Row-Buffer Reorganization: Simultaneously Improving Performance and Reducing Energy in DRAMs

Nagendra Gulur
Texas Instruments, nagendra@ti.com

R Manikantan
Indian Institute of Science, rmani@csa.iisc.ernet.in

R Govindarajan
Indian Institute of Science, govind@serc.iisc.ernet.in

Mahesh Mehendale
Texas Instruments, m-mehendale@ti.com

***Abstract*—In this paper, based on the temporal and spatial locality characteristics of memory accesses in multicores, we propose a re-organization of the existing single large row-buffer in a DRAM bank into multiple smaller row-buffers. The proposed configuration helps improve the row hit rates and also brings down the energy required for row-activations. The major contribution of this work is proposing such a reorganization without requiring any significant changes to the existing widely accepted DRAM specifications. Our proposed reorganization improves performance by 35.8%, 14.5% and 21.6% in quad, eight and sixteen core workloads along with a 42%, 28% and 31% reduction in DRAM energy. Additionally, we introduce a NeedBasedAllocation scheme for buffer management that shows additional performance improvement.**

***Keywords*-DRAM, Multicore, Row-Buffer Organization**

I. Introduction and Motivation

A typical DRAM read request has to first *Activate* the corresponding row by bringing the row data to the row-buffer. This is followed by a column read/write that reads/writes the selected words from/to the row-buffer. Finally *Precharge* writes back the row-buffer to the appropriate row. *Precharge* is needed even on rows that were only read, since row activation depletes the charge in the corresponding row in the DRAM device. Each bank is equipped with its own row buffer and logic to perform these operations. The width of the row-buffer is the width of the entire row and holds a few KB (say 8 KB) of data. Thus, each Activate/Precharge operation involves charging/discharging a large number of capacitors. If successive requests to a bank are all to the same row in that bank, then only column accesses are needed (termed *row-hits*) and incur lesser latency. On the other hand, a *row-miss* occurs when a new row has to be activated and incurs larger latency (typically 3X the latency of a row-hit).

A. Improving row-buffer hit-rates in multicore workloads

As observed in [1], memory controllers in multi-core architectures observe an interleaved sequence of requests from across all cores thereby destroying the inherent spatial locality (i.e, the row-buffer hit-rate) present in any one core's program. In order to re-capture this lost locality, we investigate the benefit of introducing multiple row-buffers.

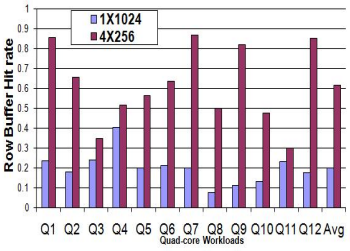

Figure 1. Row-buffer hit-rate improvement

In order to keep the overall row-buffer storage the same, we replace the single large row-buffer with 4 small row-buffers, each $\frac{1}{4}$-th the size of the original row-buffer. Figure 1 shows a sharp increase in row-buffer hit-rate that is now possible by virtue of the 4 row-buffers capturing the inherent spatial locality much better.

B. Energy benefits

Activate and Precharge energy consumption is significantly reduced in the proposed organization owing to two improvements: first, the improved hit-rate directly reduces the number of activate and precharge operations, and second, the activate and precharge operations work on smaller rows, thereby needing to charge/discharge fewer capacitors.

II. Multiple row-buffer organization

We propose to replace the one large row-buffer by four small row-bufers, each one-fourth the size of the original. While a similar scheme is considered in [6] for phase-change memories, our work applies this to the more commonly used DRAMs with very little change to the JEDEC standard. This organization consists of three components, namely (i) Sub-row activation, (ii) Row-buffer selection, and (iii) Row-Buffer allocation. Figure 2 provides an overview of a DRAM bank organized to support multiple small row buffers.

