Power Aware External Bus Arbitration for System-on-a-Chip Embedded Systems

Ke Ning[1][2] and David Kaeli[1]

[1] Northeastern University 360 Huntington Avenue, Boston MA 02115
[2] Analog Devices Inc. 3 Technology Way Norwood MA 02062

Abstract. Power efficiency has become a key design trade-off in embedded system designs. For system-on-a-chip embedded systems, an external bus interconnects embedded processor cores, I/O peripherals, a direct memory access (DMA) controller, and off-chip memory. External memory access activities are a major source of energy consumption in embedded systems, and especially in multimedia platforms. In this paper, we focus on the energy dissipated due to the address, data, and control activity on the external bus and supporting logic. We build our external bus power model on top of a cycle-accurate simulation framework that quantifies the bus power based on memory bus state transitions. We select an Analog Devices ADSP-BF533 multimedia system-on-a-chip embedded system as our target architecture model. Using our power-aware external bus arbitration schemes, we can reduce overall power by as much as 18% in video processing applications, and by 12% on average for the test suites studied. Besides reducing power consumption, we also obtained an average performance speedup of 24% when using our power-aware arbitration schemes.

Keywords: power-aware, external memory, bus arbitration, embedded systems, media processor.

1 Introduction

Modern embedded systems are becoming increasingly limited by memory performance and system power consumption. The power associated with off-chip accesses can dominate the overall power budget. The memory power/speed problem is even more acute for embedded media processors that possess memory intensive access patterns and require streaming serial memory access that tends to exhibit low temporal locality (i.e., poor data cachablity). Without more effective bus communication strategies, media processors will continue to be limited by memory power and memory performance.

One approach to addressing both issues is to consider how best to schedule off-chip accesses. Due to the intrinsic capacitance of the bus lines, a considerable amount of power is required at the I/O pins of a system-on-a-chip processor when data has to be transmitted through the external bus [1, 2]. The capacitance associated with the external bus is much higher than the internal node capacitance inside a microprocessor. For example, a low-power embedded microprocessor system like an Analog Devices ADSP-BF533 running at 500 MHz consumes about 374 mW on average during normal execution. Assuming a 3.65 V voltage supply and a bus frequency of 133 MHz, the average external power consumed is around 170 mW, which accounts for approximately 30% of the overall system power dissipation [3].

In modern CMOS circuit design, the power dissipation of the external bus is directly proportional to the capacitance of the bus and the number of transitions ($1 \rightarrow 0$ or $0 \rightarrow 1$) on bus lines [4, 5]. In general, the external bus power can be expressed as:

$$P_{bus} = C_{bus} V_{ext}^2 f k \mu + P_{leakage} \tag{1}$$

In the above equation, C_{bus} denotes the capacitance of each line on the bus, V_{ext} is the bus supply voltage, f is the bus frequency, k is the number of bit toggles per transition on the full width of the bus, and μ is the bus utilization factor. This power equation is an activity-based model. It not only accounts for the dynamic power dissipated on the bus, but includes the pin power that drives the signal I/O's related to external bus communication. $P_{leakage}$ is the power dissipated on the bus due to leakage current.

The techniques to minimize the power dissipation in buses have been well explored in previous research. The main strategies have been to utilize improved bus encodings to minimize the bus activity. Various mixed-bus encoding techniques (e.g., Gray codes and redundant codes) were developed to save on bus power. Gray code addressing is based on the fact that bus values tend to change sequentially and they can be used to switch the least number of signals on the bus.

However, better performance can be obtained by using redundant codes [1]. A number of redundant codes have been proposed that add signals on the bus lines in order to reduce the number of transitions. Bus-invert coding [6]is one class of the redundant codes. Bus-invert coding adds an INV signal on the bus to represent the polarity of the address on the bus. The INV signal value is chosen by considering how best to minimize the hamming distance between the last address on the bus and the current one. Some codes can be applied to both the data and address buses, though some are more appropriate for addresses.

In our previous work, we described a bus modeling system that can capture bus power in the same framework of a cycle-accurate simulator for an embedded media processor [7]. We discussed an initial design of a power-aware bus arbitration scheme. The main contributions of this paper are a completed design of our power-aware bus arbitration scheme that also considers using pipelined SDRAM, and we also consider a broader range of multimedia applications. This paper is organized as follows. In section 2 we describe the target architecture for our work, which contains a system-on-a-chip media processor, SDRAM memory, and an external bus interface unit. We also present our power modeling methodology. Section 3 describes a number of different bus arbitration algorithms that we consider for power and performance optimizations. Section 4 presents power/performance results of MPEG-2, JPEG, and PGP benchmarks for traditional arbitration schemes and our power-aware schemes. Finally, Section 5 presents conclusions.

2 System-on-a-Chip Architectures

2.1 Interconnect Subsystem

Modern system-on-a-chip embedded media systems include many components: a high-speed processor core, hardware accelerators, a rich set of peripherals, direct memory access (DMA), on-chip cache and off-chip memory. The system architecture considered in our study includes a single core, several peripherals, and off-chip SDRAM memory, and is similar to many current embedded platforms.

