History-Based Memory Mode Prediction for Improving Memory Performance

Seong-Il Park and In-Cheol Park
Division of EE, Dept. of EECS, KAIST,
373-1, Guseong-dong, Yuseong-gu, Daejeon, 305-701, Republic of Korea
Tel : 042-869-4405 Fax : 042-869-4410
Email : sipark@ics.kaist.ac.kr, icpark@ee.kaist.ac.kr

Abstract

To increase the bandwidth of synchronous memories that are widely adopted for high performance memory systems, a predictive mode control scheme is proposed to reduce memory latency by effectively managing the states of banks. The local access history of each bank is considered to predict the memory mode. Experimental results show that the proposed scheme, at the cost of negligible area overhead, reduces the memory latency by 19.0% over the conventional scheme that always keeps the memory in idle state.

1. Introduction

In many applications such as portable wireless devices and multimedia systems, several factors such as increased system complexity, time-to-market pressure, and various functionality requirements have made the trend of system-on-a-chip (SoC) design indispensable. In general, SoC devices are connected to off-chip memories that feed instructions and data to the programmable processors and temporarily store data to be transferred between functional blocks, as shown in Fig. 1. As the SoC integrates more functional blocks and needs higher performance to carry out ever increasing tasks, high data bandwidth is required to meet a given system specification.

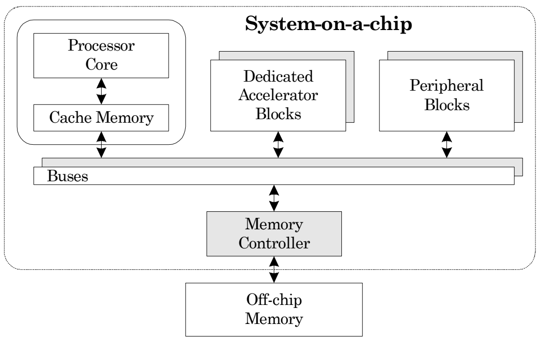

Fig. 1. An illustrative SoC architecture.

Synchronous memories such as Synchronous DRAM (SDRAM) and Rambus DRAM are widely used to enhance the performance of memory systems in various applications [1]. Several architectural features developed to alleviate memory latency enable the synchronous memories to meet the bandwidth requirement [2]. The features are based on the fact that all the cells along a word line are latched to sense amplifiers and reused without row-activations and precharges as long as the row addresses of successive accesses are identical. The row-active state can be used to reduce the latency and the power consumption of memory operations if the successive memory access refers to the same row in the same bank (a page hit). However, if the row address differs from the previous one (a page miss), additional cycles that cannot be hided are needed for a precharge and a row-activation, resulting in performance degradation. Therefore, to increase memory performance, the memory controller has to control the operation mode by efficiently predicting whether the next memory reference causes a page hit or not.

Several optimizations have been proposed to reduce page misses by statically scheduling the address sequence in memory and controlling the memory operation mode [3][4][5]. Those techniques are successfully applied to image and video processing applications, in which memory access patterns are relatively regular enough to be known in advance.

A dynamic memory mode control scheme [6] has been proposed to manage the memory operation mode according to runtime behavior of memory access patterns. The state of SDRAM is changed from idle to row-active state if a memory access leads to a page hit and sustains the row-active state until the number of the successive page misses exceeds a threshold value. The dynamic scheme is effective if in-row accesses are dominant. For example, in a system with a cache memory, several consecutive words in a cache block have to be transferred for a cache miss. However, if in-row accesses are not dominant and the pattern of memory accesses is irregular, frequent mode transitions lead to many overhead cycles for precharges and row-activations.

In this paper, we propose a new dynamic memory mode control scheme to reduce memory latency by predicting the next operation mode. The prediction is based on the history of memory references.

The rest of the paper is organized as follows. Section 2 gives a brief background on the architecture, the bank states, and the operations of SDRAM. We describe the proposed dynamic memory mode control scheme in Section 3. In Section 4, experimental methodology and results are presented. Finally, conclusions are made in Section 5.

2. Background on the SDRAM

Fig. 2 shows a simplified block diagram of SDRAM, which consists of four independent banks. The four banks share address buffers and I/O buffers, while each bank has its own row decoders, column decoders, sense amplifiers, and a memory array. The state of resources of a bank is maintained independently.

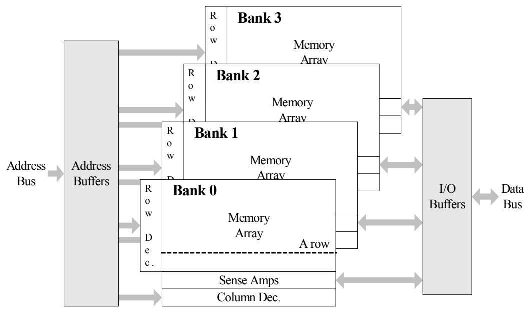

Fig. 2. Block diagram of the SDRAM.

