A Cost-Effective Latency-Aware Memory Bus for Symmetric Multiprocessor Systems

Jongsun Kim, Member, IEEE, Bo-Cheng Lai, Member, IEEE, Mau-Chung Frank Chang, Fellow, IEEE, and Ingrid Verbauwhede, Senior Member, IEEE

Abstract—This paper presents how a multicore system can benefit from the use of a latency-aware memory bus capable of dual-concurrent data transfers on a single wire line. Source synchronous CDMA interconnect (SSCDMA-I) has been adopted to implement the memory bus of a shared-memory multicore system. Two types of bus-based homogeneous and heterogeneous multicore systems are modeled and simulated by a cycle-accurate simulation platform. Unlike the conventional time-division multiplexing (TDM) bus-based multicore system that shows degradation in performance as the number of processing cores increases, the proposed SSCDMA bus-based multicore shows higher performance up to 23.1 percent for four cores. The maximum latency of a heterogeneous multicore system with a mix of traffic loads has been reduced up to 78 percent. These results demonstrate that the performance of multicore systems can be improved with less cost and network complexity by reducing the bus contention interferences and by supporting higher concurrency in memory accesses that brings shorter critical word access latency.

Index Terms—Memory bus, I/O interconnect, concurrency, memory latency, multicore processor, SMP, CMP, DRAM.

1 INTRODUCTION

Most digital systems consist of interconnection networks to transfer and receive data between subcomponents of a system. Depending on the scale of a system, the network is chip, package, board, or system level. The interconnection network must be programmable to change the connections between subcomponents in time, and therefore, it is composed of physical interconnect channels (such as wires or cables), switches, routers, bridges, as well as logical controllers to control data flows. As systems become more complex, the interconnection between subcomponents is becoming increasingly difficult. Therefore, communication-aware design approach is essential in cost-effective high-performance digital systems today. To design an efficient interconnection network, the system’s bandwidth and latency sensitivities should be taken into account together. This means that today’s system design approach should be latency aware as well as bandwidth aware [1].

Multicore processor systems such as a symmetric multiprocessor (SMP) system consisting of two or more chip-level multiprocessors (CMP) cores are now popular in servers, workstations, and personal computing market today. In multicore systems based on traditional bus architectures, two or more identical CPU cores are connected to a shared DRAM-based main memory via a memory bus on a printed circuit board. Unfortunately, as the number of CPU cores increases, the performance of many applications could be degraded due to the limited bandwidth or excessive latency of the interconnection network. The reason is that only one CPU core performs data transaction with a shared memory at a time, and other CPUs must wait to eliminate electrical contention until the bus access is granted. This communication resource arbitration or scheduling is done by an arbitrator or a memory controller. If multiple CPU cores attempt to access the shared memory bus simultaneously, the access is controlled in a serial way by the progression of time to avoid bus contention. Therefore, as the number of CPUs increases, the limited capability for real-time memory bus allocation increases response time (or request latency) to shared memory access and therefore becomes the important performance bottleneck factor that was not considered seriously in previous researches.

In traditional multicore systems, the available number of I/O pins and the available bandwidth (or data rate) of a memory bus are usually limited due to many practical reasons. Instead of using a bus-based interconnection network, using dedicated lines (such as point-to-point serial links) for each CPU core-to-memory interconnection may also support off-chip bandwidth and simultaneous multiple data transactions. However, as the number of processor cores increases, this approach increases the cost and the complexity of the interconnection network seriously; especially communication between processors becomes more difficult and costly. Although point-to-point serial links based on packet switching [10], which are most popular in high-speed off-chip communications nowadays, may be suitable for use in bandwidth-sensitive applications, these may not be efficient for use in latency-sensitive embedded system applications due to the large packet overhead and network latency overhead [1].