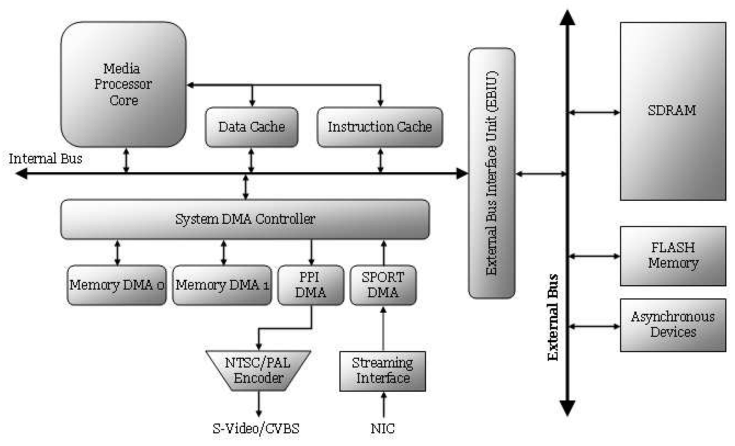

Fig. 1. Our target embedded media system architecture.

For multimedia applications, throughput requirements are increasing faster and faster. Today, for a D1 (720x480) video codec (encoder/decoder) media node, we need to be able to process 10 million pixels per second. This workload requires a media processor for computation, devices to support high speed media streaming and data conversion via a parallel peripheral interface (PPI), and a synchronous serial port (SPORT) for interfacing to high speed telecom interfaces. The high data throughput requirements associated with this platform make it impossible to store all the data in an on-chip memory or cache. Therefore, a typical multimedia embedded system usually provides a high-speed system-on-a-chip microprocessor and a very large off-chip memory. The Analog

Devices Blackfin family processors, the Texas Instrument OMAP, and the SigmaDesign EM8400 series are all examples of low-power embedded media chipsets which share many similarities in system design and bus structure. The system architecture assumed in this paper is based on these designs and is shown in Figure 1.

When trying to process streaming data in real-time, the greatest challenge is to provide enough memory bandwidth in order to sustain the necessary data rate. To insure sufficient bandwidth, hardware designers usually provide multiple buses in the system, each possessing different bus speeds and different protocols. An external bus is used to interface to the large off-chip memory system and other asynchronous memory-mapped devices. The external bus has a much longer physical length than other buses, and thus typically has higher bus capacitance and power dissipation. The goal of this work is to accurately model this power dissipation in a complete system power model so we can explore new power-efficient scheduling algorithms for the external memory bus.

2.2 External Bus Interface Unit

In the system design shown in Figure 1, there are two buses, one internal bus and one external bus, These two buses are bridged by an external bus interface unit (EBIU), which provides a glue-less interface to external devices (i.e., SDRAM memory, flash memory and asynchronous devices).

There are two sub-modules inside the EBIU, a bus arbitrator and a memory controller. When the units (processor or DMA's) in the system need to access external memory, they only need to issue a request to the EBIU buffer through the internal bus. The EBIU will read the request and handle the off-chip communication tasks through the external bus. Due to the potential contention between users on the bus, arbitration for the external bus interface is required. The bus arbitrator grants requests based on a pre-defined order. Only one access request can be granted at a time. When a request has been granted, the memory controller will communicate with the off-chip memory directly based on the specific memory type and protocol. The EBIU can support SDRAM, SRAM, ROM, FIFOs, flash memory and ASIC/FPGA designs, while the internal units do not need to discriminate between different memory types. In this paper, we use multi-banked SDRAM as an example memory technology and integrate SDRAM state transitions into our external bus model (our modeling framework allows us to consider different memory technologies, without changing the base system-on-a-chip model).

2.3 Bus Power Model

The external bus power includes dynamic power to charge and discharge the capacitance along the external bus, and the pin power to drive the bus current. The external bus power is highly dependent on the memory technology chosen. In past work on bus power modeling, little attention has been paid to the impact of the chosen memory technology. While we have assumed an SDRAM in our power model in this work, we can use the same approach with other types of memory modules. The external bus power associated with each transaction will be the total number of pins that toggle on the bus. We include in our model the power consumption due to the commands sent on the control bus, the row address and column address on the address bus, and the data on data bus. The corresponding leakage power is also considered in our model.

