An Adaptable Memory System Using Reconfigurable Row DRAM To Improve Performance Of Multi Core For Big Data

Nagi Mekhiel*
Ryerson University, Toronto, ON, CANADA M5B 2K3

Abstract

Multi-core based systems access DRAM using multiple different addresses that could map to different rows in the same bank at the same time, causing row conflicts forcing them to wait to activate one row at a time. We present an adaptable memory using reconfigurable row DRAM that divides rows into many segments and uses special latches to allow many accesses that map to different physical rows to form one adaptable logical row accessed by multi-core as one physical row. The adaptable row accesses different rows in a pipeline fashion by overlapping the long DRAM access time between the different accesses. The results show that the adaptable memory system improves the scalability of multi-core by up to 300% and could gain more from improving processor speed and global cache miss rate and memory-processor bus bandwidth.

Key Words: Reconfigurable architectures; memory systems; DRAM; access patterns; pipelining; scalability of multi-core; big data.

1 Introduction

The processor speed and computing power have continuously increased due to advancements in technology. This increase in processor power depends on delivering data and instructions to the processor from memory at the processor speed. Unfortunately, current memory systems cannot offer the processor its data at the required rate [1,5,11]. Computer performance depends on the memory system to provide multi core with data at high rate. If the memory system fails to deliver the required data rate, it will become the limiting factor to the system performance [1,2].

The cache system has been used to solve this problem by moving parts of data to a fast memory that can match the processor speed. The cache system cannot fully isolate fast processor from the slow main memory. The cache has misses and the processor have to visit the main memory to get data when information is not in cache (misses). These misses represent the portion of processor time that limits the overall system performance improvements. Furthermore, cache performance cannot be improved continuously by changing any of its parameters (size, associativity, speed, ..) and reaches a point of no return with respect to its parameters [2].

DRAM designs offer high bandwidth that depend on using multibank and fast page access mode. In fast page mode, accesses that exist in same DRAM row can be retrieved at a much faster speed (fast page mode). The large data streams in the applications can be mapped to the same row and benefit from this mode. With multibank, more than one active row can supply a processor with fast accesses from different parts of memory [3,9]. However, a multi-core system has many different accesses to different rows at the same time for the same bank.

The contribution of this paper is to reduce the impact of DRAM on the performance and scalability of parallel computing when accessing big data with irregular access patterns. Irregular accesses map to different DRAM rows, cause row conflicts and increase DRAM access time. Performance gain of parallel computing is limited by the slowest portion that cannot be improved. Our method applies parallelism to the slowest DRAM accesses that is related to activating rows by pipelining and overlapping these accesses when mapped to different rows.

2 Backgound

DRAM has been through numerous changes to its design from the basic DRAM architecture to the asynchronous to the fast page mode (FPM) to the extended data-out (EDO) to the burst-mode EDO to the synchronous(SDRAM) [4].

The changes have been relatively minor in terms of their implementation cost and have increased DRAM throughput significantly. Compared to the asynchronous DRAM, FPM simply allows the row to remain open across multiple CAS commands, requiring very little additional circuitry. To this, EDO changes the output drivers to become output latches so that they hold the data valid on the bus for a longer period. To this, BEDO (EDO with Burst) adds an internal counter that drives the address latch, so that the memory controller need not supply a new address to the DRAM on every CAS command if the desired address is simply one-off from the previous CAS command. Thus, in BEDO, the DRAMs column-select circuitry is driven from an internally generated signal, not an externally

*Department of Electrical, Computer and Biomedical Engineering. Email: nmekhiel@ee.ryerson.ca

generated signal: the source of the control signal is close to the circuitry that it controls in space and therefore time, and this makes the timing of the circuits activation more precise. Lastly, SDRAM takes this perspective one step further and drives all internal circuitry (row select, column select, data read-out) by a clock, as opposed to the RAS and CAS strobes [4].