Therefore, in case of latency-sensitive multicore systems based on traditional bus or link architectures, the conventional interconnection network design approach focused on bandwidth alone is not efficient and cost-effective. To solve this problem, a new type of interconnect technology named source-synchronous CDMA interconnect (SSCDMA-I), which focuses on both latency and concurrency in communication, was introduced in our previous work [1], [2], [7]: A single SSCDMA-I line enables dual-concurrent data transactions on a shared interconnect. This is explained more in detail in Section 2.

In this paper, the SSCDMA-I is used to implement the memory bus (based on either a shared bus or serial link) of multicore systems (shown in Figs. 3 and 8) consisting of multiple processing cores. The motivation of this paper is to explore how the standard memory bus could be modified for higher system performance without increasing cost and complexity. Unlike the previous work [1], [2] focused on circuity- and architecture-level hardware design, this paper demonstrates how a latency-aware multicore system can benefit from the use of the novel interconnect in system level. A data-flow JPEG encoder and a mix of heterogeneous traffic loads are simulated by using the GEZEL [4] cosimulation kernel. Compared to the conventional TDM bus-based homogeneous multicore system, the overall system performance of the SSCDMA bus-based system has been increased by up to 23.1 percent for four cores by reducing the bus contention and by utilizing the reduced read request latencies. In a heterogeneous multicore system, it is shown that the SSCDMA-I is able to support different traffic demands easily and therefore reduce the maximum latency up to 78 percent.

This paper is organized as follows: Section 2 discusses the interconnect-related latency and concurrency issues of a multicore system design. Section 2 also introduces SSCDMA interconnect and compares its transaction operation with a conventional TDM interconnect. Section 3 describes the cycle-accurate system-level simulation results to explore how to exploit parallelism in communication by comparing the TDM and SSCDMA bus-based homogeneous multicore systems. Section 4 describes a different application example that shows how a heterogeneous multicore system can benefit from the use of SSCDMA-I. Section 5 draws conclusion.
2 DESIGN OF A COST-EFFECTIVE AND LATENCY-AWARE
SHARED-MEMORY MULTICORE SYSTEM

In many applications, the traditional bandwidth-focused approach is not enough to improve overall system performance [6]. For example, in the effective bandwidth versus peak bandwidth, the 200-MHz SDRAM and the 100-MHz DDR SDRAM both have the same peak memory bus bandwidth of 200 Mbps/pin. Although the double data rate (DDR) bus supports double bus peak bandwidth by carrying data on both the rising and falling edges of the clock, the 200-MHz SDRAM has higher effective bandwidth than the 100-MHz DDR, since the 200-MHz SDRAM’s initial read latency is shorter than that of the 100-MHz DDR [8]. Therefore, to design cost-effective and high-performance shared-memory multicore systems based on bus or link I/Os for short-range onboard communications, a latency-aware system design methodology should be followed by considering a system-level point of view: Less circuit complexity as well as smaller number of pins and PCB traces is desired; lower latency, higher concurrency, higher bus utilization, and proper bandwidth are design targets.

2.1 Interconnect Technology for Dual-Concurrent Transactions on a Shared Bus or Link

Among the various conventional interconnection networks, multi-point shared bus (shown in Fig. 1a) is the most simple and cost-effective solution for small-scale shared-memory multiprocessor systems. Similarly, the multicore CPUs can be connected to the memory by using a serial link (two-point to two-point in this example), as shown in Fig. 1b. The communication between the masters and slaves of these bus-based and link-based systems is traditionally based on a time-division multiplexing (TDM) protocol [9] (as shown in Fig. 1c upper side) that support split transactions: after cpu0’s access to m0 for data a0 to a3 through a transmission line, cpu1 is able to access m1 for b0 to b3 through the same line. TDM is often called time-interleaving or time-division multiple access (TDMA). Generally, TDM can be divided into three classes of arbitration protocols: first, static order scheduling; second (fixed or dynamic) time-driven scheduling; and third (static or dynamic) priority-driven scheduling. Thus, the conventional interconnection using the TDM protocol is called TDM-I in this paper. TDM-I is usually based on two-level (or 2-PAM) binary signaling that utilizes one data eye to send 1 bit of data at a time. 2-PAM TDM-I is simple but not efficient in dealing with multiple data transactions simultaneously. The TDM-I-based buses or links may have long request latency for back-to-back or concurrent requests due to the bus contention and arbitration delays [1].