Each bank has two stable states that are idle and row-active states as shown in Fig. 3. The idle state is entered by the precharge operation. The state transition from idle to row-active is made by the row-activation operation. Column access operations do not change the state of the bank. Thus the bank is in row-active state as long as the precharge operation is not performed.

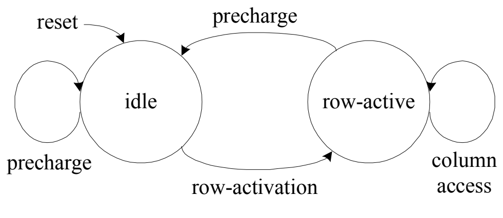

Fig. 3. State diagram for each bank.

The operation mode of the SDRAM is controlled by a memory controller that translates a read/write request into a sequence of memory commands. Three major operations of the SDRAM are as follows:

- Row-activation: The bank and the row where the data are accessed are selected and activated. Then, all the cells along a word (row) line of the bank are latched to the corresponding sense amplifiers. The bank is in row-active state after completing the operation.
- Column access: The column access operation selects and accesses a column of the corresponding activated row. A number of words equal to the burst length are read out from the sense amplifiers to the I/O buffers, one word per clock.
- Precharge: By the precharge operation, the sense amplifiers are precharged and the bank of the SDRAM is made to stay in idle state. A row-activation command can be issued when the state of the corresponding bank is idle.

3. History-Based Memory Mode Prediction

The operation mode is controlled by commands generated by the memory controller. The read/write commands with the auto-precharge option change the memory to idle state after completing the corresponding operations as depicted in Fig. 4. As the precharge time (t_{RP}) can be overlapped with burst accesses or data transfer between the memory controller and the processor, the effective latency is the sum of the row-activation time (t_{RCD}) and the column select latency (t_{CL}). The read/write commands without the auto-precharge option maintain the memory in row-active state. If the successive access brings a page hit, the precharge and row-activation operations are not necessary. In this case, the effective latency can be reduced to t_{CL}. If the successive access leads to a page miss, a precharge, a row-activation, and a column select operation have to be performed, increasing the effective latency to $t_{RP}+t_{RCD}+t_{CL}$. Therefore, the memory mode must be controlled to stay in row-active state as long as possible and to minimize the number of page misses.

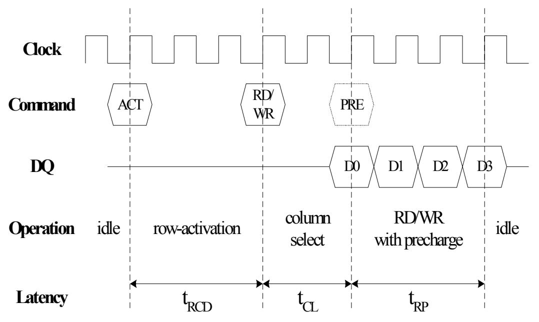

Fig. 4. Read/write operation with the auto-precharge option.

Although the address requested by the processor is random and unknown in advance, the principle of locality of memory reference [7] makes it possible to predict whether the successive access refers to the same row or not. Using the past history of memory references, we predict if the next access causes a page hit and control the memory mode according to the prediction. If the history predicts the successive access to refer to the same row, the memory controller makes the bank remain in row-active state. Otherwise, the bank is made to stay in idle state.

To store the past history of memory accesses, a state machine that can be built with a two-bit saturated up/down counter is used for each row of banks (per-row counter), as shown in Fig. 5. The corresponding state machine is incremented on a page hit and decremented on a page miss after comparing the row and the bank addresses with those of the previous access.

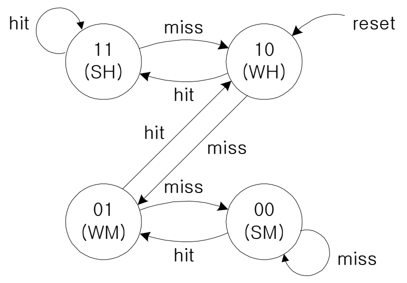

Fig. 5. State machine for storing page hit history information.

For a pending memory access, the memory controller issues a command without the auto-precharge option if the state machine selected by the row and the bank addresses is in strongly hit (SH) or weakly hit (WH) state. If the state machine is in strongly miss (SM) or weakly miss (WM) state, a command with the auto-precharge option is issued. The state transition utilizing the past reference history is depicted in Fig. 6.

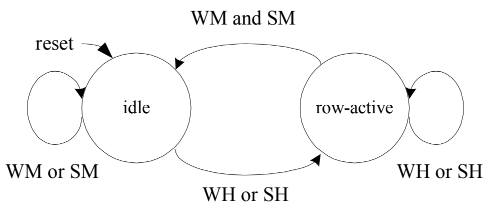

Fig. 6. State transition diagram based on the past reference history.

Although the per-row predictor can accurately reflect the behavior of memory references to the corresponding row, area overhead is considerable. For example, if a memory has a N-bit row address and M banks, $M \bullet 2^N$ two-bit counters are required. To reduce the area overhead while keeping prediction accuracy moderate, one two-bit counter is used for each bank (per-bank counter) instead of each row. As only M state machines are required in this case, significant area reduction is achieved at the loss of small prediction accuracy. Out of M state machines, one is selected by the bank address.