3 Motivations

In general, DRAM architectures have limitations due to the following problems :

- DRAM architectures allow one active row per bank, which limits the ability of multiple accesses from current multi-core chip to access the same bank if the accesses map into different rows in the same bank.
- Only one row at a time can be activated, thus can have one open row per bank at any time. Multiprocessor shares data that map to the same area of memory, and likely in the same bank with different rows, therefore current architectures are not suitable for multi-core.
- Characteristics of software applications are not suitable for DRAM architectures. Program data could map in two different rows in one bank. Each time the program's accesses one row, the other row is forced to be closed because it is in the same bank, when it finishes, it goes back to the closed row and must open it, necessitating it to close the current row that has just been activated. Each time a row is closed, it precharges the bank, then latches the accessed row which wastes time and limit the ability of DRAM to provide fast accesses to processors.
- As performance of single processor is reaching a diminishing return, processor makers are now moving towards multi-core architecture with 8-core or 16-core in a single chip. The performance of the computer is limited by memory bandwidth [7,8].

What we need is a DRAM that is capable of providing a single processor or multiprocessors with high bandwidth using a new architecture that allows DRAM to have more than one location active in the same bank and is not restricted to one active row per bank in DRAM. Therefore, it must have many partial rows active and map them to multiple of physical DRAM rows in one bank, allowing multiple threads to access these active partial rows as if there is a logical row that changes its mapping to become adaptable to processors access patterns and includes many physical rows in a bank. The active row must be constructed based on the processor order of accesses and not by the order of columns in a physical row (adaptable to processor access patterns) [6].

4 The Concept of Reconfigurable Row DRAM "RRDRAM"

DRAM physical row consisting of a number of columns that are fixed and defined on DRAM array. Through this paper, we assume a simple DRAM array of 1024 rows, each row has 1024 columns that could be accessed when the row becomes active. If the processor requests an access that is not in the same active row in the accessed bank, it must activate a new row and adds all the time delays associated with opening a new row.

The processor access is usually for one location that maps to a single column in the accessed row, then a burst mode is used to transfer one cache block from the same active row. With multi-core many different accesses are requested from the memory, these accesses could map to different rows, forcing memory controller to close some active rows and open new rows which increases the access time.

The frequency of accessing DRAM in a multi-core system is multiple times of the single processor. This causes DRAM to open and close many rows very frequently increasing the accesses time and power. The root of the problem is having one fixed physical row activated per bank. A row consists of a number of columns that are used for accesses, and may be only one location of DRAM array that is 1024x1024, could be used in the fast page mode. This is less than .0001% of the DRAM array size. Having one physical row which either can be active or not at a time is not suitable for accesses that come from the multiprocessor and more likely map into many rows at a time.

We should have a memory that is not restricted to a fixed physical row as the current DRAM architecture dictates. A reconfigurable row is a logical row that is not restricted to the DRAM physical row.

If we divide each physical row into segments or sections, that each represents a smaller row of locations, that has for example, only 64 columns each representing a partial row. The size of row segment or section depends on the cache block size and application requirements and implementation. Each row segment or section could be active, therefore creating a logical row that is adaptable for physical rows when changing [6].