On the other hand, as shown in Fig. 1c (lower, case 1), a single SSCDMA-I wire is capable of handling dual data transactions simultaneously by using three-level (or 3-PAM) signaling. This means that a single SSCDMA-I is able to operate as if it consists of dual virtual TDM-I channels: Each SSCDMA channel is based on the TDM protocol but operating individually without interference. Fig. 1c illustrates how the communication concurrency can be increased twice (without increasing the number of wires) by adding one more level to the conventional two-level binary signaling interconnect. For instance, while cpu0 is reading data from m0 through a single transmission line, cpu1 can access m1 and read data through the same line at the same time. Two 2-level data eyes are used to send 1-bit data due to the use of orthogonal CDMA coding. The use of source synchronous multilevel superposition technique enables that the two 2-level data streams are superposed and then become three-level signals on the transmission line but are separated and isolated from each other [1].

Both the 2-PAM TDM-I and the 3-PAM SSCDMA-I have the same data rate per wire line at the same signaling frequency. If simultaneous multiple access is not required, SSCDMA is able to allocate the channel and the bandwidth just like TDM, as shown in Fig. 1c (case 2). This will be further discussed in Section 3.2. The SSCDMA bus is simple and inexpensive to implement hardware configuration. The circuit details, signaling comparison, and the feasibility of this new interconnect technology are explained in [1]. Based on the assumption that both TDM and SSCDMA utilize the same type of current mode driver for the same data rate, the signaling power dissipations of TDM and SSCDMA for concurrent N-bit bus transactions are $N^2 V_s^2 I$ and $N/2 V_s^2 I$, respectively, where $V_s$ is the maximum channel swing and $I$ is the driver current. Although it looks like SSCDMA dissipates only 50 percent of TDM, there will be no difference in signaling power consumption if we maintain the same voltage noise margin by either increasing the swing of SSCDMA twice or reducing the swing of TDM to $V_s/2$.

2.2 Comparison of Latency and Concurrency in Memory Accesses

Fig. 2 describes the memory access timing diagram in Fig. 1, comparing the read request latencies of the TDM-I and SSCDMA-I
memory buses. We assume that both the 8-bit-wide memory buses have the same aggregate data rate per wire. Each CPU attempts to access different memory at the same time (i.e., CPU0-M0 and CPU1-M1). R0 and R1 are row activate commands, and C0 and C1 are read commands. In the TDM-I memory bus shown in Fig. 2a, the critical word latency for the second request (data1) is given as follows:

\[
t_{\text{Critical}} (\text{TDM-I}) = t_{\text{RAC}} + 16 \text{ cycles for data0} + 4 \text{ cycles} = 28 \text{ cycles.} \tag{1}
\]

We assume packet protocol with a minimum packet size of four cycles. The first four-cycle data of each data packet (= 16 cycles) transaction is assumed to be a critical word (\(t_{\text{RAC}} = \text{activate command to data out latency} = 1\text{RC} + CL, CL \text{ is CAS latency}).

On the other hand, in the SSCDMA-I memory bus shown in Fig. 2b, the critical word latency for the second request (data1) is dramatically reduced to the following equation:

\[
t_{\text{Critical}} (\text{SSCDMA-I}) = t_{\text{RAC}} + 8 \text{ cycles} = 16 \text{ cycles}. \tag{2}
\]

The delay time for transferring data0 has been eliminated in SSCDMA-I. This is essential since the CPU core usually requires only one word (e.g., 8 bytes) of the full cache line block at a time. This result shows that the SSCDMA-I bus can remove the bus contention problem of a conventional TDM-I bus. This unique multiple access capability of SSCDMA-I makes it possible to achieve higher concurrency and lower request latency in memory access applications for embedded multiprocessor systems with multiple homogeneous or heterogeneous CPU cores.