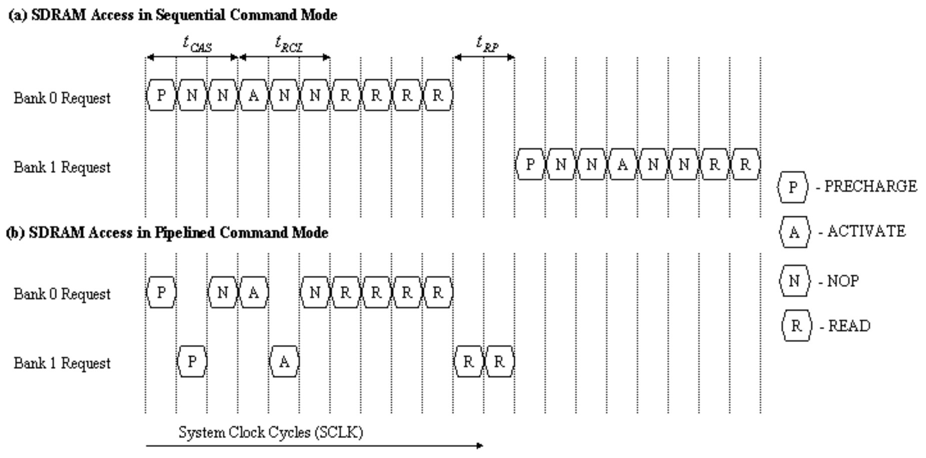

Fig. 2. Timing diagram showing two memory accesses for both sequential and pipelined command SDRAM.

SDRAM is commonly used in cost-sensitive embedded applications that require large amounts of memory. SDRAM has a three-dimensional structure model. It is organized in multiple banks. Inside each bank, there are many pages, which are selected by row address. The memory access is on a command-by-command basis. An access involves processing a PRECHARGE and an ACTIVATE command before a physical READ/WRITE command. At the same time of an ACTIVATE and READ/WRITE command, the corresponding row and column addresses are sent on the address bus.

To maximize memory bandwidth, modern SDRAM components allow for pipelining memory commands [8], which eliminates unnecessary stall cycles and NOP commands on the bus. While these features increase the memory bandwidth, they also reduce the bus command power. Consecutive accesses to different rows within one bank have high latency and cannot be pipelined, while consecutive accesses to different rows in different banks can be pipelined. Figure 2 is a timing diagram for processing two read operations in sequential access SDRAM and pipelined access SDRAM.

In our bus model, we assume that the power to drive the control bus and address bus are the similar. For each read/write request, we first determine the series of commands needed to complete that request. For each command, the bus state transitions, the number of pin toggles, and the bus utilization factor is recorded. Finally, the average bus power dissipated is calculated using Equation 1.

3 Bus Arbitration

The bandwidth and latency of external memory system are heavily dependent on the manner in which accesses interact with the three-dimensional SDRAM structure. The bus arbitration unit in the EBIU determines the sequencing of load/store requests to SDRAM, with the goals of reducing contention and maximizing bus performance. The requests from each unit will be queued in the EBIU's wait queue buffer. When a request is not immediately granted, the request enters stall mode. Each request can be represented as a tuple (t, s, b, l), where t is the arrival time, s identifies the request (load or store), b is the address of the block, and l is the extent of the block. The arbitration algorithm schedules requests sitting in the wait queue buffer with a particular performance goal in mind. The algorithm needs to guarantee that bus starvation will not occur.

3.1 Traditional Algorithms

A number of different arbitration algorithms have been used in microprocessor system bus designs. The simplest algorithm is *First Come First Serve* (FCFS). In this algorithm, requests are granted on the bus based on the order of arrival. This algorithm simply removes contention on the external bus without any optimization and pre-knowledge of the system configuration. Because FCFS schedules the bus naively, the system performs poorly due to instruction and data cache stalls. The priority of cache accesses and DMA access are equal (though cache accesses tend to be more performance critical than DMA accesses). An alternative is to have a *Fixed Priority* scheme where cache accesses are assigned higher priority than DMA accesses. For different DMA accesses, peripheral DMA accesses will have higher priority than memory DMA accesses. This differentiation is needed because if a peripheral device access is held off for a long period of time, it could cause the peripheral to lose data or time out. The Fixed Priority scheme selects the request with highest priority in the waiting queue instead of just selecting the oldest. Using Fixed Priority may provide similar external bus performance as the FCFS algorithm, but the overall system performance should be better if the application is dominated by cache accesses. For real-time embedded applications which are dominated by DMA accesses, cache accesses can be tuned such that cache misses are infrequent. Cache fetches can be controlled to occur only at non-critical times using cache prefetching and locking mechanisms. Therefore, for real-time embedded applications, the FCFS and Fixed Priority schemes produce very similar external bus behavior.

3.2 Power Aware Algorithms

To achieve efficient external bus performance, FCFS and Fixed Priority are not sufficient. Power and speed are two major factors of bus performance. In previous related work, dynamic external bus arbitration and scheduling decisions were primarily driven by bus performance and memory bandwidth [8, 9]. If a power-efficient arbitration algorithm is aware of the power and cycle costs associated with each bus request in the queue, each request can be scheduled to achieve more balanced power/performance. The optimization target can be to minimize power P, minimize delay D, or more generally to minimize $P^n D^m$. This problem can be formulated as a shortest Hamiltonian path (SHP) on a properly defined graph. The Hamiltonian path is defined as the path in a directed graph that visits each vertex exactly once, without any cycles. The shortest Hamiltonian path is the Hamiltonian path that has the minimum weight. The problem is NP-complete, and in practice, heuristic methods are used to solve the problem [10].