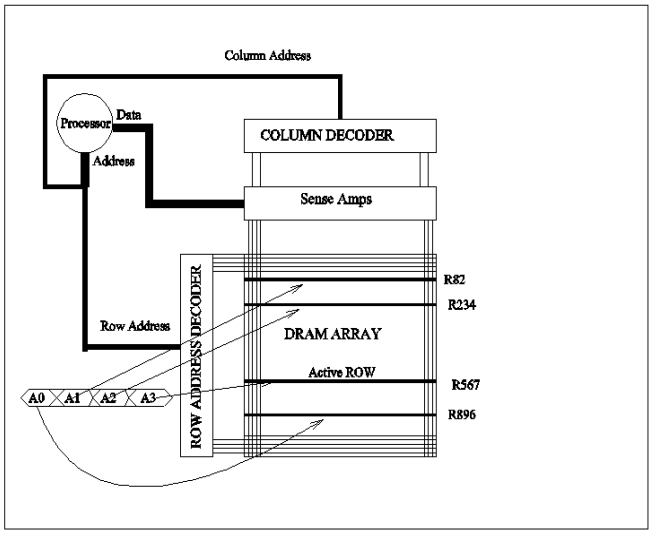

Figure 1: DRAM organization

Figure 1, shows a conventional DRAM organization being

accessed by a processor to deliver data from four locations in the DRAM array: A0, A1, A2, and A3. Assuming that A0 maps to row Row896, A1 maps to Row82, A2 maps to Row234 and A3 maps to Row567. This conventional DRAM system must precharge and activate R896 (Row896) to supply data for A0, then precharge and activate R82 and wait to supply data for A1, followed by the same operations for A2 and A3. This requires that total access time is four of random accesses = 4 *(precharge + ROW access + Column access + Transfer). In the reconfigurable row system, it will only use one active logical row and burst the four accesses from the latched data on sense amps, therefore total time of four accesses is one random access plus three fast page accesses= Precharge + Row access + Column access + 4 *transfer.

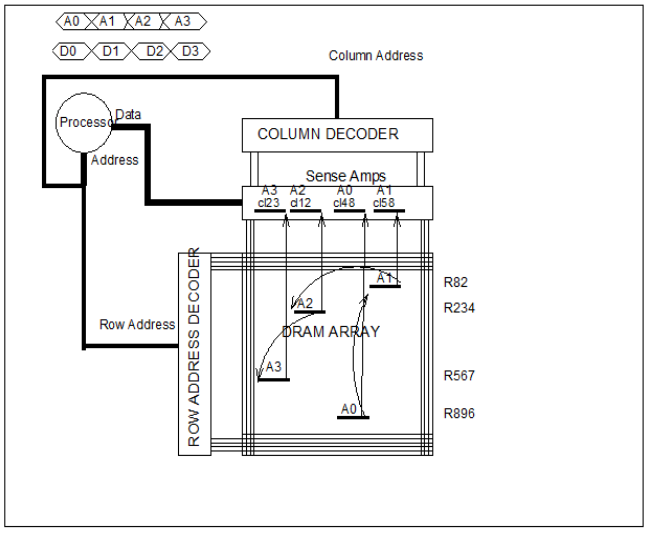

Figure 2: Mapping accesses in adaptable row DRAM

Figure 2 shows the adaptable row system in accessing the four addresses A0, A1, A2 , A3 for the above example. Although these addresses map to different four rows R896, R82, R234 and R567, the logical row is formed to contain only four partial rows. The other row segments are not included in the activated logical row, therefore leaving 12 partial rows available and precharged for the next group of accesses. It uses only a portion of the total sense amps leaving the rest available for next access. It should also be noted that in the reconfigurable row, because a large portion of DRAM array will be ready and precharged, latching new rows, and columns for the next accesses could be completely overlapped and interleaved with data transfer to the processor.

The latching of row segments allows the row address to change and be decoded, while accessing present row segments. This is a pipelining of accessing DRAM in which segments are used and accessed in parallel by overlapping their delay time of precharging, latching, decoding and access as explained below in system operation.

5 The Reconfigurable ROW DRAM "RRDRAM" System

5.1 The Basic System

The basic reconfigurable row for DRAM maintains the same interface to the outside system. It uses the same DRAM core of an array of columns and rows and the same storage elements of single capacitor. The main difference is in using multiple row segment latches, with each latch having a number of flip flops equal to the number of rows in the DRAM array. For example, if the DRAM array is 1024x1024, then these row segment latches have 1024 flip-flops each. Each physical row in DRAM array is divided into multiple segments, each segment is connected to one output of the corresponding segment row latch [6].

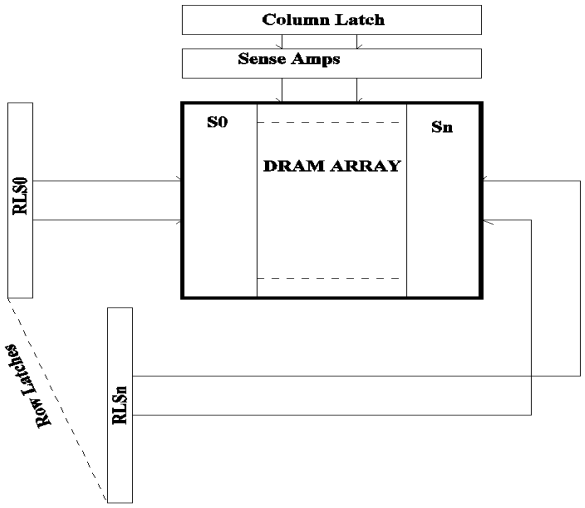

Figure 3: Block diagram for reconfigurable row DRAM

Figure 3 shows a block diagram for a reconfigurable row for DRAM. Multiple row segment latches. RLSn are used to latch the decoded row of an accessed segment, each time there is a request to access DRAM.