### 3 Performance Evaluation of a Homogeneous Multicore System

Fig. 3a shows the small-scale (one to four processing cores) bus-based shared-memory multicore system, which is used to demonstrate the benefits of SSCDMA-I. Similarly, a link-based shared-memory multicore system can also be implemented, as shown in Fig. 3b. In this paper, we focus on our analysis for the bus-based system, but the same approach can be applied to the link-based system to achieve the same benefits [7]. In Fig. 3a, the CMP microprocessor (MPU) consists of four ARM core processors (CPU0-CPU3) that can be implemented on a single chip or a single package containing multiple dies. Each ARM (five-stage pipelined StrongARM microarchitecture) has its own private Level 1 (L1) cache and I/O. The cache consists of 2,000-byte data cache and instruction cache with a cache line block size of 16 bytes. This embedded SMP system is modeled and simulated by a cycle-accurate simulation platform described in [3]. It is based on the GEZEL [4] cosimulation kernel. In [3], a thread-based programming model is used: a single-thread application program is partitioned into multiple threads, and then, each thread is executed by each core concurrently. This homogeneous multicore system supports MSI-like cache coherence protocol (to guarantee data integrity when data are shared and cached within each core) and write-back policy. The bus arbitration is based on the priorities to each processing core. Every core has the same priority, and therefore, the off-chip shared-memory access is based on a first-come-first-serve basis. For a heterogeneous multicore system, which will be discussed in Section 4, the priorities are given based on the latency sensitivity of a traffic load. A hardware test-and-set lock is used to support communication between cores, synchronization, and memory bus access [3].

Two types of shared memory SMP system based on either the 3-PAM SSCDMA-I or the conventional 2-PAM TDM-I will be compared. We assume the memory consists of ideal DDR SDRAMs without misses and bandwidth restrictions for simplicity. Also, we assume the memory supports two memory access requests concurrently and there are no bank conflicts that accessed the same memory bank simultaneously. However, this assumption of no bank conflicts is not realistic since the best DRAM system performance is obtained by maximizing the number of row-buffer hits while minimizing the number of bank conflicts. However, there are some techniques to reduce bank conflicts by tuning DRAM address mapping [11]. So, the effect of bank conflicts can be reduced depending on how effectively physical processor addresses are mapped to the channel and bank of a memory system. Since the address mapping (or some other technique) itself may affect the system performance directly, we believe that the assumption of ideal no bank conflicts is reasonable. The memory controller is based on the close-page-auto precharge policy, which is simple but efficient for SMP systems.

#### 3.1 Latency-Aware Memory Bus

In Fig. 3a, instead of using a fully routed and switched network for communication between the CPU cores and the shared memory, the CPU cores are connected to the primary memory (DDR SDRAMs) through a simple shared bus, using a memory controller for arbitration. We assume this memory controller is being located in the MPU, although it can be located in a companion chip (such as northbridge) to be a bridge between multiple master and slave devices. Communication between the cores is done by the shared bus. The bidirectional memory bus between the controller and the DRAM is implemented based on either the 3-PAM SSCDMA-I or the conventional 2-PAM TDM-I for simulation. The SSCDMA-I memory bus supports dual transactions on a single interconnect and the TDM-I memory bus supports only a single transaction at a time. Both the memory buses have the same number of data, control, and address lines with the same aggregate throughput (or bandwidth) per wire. Both the memory buses consist of 16-bit-wide data channels operating at 200 MHz (400 Mbps/pin). Thus, both buses can support the same theoretical peak memory bandwidth of 800 Mbyte/s. If we use a 64-bit-wide data bus, then it can support a peak memory bandwidth of 3.2 Gbyte/s. The
lattices to access the bus and memory are represented by CPU core cycles operating at 200 MHz.