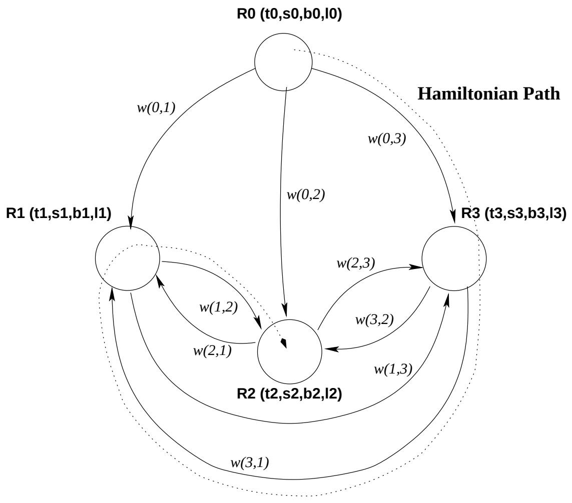

Fig. 3. Hamiltonian Path Graph

Let R_0 denote the most recently serviced request on the external bus. R_1, R_2, ... R_L are the requests in the wait queue. Each request R_i consists of four elements (t_i, s_i, b_i, l_i), representing the arrival time, the access type (load/store), the starting address, and the access length. The bus power and delay are dependent on the current bus state and the following bus state for each request. The current bus state is the state of the bus after the previous bus access has completed. $P(i, j)$ represents the bus power dissipated for request R_j, given R_i was the immediate past request. $D(i, j)$ is the time between when request R_j is issued and when R_j is completed, where R_i was the immediate past request. The cost associated with scheduling request R_j after request R_i can be formulated as $P^n(i,j)D^m(i,j)$. We can define a directed graph $G = (V, E)$ whose vertices are the requests in the wait queue, with vertex 0 representing the last request completed. The edges of the graph include all pairs (i, j). Each edge is assigned a weight $w(i, j)$, and is equal to the power delay product of processing request R_j after request R_i.

$$w(i,j) = P^n(i,j)D^m(i,j), n, m = 0, 1, ... \tag{2}$$

The problem of optimal bus arbitration is equivalent to the problem of finding a Hamiltonian path starting from vertex 0 in graph G with a minimum path traversal weight. Figure 3 describes a case when there are 3 requests are in the wait queue. One of the Hamiltonian paths is illustrated with a dot line. The weight of this path is $w(0, 3) + w(3, 1) + w(1, 2)$. For each iteration, a shortest Hamiltonian path will be computed to produce the minimum weight path. The first request after request R_0 on that path will be the request selected in next bus cycle. After the next request is completed, a new graph will be constructed and a new minimum Hamiltonian path will be found.

Finding the shortest Hamiltonian path has been shown to be NP-complete. To produce a shortest path, we use heuristics. Whenever the path reaches vertex R_i, the next request R_k with minimum $w(i, k)$ will be chosen. This is a greedy algorithm, which selects the lowest weight for each step. The bus arbitration algorithm only selects the second vertex on that path. We avoid searching the full Hamiltonian path, and so the bus arbitration algorithm can simply select a request based on finding the minimum $w(0, k)$ from request R_0. The complexity of this heuristic is $O(L)$. When $w(i, j) = P(i, j)$, arbitration will try to minimize power. When $w(i, j) = D(i, j)$, then we can minimize delay. To consider the power efficiency, the power delay product can be used. Selecting different values for n and m change how we trade off power with delay using weights $w(i, j)$.

3.3 Target Architecture

In our experimental study, we used a power model of the Analog Devices Blackfin family system-on-a-chip processors as our primary system model. We run code developed for ADSP-BF533 EZ-Kit Lite board using the VisualDSP++ toolset. This board provides a 500 MHz ADSP-BF533 microprocessor, 16 MB of SDRAM, and a CCIR-656 video I/O interface. Inside the ADSP-BF533 microprocessor, there are both L1 instruction and data caches. The instruction cache is 16 KB 4-way set associative. The data cache is 16 KB 2-way set associative. Both caches use a 32 byte cache line size. The SDRAM module selected is the Micron MT48LC16M16A2 16 MB SDRAM. The SDRAM interface connects to 128 Mbit SDRAM devices to form one 16 MB of external memory. The SDRAM is organized in 4 banks, with a 1 KB page size. It also has following characteristics to match the on-chip SDRAM controller specification: 3.3V supply voltage, 133 MHz operating frequency, burst length of 1, column address strobe (CAS) latency t_{CAS} of 3 system clock cycles, t_{RP} and t_{RCD} equal to 2 system clock cycles, refresh rate programmed at 4095 system clock cycles. We used the Analog Devices Blackfin frio-eas-rev0.1.7 toolkit to integrate this model. The power model has been validated with physical measurements as described in [11]. To make sure the arbitration algorithm does not produce long-term starvation, a time-out mechanism was added for the requests. The timeout values for cache and memory DMA are 100 and 550 cycles, respectively.