The outputs of these RLS latches control the word lines of a specific row segment and each output is connected to 1 word line of a segment. For 1024x1024 and 16 segments, there are 1024 outputs from each of RLS and each output is connected to 64 of word lines to activate the segment for 64 cells.

Figure 4 shows a more detailed schematic diagram to the RRDRAM for each segment of the different rows being latched by its corresponding segment row latch using k flip-flop, where k is the number of rows in DRAM array and n is the number of segments in each row, thus requiring nxk Flip-Flops in total per the DRAM array. The flip-flops use RlClk to activate one flip-flop that has its row = 1. ROW signals are activated by the row decoder when a valid access is requested. The column decoder shown is used to generate the proper RlClk for one segment at a time. RlClk is generated to latch 1 at flip flop for the segment in the intersection of accessed row and accessed column in DRAM array.

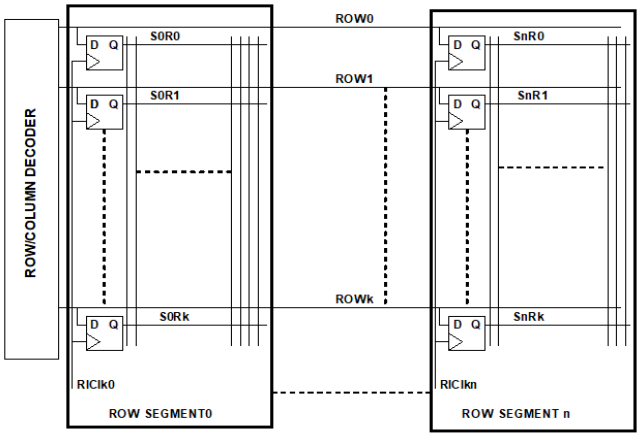

Figure 4: Block diagram for row segments of reconfigurable row DRAM

6 System Operation

6.1 Pipelined Operation

Our RRDRAM uses the same known DRAM technology, therefore it requires the same timing to access data from one location. In conventional DRAM the following are the basic operations that must be performed for a DRAM operation (read or write):- 1-Precharge operation wait for Tpr then 2-Latch a valid row address wait for Trcd then 3-Latch a valid column wait for Tcac then 4-access data In RRDRAM, the use of different latches for row address, column address and latches for the decoded rows in a segment (RL) and the latch for multiple decoded columns (Column decoder Latch) allows the system to overlap multiple accesses to different DRAM rows similar to processor pipelining of instruction execution.

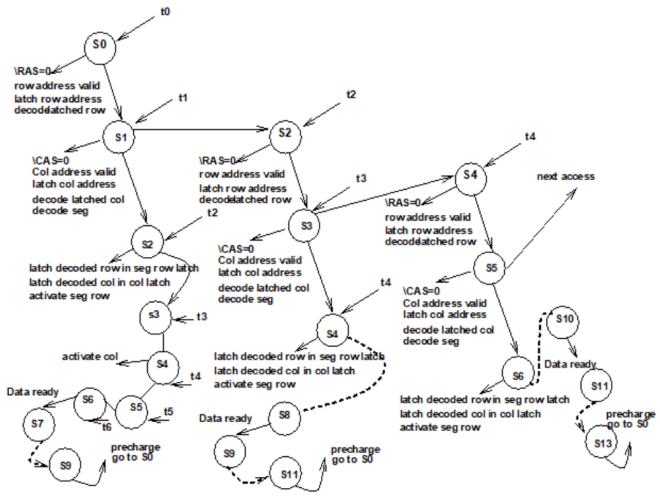

Figure 5: State diagram for pipelined operation in reconfigurable row DRAM

Figure 5 shows a state diagram for the pipelined operation of RRDRAM using the system of latches given above. The following are the different states for multiple DRAM accesses to different rows in RRDRAM:

- S0 (time t0): The row address of the first access is applied to the DRAM multiplexed address and is latched by row address latch at time t0. The row decoder immediately starts to decode this row address.
- S1 (time t1): The memory controller applies the column address of first access to DRAM multiplexed bus and is latched by the column address latch at time t1. The column decoder starts immediately decoding this column address. The segment decoder decodes a portion of that address corresponding to the segment selection and makes one output active that corresponds to the accessed row segment.
- S2 (time t2) and S0 of second access: The system starts the second access by applying the row address of the second access similar to the first access in S0. It also continues with the first access by latching the decoded row from the row decoder in the corresponding RL latch by one of RlClk generated from the segment decoder circuit. It also latches the decoded column for the first access in the column decoder latch and keeps previous accessed columns in the active latch by the feedback.
- S3 (time t3) and S1 of second access: The first access is waiting for Trcd and the accessed row is being active and applied to the word line of the selected row segment from the corresponding RL latch. The second access is continuing in S1 similar to access 1 to latch its column address. It is important to note that the first access column has been captured and stored in the column decoder latch in S2.
- S4 (time t4), S2 of second access and S0 of third access: The first access asserts its active column to the bit line to access the data. The second access latches the decoded row in RL latch similar to first access in S2 and also latches the decoded column. The third access starts in applying a row address to the multiplexed bus similar to first access in S0.
- S5 (time t5) ,S3 of second access and S1 of third access: The first access waits for Tcac to get data. The second access waits for Trcd and accessed row is applied to word line as in S3 for first access. Third access is in S1 to latch column address and decode segment.
- S6 (time t6), S4 of second access and S2 of third access: First access has its data ready and valid. The second access asserts its active column to the bit line to access the data. The third access is latching the decoded row from the row decoder in the corresponding RL latch by one of RlClk generated from the segment decoder circuit. Also it latches the decoded column for the first access in the column decoder latch and keeps previous accessed columns in the active latch by the feedback.

At S6 data is ready for access 1, after two cycles data is ready for access 2, after two cycles data is ready for access 3. When

access 1 delivers its data at S6, RRDRAM starts immediately precharging the segment of access1. In this way the precharge time is hidden and pipelined with other accesses. The second and third accesses follow the same method making all accesses pipelined in precharge, latching, decoding, access time and data delivery.

6.2 Timing for System Operation

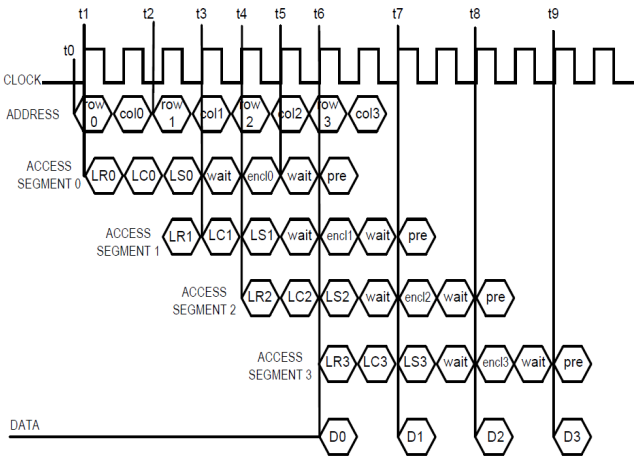

Figure 6: The timing of pipelined operation in RRDRAM

Figure 6 shows the timing of the RRDRAM operation given above with the state diagram. The address bus supplies row and column address for each access through the use of a multiplexer in the memory controller as in any conventional DRAM. The next access address is supplied to the address bus without the waiting for data to be accessed from DRAM for the first access. The memory controller keeps supplying the RRDRAM address bus by new accesses every two clock cycles. The bus speed could be increased compared to conventional DRAM because of the use of latches in RRDRAM. Each address is latched immediately and stored in the latch to be decoded.

The access to first address starts by latching the row in LR0 assuming that LR0 is used for access 0. LC0 is used to latch column address for first access 0. The partial row is then activated through applying the output of its RL latch to the word line of the segment in LS0. The system waits, then uses the output of column latch to activate bit line and access the data after waiting for one cycle as shown in data bus as D0. After each segment delivers its data it goes through precharge individually without the need to precharge the whole bank as in conventional DRAM.