To access the shared primary memory (DDR SDRAM), each CPU core first sends a request for the control of a bus (or simply “bus control”) to the bus arbiter located in the memory controller and then the core gets the bus control. Then, the CPU core sends read or write requests to the memory through the shared memory bus, which takes a channel flight time $t_f$. After the CPU sends its requests, it releases the bus control. After the memory receives the request, the DRAM prepares to send out the requested data, which requires a DRAM core access latency $t_{DRAM}$ for a read request. Depending on the consecutive command types, the DRAM core access latency could be only a CAS latency delay ($t_{CAS}$) or two other latency factors could be added [RAS to CAS delay ($t_{RCD}$) or row precharge time ($t_{RP}$)]. Based on the close-page-auto precharge policy that closes and then precharges the bank being accessed after each memory access, the DRAM core access latency for read request will be $t_{RAC}$ in most cases of our simulation model. In the meantime, the DRAM sends a request for the bus control to the memory controller. Then, the memory transmits the requested data through the shared bus. We assume that this memory supports critical-word-first fetch. It also takes channel flight time $t_f$ to arrive at the CPU.

Based on the above request procedures, the total bus arbitration process for a memory read access is assumed to be three processor cycles for simplicity: $t_{initial}$ is the initial delay time for bus arbitration (reply for a request) and initial setup time for transmission. Table 1 describes the single transaction read/write request latency parameters used in the system simulation. We assume the burst length of 4, which takes four and eight cycles for the transmission time ($t_{trans}$) to transmit 16 bytes of read data for cache line fill through the TDM and SSCDMA memory buses, respectively. It takes $t_{DRAM}$ (= $t_{RAC}$) of 8 cycles (= 40 ns at 200 MHz) to read data from the DDR memory banks. The write requests do not need to wait for memory access by assuming that the memory supports a write-buffer scheme.

### 3.2 Simulation Results and Analysis

Both the TDM and SSCDMA bus-based shared-memory SMP systems are simulated by a cycle-accurate simulation framework based on GEZEL with a data-flow multithreaded JPEG encoder program [3], [4]. Fig. 4 illustrates the data flow of the JPEG encoder. It consists of actors of different operations. The actors are communicating through intermediate queues. Each actor is initiated as a thread, which can be executed on any ARM core in the system.

Fig. 5 shows the simulated overall system performance of the shared-memory multicore SMP systems. The $x$-axis represents the number of ARM cores varying from 1 to 4. The $y$-axis represents the number of CPU cycles to perform the application program. From Fig. 5, when the number of operating CPU is one (single core case), both the TDM- and SSCDMA-based systems show no difference in performance. However, unlike the TDM bus-based system that shows performance degradation accordingly as the number of operating cores increases, the SSCDMA bus-based system shows continuously increased overall performance as the number of operating cores increases. It shows that SSCDMA improves the overall system performance about 5.8 percent (two cores: TDM of 82,622,747 cycles versus SSCDMA of 77,817,739 cycles) to 23.1 percent (four cores: TDM of 82,155,983 cycles versus SSCDMA of 63,216,047 cycles) due to the reduced bus contention and request latency as well as the reduced CPU stalls.

In the case of the TDM memory bus, each processing core releases the shared bus only after receiving the whole request data packet, which means the other pending requests from other processing cores have to wait until the end of the current data transmission. So, the conventional TDM memory bus has longer back-to-back read request latency, which delays the critical word fetch due to the bus contention and queuing delays. However, in the case of the SSCDMA memory bus, two read requests from two cores can be processed simultaneously without contention, which delivers the critical word for each core without bus contention interference as already illustrated in Fig. 2b. This improves the overall system performance since memory access is usually read dominant [5] and each CPU core requires the critical word as soon as possible to start computation without stalls.