Table 1. Benchmarks

Name	**Description**
MPEG2-ENC	MPEG-2 Video encoder with 720x480 4:2:0 input frames.
MPEG2-DEC	MPEG-2 Video decoder of 720x480 sequence with 4:2:2 CCIR frame output.
JPEG-ENC	JPEG image encoder for 512x512 image.
JPEG-DEC	JPEG image decoder for 512x512 image.
PGP-ENC	Pretty Good Privacy encryption and digital signature of text message.
PGP-DEC	Pretty Good Privacy decryption of encrypted message.

4 Experimental Setup

4.1 Benchmarks

Experiments were run on a set of multimedia workloads. We chose MPEG-2 for video processing, JPEG for image compression and PGP for cryptography. All three benchmark suites are representative and commonly used applications for multimedia processing. MPEG-2 is the dominant standard for high-quality digital video transmission and DVD. We selected real-time MPEG-2 encoder and decoder source codes that include optimized Blackfin MPEG-2 libraries. The input datasets used are the *cheerleader* for encoding (the size is 720x480 and the format is interlaced video) and *tennis* for decoding (this image is encoded by the MPEG-2 reference encoder, the size is also 720x480, and the format is progressive video). Both inputs are commonly used by the commercial multimedia community.

JPEG is a standard lossy compression method for full color images. The JPEG encoder and decoder used also employ optimized Blackfin libraries. The input image is Lena (the size is 512x512 in a 4:2:2 color space).

PGP stands for *Pretty Good Privacy*, and provides for encryption and signing data. The signature we use is a 1024 bit cryptographically-strong one-way hash function of the message (MD5). To encrypt data, PGP uses a block-cipher IDEA and RSA for key management and digital signature.

In order to measure the external bus power and be able to assess the impact of our power-efficient bus arbitration algorithm, we developed the following simulation framework. First, we modified the Blackfin instruction-level simulator to include the system bus model and cache activity. From this model, we feed all accesses generated on the external bus to an EBIU model. The EBIU model faithfully simulates the external bus behavior, capturing detailed SDRAM state transitions and allows us to considered different bus arbitration schemes. The average bus power and performance are computed from the simulation results produced by our integrated simulator.

4.2 Results

There are eleven different bus arbitration schemes evaluated in our simulation environment. We considered two traditional schemes: (1) Fixed Priority (FP), (2) First Come First Serve (FCFS), and 9 different power-aware schemes. For Fixed Priority, we assign the following priority order (from highest to lowest): instruction cache, data cache, PPI DMA, SPORT DMA, memory DMA. In the power-aware schemes, each scheme is represented by the pair of power/delay coefficients (n, m) of the arbitration algorithm. n and m are the exponents shown in Equation 2. Different n and m values will favor either power or delay. (1, 0) is the minimum power scheme, (0, 1) is the minimum delay scheme, and (1, 1), (1, 2), (2, 1), (1, 3), (2, 3), (3, 2), (3, 1) consider a balance between power and delay by using different optimization weights. We present experimental results for both power and delay. The MPEG-2 encoder and decoder simulation results are shown in Figure 4, JPEG encoder and decoder are shown in Figure 5 and PGP encryptor and decryptor are shown in Figure 6. All the experiments consider both sequential command mode SDRAM and pipelined command mode SDRAM.

In all applications, the power dissipation for the power-aware arbitration schemes is much lower when compared to using Fixed Priority or FCFS. The power-aware schemes also benefit from fewer bus delays. These conclusions are consistent across all of the applications studied and are also consistent when using either sequential command SDRAM or pipelined command SDRAM. In the MPEG-2 case, the power-aware scheme (1, 0) enjoys an 18% power savings relative to a Fixed Priority scheme for encoder and 17% for decoder. The same power-aware scheme also achieved a 40% reduction in cycles when compared to the Fixed Priority scheme on MPEG-2 decoder, and a 10% reduction for MPEG-2 encoder.

To factor out the impact of sequential versus pipelined command mode from the power savings, we show in Table 2 the bus power savings and we show in Table 3 the cycle savings. Inspecting the two tables, we can see that the power-aware arbitration scheme achieves an average power savings of 12% and an average speedup of 24% over all 6 applications. There exist some variations in power savings and speed up achieved. These variations are primarily due to differences in bus utilization across the different applications. For high traffic applications, external memory access requests are more bursty, In those cases, our power-aware schemes provide a larger improvement than in low traffic applications, in which the requests are less bursty. The greater the number of requests in the queue, the greater the opportunity that the arbitrator can effect an improvement. Similarly, in Tables 4 and 5, the pipelined command SDRAM obtains on average a 13% power savings and a 18% performance speedup.