Sub-row activation essentially requires the memory controller to issue additional address bits to the DRAM device(s) to activate one of the four parts of the row and bring it

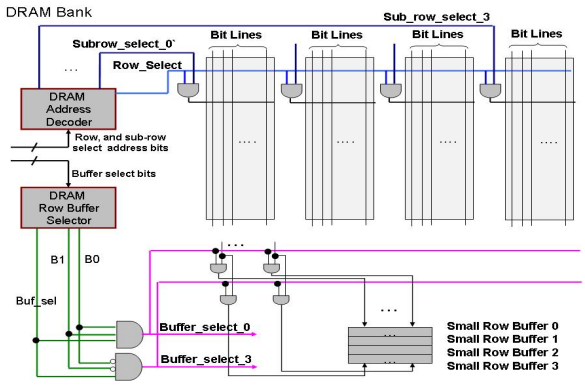

Figure 2. Enhanced DRAM bank to support sub-rows and buffer selection

into a row-buffer. Our implementation of the DRAM core organization is similar to that in [1]. Further, we propose a novel *double address rate* signalling scheme wherein all the address bits (row-select bits, and sub-row select bits) are issued at twice the bus clock leveraging the double-data-rate technology. Row-buffer selection is needed since there are multiple row-buffers and each of the commands (activate/precharge/column-access) needs access to a row-buffer. We let the memory controller specify the row-buffer to use with each DRAM command. This requires an additional 2 bits (for 4 row-buffers) of signalling and we propose to leverage the double-address-rate mechanism to implement this. With these changes, the controller can signal the DRAM to, say, activate the i^{th} part of a row (i can be 0, 1, 2 or 3) and bring that into row-buffer j (j can be 0, 1, 2 or 3). In the DRAM, additional control lines are added to select the specified row-buffer for each operation.

A. Role of the memory controller

A key aspect of the proposed organization is that the memory controller takes on the task of allocating and managing row-buffers (which are physically located in the DRAM). This enables the DRAM logic to stay simple while also providing additional flexibility to implement various buffer allocation policies in the controller. In addition to the default LRU scheme for buffer management, we implemented a NeedBasedAllocation scheme wherein buffers were partitioned out to cores in proportion to their L2 miss rates.

III. Methodology

Using the M5 simulation infrastructure [2], we simulated 4-, 8- and 16-core processors with a 2-level cache hierarchy and multiple memory controllers connected to DDR3 memories. Workloads were formed from SPEC ([3]) CPU 2000 and 2006 benchmarks. Estimated energy savings were computed using Micron's power calculator spreadsheet ([4]).

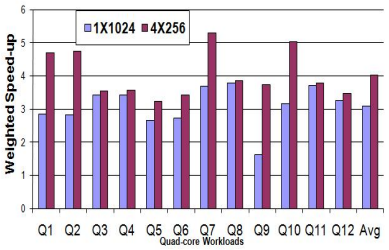

Figure 3. Gain in Weighted speed-up

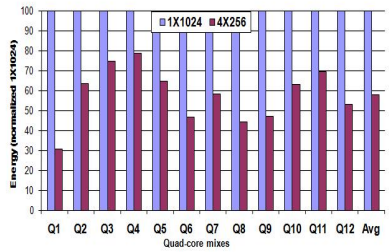

Figure 4. DRAM Energy Savings

IV. Results

4-, 8- and 16-core workloads showed average weighted speed-up improvements of 35.8%, 14.5% and 21.6% respectively. Simultaneously, we obtained energy reduction of 40%, 28% and 31% respectively owing both to high hit-rates as well as smaller rows. Figures 3 and 4 show weighted speed-up and normalized DRAM energy consumption in the baseline and the multiple row-buffer configurations for 4-core workloads. NeedBasedAllocation showed an additional performance improvement of about 1.9%. Area overhead of the additional control logic in the DRAM is estimated to be less than 10% extrapolated from the results in [1] and [5].

References

[1] A. N. Udipi, N. Muralimanohar, N. Chatterjee, R. Balasubramonian., A. Davis, and N. P. Jouppi. *Rethinking DRAM Design and Organization for Energy-Constrained Multi-Cores.* In ISCA 2010.

[2] N. L. Binkert, R. G. Dreslinski, L. R. Hsu, K. T. Lim, A. G. Saidi, and S. K. Reinhardt. *The M5 Simulator: Modeling Networked Systems.* In Micro 2006.

[3] The SPEC Consortium. www.spec.org/consortium/

[4] Micron. *Calculating Memory System Power for DDR3* http://download.micron.com/pdf/technotes/ddr3/TN41_01DDR3%20Power.pdf.

[5] David Wang. *Modern DRAM Memory Systems: Performance Analysis and a High Performance, Power-Constrained DRAM-Scheduling Algorithm.* Phd Thesis. 2005.

[6] B. C. Lee, E. Ipek, O. Mutlu and D. Burger. *Architecting Phase Change Memory as a scalable DRAM alternative.* In ISCA 2009.