Another important advantage of SSCDMA-I is that the shared bus channel can be dynamically allocated between cores. This is because that the single SSCDMA-I wire line is able to support dual virtual TDM-I communication channels. This means that there could be three scenarios, as shown in Fig. 6, for dynamic communication channel allocation: first, single core (CPU0) can fully occupy the memory (M0) bus with a full bandwidth when other CPUs do not need to access memory; second, single core (CPU0) communicates with two memories (M0 and M1) simultaneously through the same memory bus; third, while single core (CPU0) occupies the memory (M0) bus with one-half bandwidth,
the other core (CPU1) is able to occupy the same memory bus simultaneously with the other half bandwidth to communicate with M1. This dynamic channel allocation capability increases the bus utilization.

In addition, the packet size or burst length of a data packet also impacts the system performance. As shown in Fig. 7, as the burst length of the data packet becomes longer from four to eight time frames, the TDM-based system shows severe performance degradation. For example, when the number of operating CPU is four and the burst length is increased from 4 to 8, the number of CPU processing cycles has been increased 24.7 percent (from 82,155,983 to 10,246,047 cycles). However, for the same case in the SSCDMA-based system, the number of processing cycles has been increased only 18.6 percent (63,216,047 to 74,961,481 cycles), which is 6.1 percent smaller than that of the TDM-based case. This is because, as illustrated in Fig. 2, the TDM-based memory bus architecture has linearly increased critical word request latency as the burst transfer length goes up. On the other hand, the SSCDMA-based memory bus has a fixed value of back-to-back critical word latency regardless of the burst length variation for each two requests, resulting in less performance degradation when the number of operating cores increases.

4 MEMORY ACCESS LATENCY OF A HETEROGENEOUS MULTICORE SYSTEM

To see the advantage of SSCDMA-I from a different application example, we have modeled and simulated a heterogeneous multicore system shown in Fig. 8. It consists of four different processing cores, an ARM processor for general-purpose processing and control, a camera-interface IP, which transmits the still-image/video data to be processed on an image-video accelerator (IVA), and a DSP for digital signal processing. Two types of shared bus based on either a TDM-I or an SSCDMA-I are implemented to deliver data to different destinations. Unlike a homogeneous multicore system, a heterogeneous multicore system usually has large varieties in behavior of traffic loads. For example, the video data sent by the camera-interface IP has long burst lengths and is latency sensitive, while the data sent from ARM is shorter in lengths and more latency robust.

A mix of heterogeneous traffic loads is used to represent the shared bus accessing behavior of a heterogeneous multicore system (see Table 2). The traffic flows used in the experiment are synthetic flows. They are mimicking the traffic flows of a typical heterogeneous system. The video data transmitted from the camera to IVA has the highest priority. Reading an audio file from the external memory to DSP has a medium priority and the general processing on the ARM core has a low priority. The camera-interface IP sends video data to IVA, which has long burst lengths and is latency sensitive. The DSP reads an audio file from the main memory, which is transmitted on a regular data rate with medium burst lengths. It is also latency sensitive. ARM conducts a general-purpose processing, which has short burst lengths and is relatively latency robust.

The application mapped on this heterogeneous multicore system is a mix of traffic loads with different behavior. Table 3 gives a more detailed description of each traffic load. For example, the video data from camera interface to IVA (traffic A) has 1,000 bursts; each burst is 512 bytes. The audio data read from the memory to DSP (traffic B) consists of 40 bursts of 128 bytes. The

<table>
<thead>
<tr>
<th>Traffic Loads</th>
<th>Source and Destination</th>
<th>Traffic Behavior</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>Video data</td>
<td>Camera to IVA</td>
<td>Sporadic and long burst lengths, latency sensitive</td>
<td>High</td>
</tr>
<tr>
<td>Audio data</td>
<td>Off-chip Memory to DSP</td>
<td>Regular and medium burst lengths, latency sensitive</td>
<td>Medium</td>
</tr>
<tr>
<td>General computing</td>
<td>ARM to on-chip Memory</td>
<td>Sporadic and short burst lengths, latency robust</td>
<td>Low</td>
</tr>
</tbody>
</table>

Table 2: A Mix of Heterogeneous Traffic Loads
TABLE 3
Data Rates of a Heterogeneous Multicore System