Other accesses, access 1 and access 2 time is overlapped with each other in a pipeline fashion to deliver data at a rate of 1 access every two cycles regardless of row number, therefore more than one physical row could be accessed in parallel.

7 Performance Evaluation of RRDRAM

7.1 Conventional Single Processor Model

$$Ts = Tp \times Ni + M \times Ni \times (Ta + Tf) \quad (1)$$

Ts is the execution time for single processor using conventional DRAM
Tp= processor cycle time and could be less than 1 for superscalar
Ni=number of instructions in an application
M=Cache Miss rate
Ta=main memory access time to activate DRAM Row and access first location
Tf=time to transfer 1 block of cache using memory bus

7.2 Conventional Multiprocessor Model Using DRAM

$$Tm = Tp \times Ni + Np \times M \times Ni \times (Ta + Tf) \quad (2)$$

Tm is the time to execute multiple parallel processes on multiprocessor system using conventional DRAM where Np is number of processors. The total number of misses to external shared main memory will be proportional to Np as they have to be serviced in a serial fashion from one shared memory.

The processor execution time is overlapped among multiple processors working in parallel, but accesses to shared memory must be serialized.

$$Scalability = Np \times Ts \div Tm \quad (3)$$

The scalability is calculated by dividing the total time that a single processor spends to execute same Np number of processes in serial fashion by the time that multiprocessor takes to execute same number of processes in parallel as given by Equation (2).

7.3 Conventional Multiprocessor Model Using RRDRAM

$$Tm(R) = Tp \times Ni + M \times Ni \times (Ta + Np \times Tf) \quad (4)$$

Tm(R) is the time to execute multiple processes on a multiprocessor system with RRDRAM where Np is the number of processors.

$$Scalability = Np \times Ts \div Tm(R) \quad (5)$$

The total time to access the external main memory RRDRAM will be the first access time Ta to DRAM then overlap the rest of access times with Np transfer times to fill Np caches.

The scalability is calculated by dividing the total time that a single processor spends to execute same Np number of processes in serial fashion by the time that multiprocessor takes to execute the same number of processes in parallel as given by Equation (4).

7.4 Parameters of the System

Assuming a SPEC CPU2000 [10] benchmark application in a processor running at 3.3 GHz.
IPC = 2 from using superscalar.
Tp = .15n average.
Ni= depends on the workload and changes from .774 to 14.6 billion instructions for SPEC CPU2000. We assume Ni = 1 billion.
M depends on the application and the cache. It can vary from .0002 to .001, We assume M=.0006 for SPEC CPU2000 , 64 KB L1 caches and is the same for all systems.
We assume Ta= 30 ns for a typical DRAM, this is the access time to get first data element.
We can calculate the cache block transfer time as: similar to Intel multi core that has about 10 GB/S (1.33 GHz and bus = 8 B) Tf = 128/10 = 12.8 ns.

7.5 Scalability of DRAM and RRDRAM Systems Results

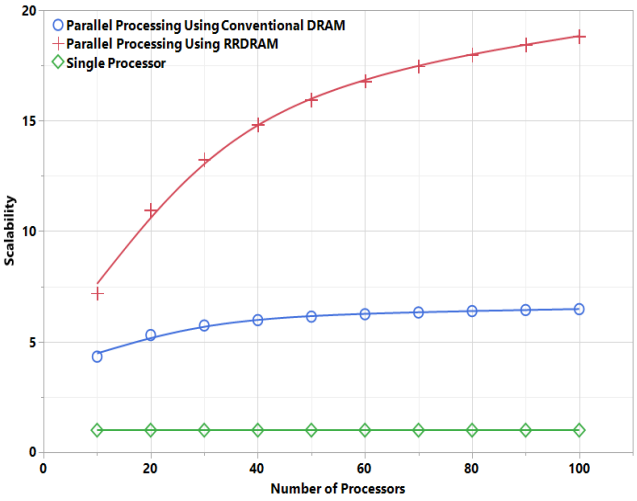

Figure 7: Scalability of multiprocessor systems using DRAM versus RRDRAM

Figure 7 shows the results of multiprocessor scalability when using conventional DRAM and RRDRAM versus the number of parallel processors.