From inspecting the results shown in Tables 2-5, we can see that the choice of using sequential versus pipelined command modes is not a factor when con-

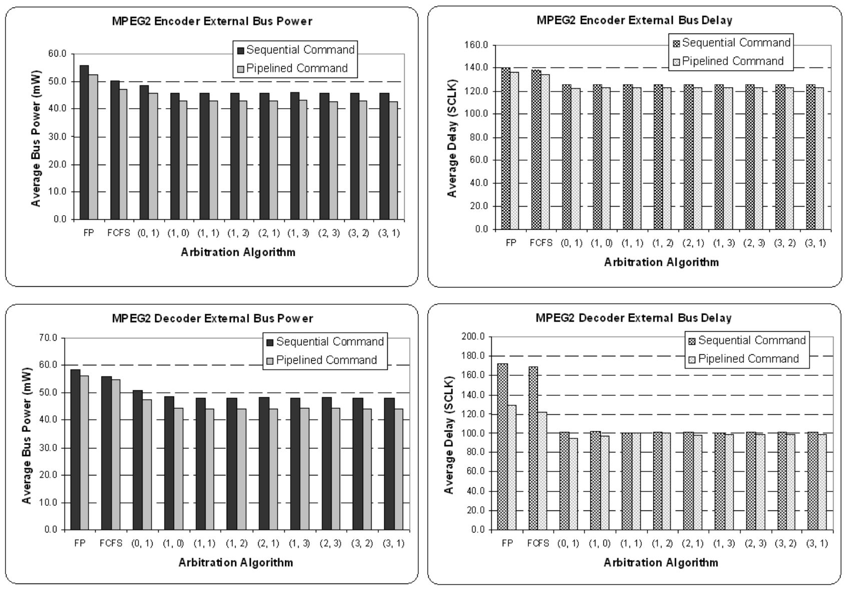

Fig. 4. External bus power/delay results for MPEG-2 video encoder and decoder.

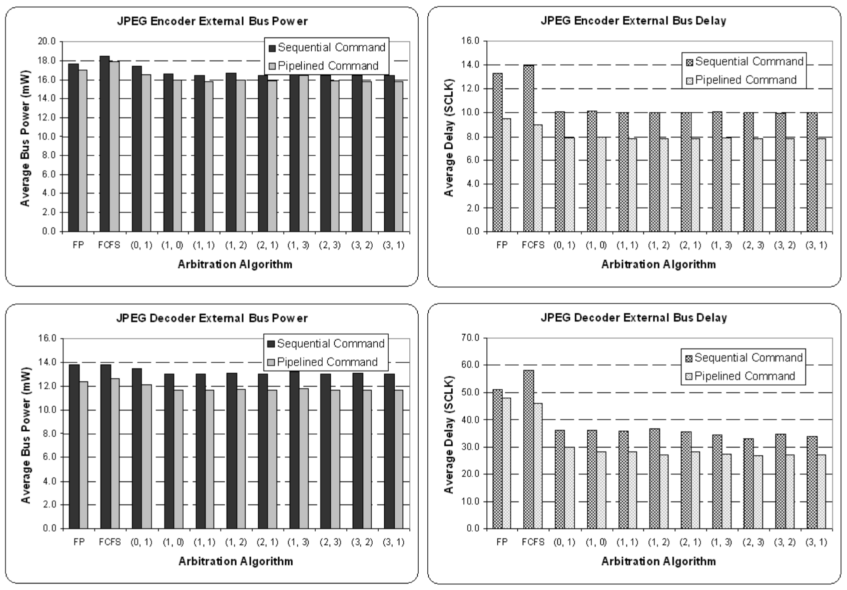

Fig. 5. External bus power/delay for JPEG image encoder and decoder.

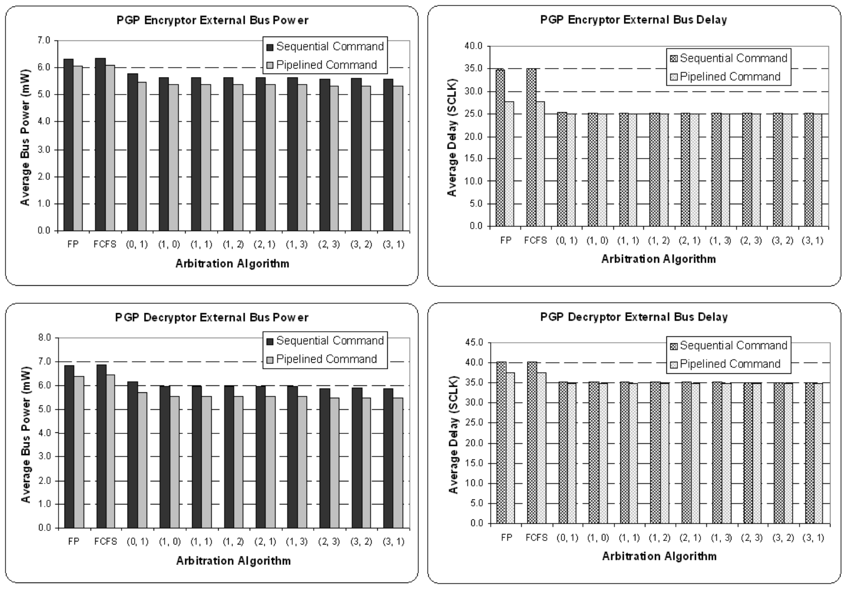

Fig. 6. External bus power/delay for PGP encryption and decryption.