<table>
<thead>
<tr>
<th>Traffic loads</th>
<th>Traffic rates of each access</th>
<th>Arbitration</th>
<th>Data Transmission of each burst (cycles)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>TDM</td>
<td>SSCDMA (on a single channel)</td>
</tr>
<tr>
<td>A Video data</td>
<td>256 bursts of 512B</td>
<td>2</td>
<td>64</td>
</tr>
<tr>
<td>B Audio data</td>
<td>64 bursts of 128B</td>
<td>2</td>
<td>16</td>
</tr>
<tr>
<td>C General computing</td>
<td>16B</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

general-purpose computing on the ARM core (traffic C) transmits 16 bytes every time it accesses the shared bus. The shared bus will service the request with the highest priority, which has been specified in Table 3.

Figs. 9a and 9b show the simulated latency results of each traffic load on the TDM and SSCDMA bus-based heterogeneous multicore systems, respectively. The latencies are measured from the time a processing core sends out a request, until the time it receives the complete request data from the memory. From Fig. 9a of the TDM bus, we can see that, due to the priority-based arbitration, traffic A (series 1) can always get access to the bus. However, it adds up the latencies of traffic B (series 2) and traffic C (series 3). Whenever traffic A has a burst, the latencies of traffic B and traffic C are increased up to 85 and 90 cycles, respectively, which are much higher than the latencies to transmit the required data (traffic $B = 2 + 16 = 18$ cycles and traffic $C = 2 + 2 = 4$ cycles, as specified in Table 3).

Fig. 9b shows the simulated latencies of traffic loads on the SSCDMA bus. Since a single SSCDMA-I line is able to operate as if it consists of two virtual TDM-I channels, here we are able to assign traffic A (series 1) to one channel and traffic B (series 2) and traffic C (series 3) to the other channel. Due to the dynamic channel allocation capability explained in Section 3, traffic loads can use the other channel if it is idle. For example, traffic A can use both channels to achieve the maximum bandwidth when traffic B and traffic C have no data to transmit. However, traffic A has to release the other channel whenever traffic B or C sends out the request to acquire the shared bus. Compared to the TDM bus case in Fig. 9a, traffic A has a higher latency of about 30 percent (from about 65 to 85), because it is limited to use only one channel. However, the maximum latencies for traffic B and traffic C have been reduced dramatically to 30 (~64 percent) and 20 (~78 percent) cycles, respectively. This demonstrates that the latencies of traffic B and traffic C become more predictable and are not interfered by traffic A as significantly as in the case of the TDM bus in Fig. 9a.

5 CONCLUSION

To address the performance limitations of the traditional TDM protocol-based memory bus (or link) of a multicore system, the advantages of the SSCDMA bus have been introduced. We have proposed a cost-effective, latency-aware SSCDMA memory bus that enables advanced parallelism in on- or off-chip communication with less network complexity and size. To demonstrate the benefit of the SSCDMA memory bus at the system level, two types of bus-based shared-memory multicore systems were modeled and simulated by a cycle-accurate simulation platform.

Fig. 9. Simulated latencies of traffic loads on (a) TDM bus and (b) SSCDMA bus.

Unlike the conventional TDM bus-based homogeneous multicore system that shows performance degradation as the number of operating CPU cores increases, the SSCDMA memory bus improves system performance by 23.1 percent for four cores. The maximum latency of the heterogeneous multicore system with a mix of traffic loads has been reduced up to 78 percent. These results show that the embedded multicore system performance can be increased with less cost and complexity by reducing the bus contention and by supporting higher concurrency in memory accesses. Through a system-level evaluation, we have shown how a circuit-level progress brings an increase in performance by allowing changes in the communication protocol of a multicore system.

ACKNOWLEDGMENTS

This work was supported in part by the Hongik University New Faculty Research Support Fund and in part by grants from the US National Science Foundation, US Defense Agency Research Projects Agency, and SRC. The authors would like to thank Patrick Schaumont and Herwin Chan for valuable discussions.

REFERENCES


For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/publications/dlib.