The systems parameters are: processor speed is 3.3 GHz, DRAM and RRDRAM access time Ta is 30 ns, Global Cache miss rate is .0006 and memory bus bandwidth is 10 GB/s.
The effect of DRAM and RRDRAM is significant to the performance and scalability of multiprocessors limiting its scalability for a large number of processors. This shows that the memory gap is still an issue that should be dealt with for parallel computing. This confirms the Amdahl's law that the portion of time that cannot be improved remains the system bottleneck to overall performance gain.

The multiprocessor using RRDRAM significantly improves scalability of the system compared to the system using conventional DRAM up to 300%.
Scalability of multiprocessor using DRAM reaches its maximum at a low number of processors while the multiprocessor using RRDRAM reaches its maximum performance at a higher number of processors.

7.6 Scalability of DRAM versus RRDRAM Systems Using different Processor Speed

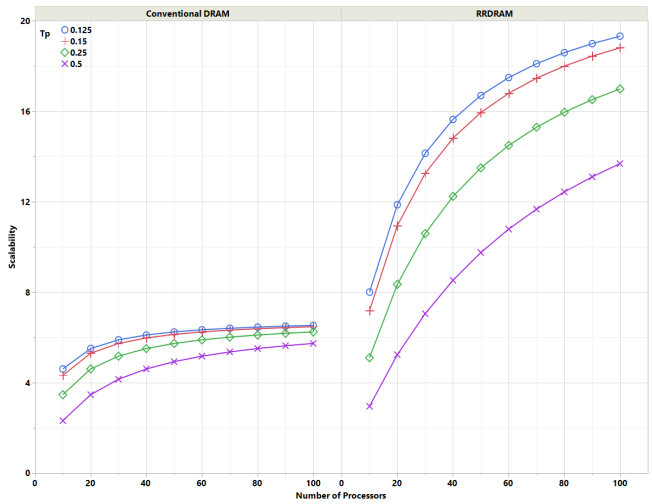

Figure 8: Scalability of multiprocessor systems using DRAM versus RRDRAM with different processor speed

Figure 8 shows the scalability of both systems when changing processor speed from 1 GHz to 4 GHz. The multiprocessor system using conventional DRAM gains much less than the system using RRDRAM for improving processor speed. The system using DRAM only improves its scalability by 20% for four times increase in processor speed. The system using RRDRAM improves by 40% for same processor speed increase. This indicates that RRDRAM gains more from increasing processor speed and technology could use both improvements in the number of transistors and improvements speed of transistors to continue in improving the gain of multiprocessor scalability.

The results also show that RRDRAM system using a faster processor gives the same scalability when using 20 faster processors as the same system using 70 processors if using slower processors. The DRAM based system with slower processor speed needs to use 100 processors to get the same scalability of 20 faster processors. This indicates that the system using RRDRAM gains more from increasing processor speed than the system using DRAM.

7.7 Scalability of DRAM versus RRDRAM Systems Using different Miss Rate

Figure 9 shows the scalability of both systems when changing global cache miss rate. The miss rate changes by eight times from .0003 to .0024.

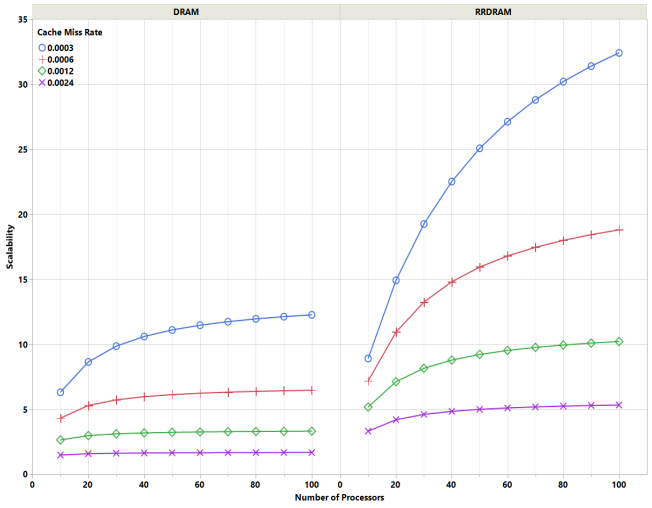

Figure 9: Scalability of multiprocessor systems using DRAM versus RRDRAM with different miss rate