Table 2. External bus power savings of (1, 0) arbitration vs. fixed priority arbitration in sequential command SDRAM.

	Fixed priority power (mW)	**Arbitration (1, 0) Power (mW)**	**Power savings**
MPEG2-ENC	55.85	45.94	18%
MPEG2-DEC	58.47	48.62	17%
JPEG-ENC	17.64	16.57	6%
JPEG-DEC	13.77	12.99	6%
PGP-ENC	6.33	5.66	11%
PGP-DEC	6.81	5.94	13%
Average savings			12%

Table 3. External bus speedup of (1, 0) arbitration vs. fixed priority in sequential command SDRAM.

	Fixed priority delay (SCLK)	**Arbitration (1, 0) Delay (SCLK)**	**Speedup**
MPEG2-ENC	140.36	126.10	10%
MPEG2-DEC	171.94	101.52	41%
JPEG-ENC	13.30	10.19	23%
JPEG-DEC	51.22	36.04	30%
PGP-ENC	34.87	25.21	28%
PGP-DEC	40.28	35.22	13%
Average Speedup			24%

Table 4. External bus power savings of (1, 0) arbitration vs. fixed priority arbitration in pipelined command SDRAM.

	Fixed Priority Power (mW)	**Arbitration (1, 0) Power (mW)**	**Power savings**
MPEG2-ENC	52.51	42.84	18%
MPEG2-DEC	56.29	44.58	21%
JPEG-ENC	16.97	15.93	6%
JPEG-DEC	12.41	11.68	6%
PGP-ENC	6.05	5.39	11%
PGP-DEC	6.40	5.54	13%
Average savings			13%

Table 5. External bus speedup of (1, 0) arbitration vs. fixed priority in pipelined command SDRAM.

	Fixed priority delay (SCLK)	**Arbitration (1, 0) Delay (SCLK)**	**Speedup**
MPEG2-ENC	136.73	122.79	10%
MPEG2-DEC	128.82	97.57	24%
JPEG-ENC	9.50	7.93	16%
JPEG-DEC	48.02	28.20	41%
PGP-ENC	27.61	24.92	10%
PGP-DEC	37.34	34.78	7%
Average Gain			18%

sidering the performance impact of the bus arbitration algorithm. However, the pipelined command mode does decrease the request delay period by overlapping bus command stall cycles with other non-collision producing commands. Pipelining also helps to reduce the bus power by using one command's PRECHARGE or ACTIVATE stall cycles to prepare for the next READ/WRITE command (versus sending NOP commands). Table 6 summarizes the results between the sequential command mode and pipelined command mode SDRAMs. The results show that the pipelined command mode SDRAM can produce a 6% power savings and a 10% speedup.

Table 6. Power and speed improvements for pipelined vs. sequential command mode SDRAM.

	Avg. power in sequential mode (mW)	**Avg. power in pipelined mode (mW)**	**Power reduction of pipelined commands**	**Avg. delay sequential mode (SCLK)**	**Avg. delay in pipelined mode (SCLK)**	**Speedup of in pipelined commands**
MPEG2-ENC	47.45	44.40	6%	128.31	124.99	3%
MPEG2-DEC	50.13	46.64	7%	113.38	103.38	9%
JPEG-ENC	16.86	16.26	4%	10.68	8.10	24%
JPEG-DEC	13.21	11.89	10%	38.74	31.28	19%
PGP-ENC	5.78	5.50	5%	26.95	25.40	6%
PGP-DEC	6.11	5.70	7%	36.09	35.25	2%
Avg. speedup			6%			10%

Comparing the results across the power-efficient schemes, we can see that the performance differences are small, and that no one scheme provides significant advantages over the rest. The scheme (1, 0) (i.e., the minimum power approach) is actually more favorable with regards to design implementation. Scheme (1, 0) basically needs a Hamming distance (XOR) computation unit and a comparator. For each iteration, the arbitrator uses the Hamming distance computation unit to accumulate the power used for each request that is pending in the wait queue, and uses the comparator to select the minimum. For $0.13\mu m$ CMOS technology and a 1.2 V power supply, an XOR transistor takes about 30 fJ to switch the transistor state in the slow N and slow P process corner. In our case, the number of transistors to implement the (1, 0) arbitrator is on the order of 10^3.

