, vol. 7, pp. 309–320, Sep. 1999.
- [19] S. Hosseini-Khayat and A. D. Bovopoulos, "A simple and efficient bus management scheme that supports continuous streams," *ACM Trans. Comput. Syst.*, vol. 13, no. 2, pp. 122–140, 1995.
- [20] Micron Technology, Inc. mt48lc16m16a2 256Mb SDRAM (2003, Jan.) [Online]. Available: <http://www.micron.com/products/datasheet.jsp?Path=/DRAM/SDRAM&fileID=10>



Kun-Bin Lee (S'02–M'05) received the B.S. degree in electrical engineering from National Sun Yat-Sen University, Taiwan, R.O.C., in 1996, and the M.S. and Ph.D. degrees in electronics engineering from National Chiao-Tung University, Hsinchu, Taiwan, R.O.C., in 1998 and 2003, respectively.

He is currently with MediaTek, Inc., Hsinchu, Taiwan, R.O.C. His current research interests include processor architecture, digital signal processing, and system-level exploration with focus on data transfer optimization and memory management for image

and video applications.

Dr. Lee is a member of Phi Tau Phi.



Tzu-Chieh Lin received the B.S. and M.S. degrees in electronics engineering from National Chiao-Tung University, Hsinchu, Taiwan, R.O.C., in 2001 and 2003, respectively.

He is currently with MediaTek, Inc., Hsinchu. His research interests include VLSI design, digital IC design, and memory management for multimedia platform SoC.



Chein-Wei Jen (S'78–M'84) received the B.S. and Ph.D. degrees from National Chiao-Tung University, Hsinchu, Taiwan, R.O.C., in 1970 and 1983, respectively, and the M.S. degree from Stanford University, Stanford, CA, in 1977.

He is currently with the Department of Electronics Engineering and Institute of Electronics, National Chiao-Tung University, Hsinchu, Taiwan, R.O.C., as a Professor. During 1985–1986, he was with the University of Southern California, Los Angeles, as a Visiting Researcher. His current research interests

include VLSI design, digital signal processing, processor architecture, and design automation.

Dr. Jen is a member of Phi Tau Phi.