4. Experimental Results

To evaluate the effectiveness of the proposed history-based mode prediction scheme, we measure memory latency by performing trace-driven simulation. Data memory traces of five SPEC92 benchmark programs are obtained on a MIPS R3000 simulator, and used as input vectors in the simulation. As given in the following equation, the total memory latency is calculated by counting all the individual latencies.

$$\text{Latency} = N_{idle} \times (t_{RCD} + t_{CL}) + N_{hit} \times t_{CL} + N_{miss} \times (t_{RP} + t_{RCD} + t_{CL})$$

where N_{idle} is the number of idle states, N_{hit} is the number of page hits in row-active state, and N_{miss} is the number of page misses in row-active state. In the simulation, a SDRAM that has a 13-bit row address, a 9-bit column address, and 4 banks is assumed as shown in Fig. 7. The precharge time (t_{RP}), row-activation time (t_{RCD}), and column select time (t_{CL}) are assumed to be three, three, and two (zero for write operations) cycles, respectively, which are quoted from a commercial SDRAM.

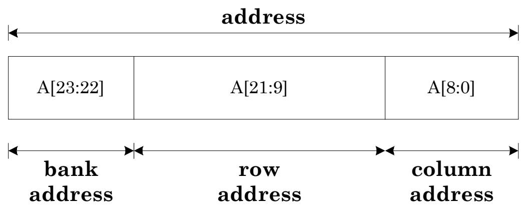

Fig. 7. Memory address allocation.

Table 1 shows the ratio of the number of correct predictions to the number of total references. The history-based scheme using per-row counters shows the highest hit-prediction ratio, and the history-based scheme using per-bank counters predicts more accurately than the previous scheme [6] by 8.1% on the average.

Table 1. Comparison of hit-prediction ratio

Benchmarks	Previous scheme (%)	Per-row counter (%)	Per-bank counter (%)
008.espresso	52.1	70.6	58.4
023.eqntott	73.9	86.9	80.5
056.ear	62.4	72.3	71.0
085.gcc	69.8	81.3	76.8
090.hydro2d	47.6	60.0	59.7
Average	61.2	74.2	69.3

As a result of more accurate prediction, the memory latency is significantly reduced even compared to the previous mode control scheme [6]. The latency results are summarized in Table 2, where we can find that the proposed per-bank prediction scheme outperforms the scheme that always maintains the SDRAM in idle state by 19.0% on the average.

Table 2. Comparison of total latency cycles

Benchmarks	Always idle	Always active	Previous scheme	Per-row counter	Per-bank counter
008.espresso	832900	869614	839083	733312	803023
023.eqntott	992517	808422	735501	645729	689577
056.ear	953780	886802	834698	766691	775580
085.gcc	941097	772872	726195	650187	679917
090.hydro2d	1087215	948270	1035732	942354	944796
Normalized average (%)	100.0	89.2	86.8	77.8	81.0

5. Conclusion

To reduce memory latency, we have proposed a memory control scheme that predicts whether the successive memory access leads to a page hit or not and changes the memory mode according to the prediction. Two-bit state machines are employed to predict the next memory mode based on the history of memory references. Experimental results show that the proposed scheme is effective in reducing the number of row-activations and precharges, thereby improving memory performance.

Acknowledgment

This work was supported (in part) by the Korea Science and Engineering Foundation through the MICROS center at KAIST, Korea and by the Ministry of Science and Technology and the Ministry of Commerce, Industry and Energy through the project System IC 2010.

Reference

[1] M. Kumanoya et al, “Trends in high-speed DRAM architectures,” IEICE Trans. Electronics, 1996, E79-C, (4), pp. 472 – 481.

[2] Y. Takai et al, “250 Mbyte/s synchronous DRAM using a 2-stage-pipelined architecture,” IEEE J. Solid-State Circuits, 1994, SSC-29, (4), pp. 426 – 431.

[3] K. Asheesh et al, “High-Level Synthesis with Synchronous and RAMBUS DRAMs,” Workshop. Synthesis and System Integration of Mixed Technologies, 1998.

[4] T. Takizawa, J. Tajime, and H. Harasaki, “High Performance and Cost Effective Memory Architecture for an HDTV Decoder LSI,” Proc. Int’l. Conf. Acoustics, Speech, Signal Processing, 1999.

[5] H. Kim and I. Park, “Array Address Translation for SDRAM-Based Video Processing Applications,” IEE Electronics Letters, 1999, vol. 35, pp. 1929 – 1931.

[6] S. Miura, K. Ayukawa, and T. Watanabe, “A dynamic-SDRAM-mode-control scheme for low-power systems with a 32-bit RISC CPU,” Proc. Int. Symp. Low-Power Electronics and Design, 2001.

[7] J. Hennessy and D. Patterson, “Fundamentals of computer design,” Chapter 1 of Computer Architecture: A Quantitative Approach, Morgan Kaufmann, 2nd edition, 1996, pp. 1 – 67.