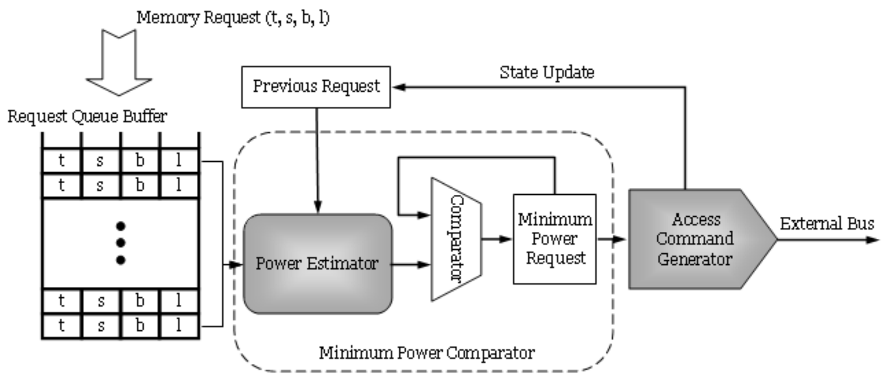

Fig. 7. Our power-aware bus arbitration architecture.

Figure 7 shows the architecture of the (1, 0) power-aware arbitrator. It contains three major components, a single request queue, a single minimum power comparator, and the memory access command generator. As memory access requests arrive, they allocate storage space while waiting for service in the request queue. The power estimator computes a request's power relative to the previous request. The comparator selects the minimum power request and stores it. When the currently active request finishes, the minimum power request will be sent to the bus by the access command generator, which also updates the previous request register which will be used to perform request selection in the next cycle. Power estimation is performed serially for each request in the queue. All requests will share the same power estimator logic. Using such a shared structure will reduce the hardware complexity, but may introduce some latency. In future work we will investigate how to provide for parallel power estimation that balances latency and power overhead.

5 Conclusions

Memory bandwidth has become a limiting factor in high performance embedded computing. For future multimedia processing systems, bandwidth and power are both critical issues that need to be addressed. This paper proposes a set of new external bus arbitration schemes that balance bus power and delay. Our experiments are based on modeling a low-end embedded multimedia architecture while running six multimedia benchmarks. Our results show that significant power reductions and performance gains can be achieved using power-aware bus arbitration schemes compared to traditional arbitration schemes. We also considered the impact of using both sequential and pipelined SDRAM models. Finally, a hardware implementation of (1, 0) power-aware arbitrator is proposed.

References

1. Benini, L., De Micheli, G., Macii, E., Sciuto, D., Silvano, C.: Address bus encoding techniques for system-level power optimization. In: Proceedings of the Conference on Design, Automation and Test in Europe, IEEE Computer Society (1998) 861–867
2. Panda, P.R., Dutt, N.D.: Reducing address bus transitions for low power memory mapping. In: Proceedings of the 1996 European Conference on Design and Test, IEEE Computer Society (1996) 63
3. Analog Devices Inc. Norwood, MA: Engineer-to-Engineer Note EE-229: Estimating Power for ADSP-BF533 Blackfin Processors (Rev 1.0). (2004)
4. Givargis, T.D., Vahid, F., Henkel, J.: Fast cache and bus power estimation for parameterized system-on-a-chip design. In: Proceedings of the Conference on Design, Automation and Test in Europe (DATE), ACM Press (2000) 333–339
5. Sotiriadis, P., Chandrakasan, A.: Low-power bus coding techniques considering inter-wire capacitances. In: Proceedings of IEEE Conference on Custom Integrated Circuits (CICC'00). (2000) 507–510
6. Stan, M., Burleson, W.: Bus-invert coding for low-power I/O. IEEE Transactions on Very Large Scale Integration (VLSI) Systems (1995) 49–58
7. Ning, K., Kaeli, D.: Bus power estimation and power-efficient bus arbitration for system-on-a-chip embedded systems. In: Workshop on Powre-Aware Computer Systems PACS'04, 37th Annual IEEE/ACM International Symposium on Microachitecture (2004)
8. Rixner, S., Dally, W.J., Kapasi, U.J., Mattson, P., Owens, J.D.: Memory access scheduling. In: ISCA '00: Proceedings of the 27th Annual International Symposium on Computer Architecture, New York, NY, USA, ACM Press (2000) 128–138
9. Lyuh, C.G., Kim, T.: Memory access scheduling and binding considering energy minimization in multi-bank memory systems. In: DAC '04: Proceedings of the 41st Annual Conference on Design Automation, New York, NY, USA, ACM Press (2004) 81–86
10. Rubin, F.: A search procedure for hamilton paths and circuits. J. ACM **21** (1974) 576–580
11. VanderSanden, S., Gentile, R., Kaeli, D., Olivadoti, G.: Developing energy-aware strategies for the blackfin processor. In: Proceedings of Annual Workshop on High Performance Embedded Computing, MIT Lincoln Laboratory (2004)