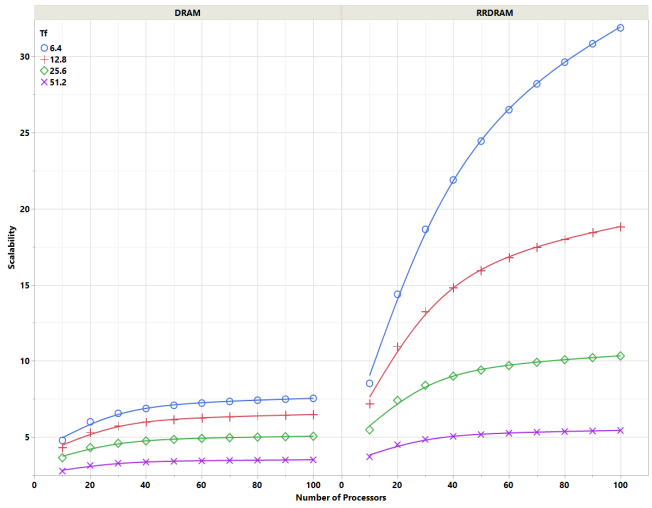

Figure 10: Scalability of multiprocessor systems using DRAM versus RRDRAM with different transfer time

The multiprocessor system using conventional DRAM improves its scalability by 200% from 6 to 12 when miss rate improves by eight times.

The multiprocessor system using RRDRAM improves its scalability by 180% from 18 to 32 when miss rate improves by eight times.

This indicates that both systems are more dependent on the performance of their DRAM and RRDRAM memory and that the memory gap between processor speed and memory speed is still a bottleneck for the performance of computer systems.

The result shows that if global miss rate is high, the scalability of both systems is drastically reduced, for the system with DRAM its scalability decreases from 12 to 1 by 1200%, when miss rate increases by eight times. The scalability of the system using RRDRAM deceases from 32 to 5 by 600% and is less dependent on miss rate.

7.8 Scalability of DRAM versus RRDRAM Systems Using different Memory Bus Bandwidth

Figure 10 shows the scalability of both systems when using different memory to processor bandwidth. Bus bandwidth changes from 2.5 GB/S to 20 GB/S by eight times. This causes transfer time Tf of a 8 B block to the cache from 6.4 ns to 51.2 ns.

The system using RRDRAM improves its scalability from 5 to 32 by about 600%. The system using DRAM improves its scalability from 2 to 7 by 350%. This indicates that RRDRAM system gains more from improving the memory bus bandwidth.

8 Conclusions

Multi-core based system performance depends on using suitable memory system able to provide it with low latency and high bandwidth. The conventional DRAM with one row per bank is not suitable to handle the requirements of a multi core that access memory with different addresses simultaneously. High bandwidth obtained by fast interface cannot help the performance of a multi core system when using conventional DRAM.

Our proposed RRDRAM is able to offer multi-core better scalability for data with irregular access patterns that have different addresses mapped to different physical rows and accessed in a pipelined fashion as if it is in one physical row. The scalability of the system using RRDRAM is multiple fold that of the same system using DRAM. Using RRDRAM also enables multiprocessor system for benefiting from faster processors and higher bandwidth of memory bus. The speed gap of processor memory becomes less severe when the system uses RRDRAM.

References

[1] D. Burger, J. R. Goodman, and A. Kagi, Memory Bandwidth of Future Microprocessors, *Proc. 23rd Ann. Int'l Symp. On Computer Architecture*, ACM Press, New York, 1996, pp. 78-89.

[2] J. Hennessy, and D.A. Patterson, *Computer Architecture: A Quantitative Approach*, Morgan Kaufmann Publishers, Inc, San Francisco, CA, 2003.

[3] Kohji Hosokawa, Toshio Sunaga, and Shinpei Watanabe, DRAM with Multiple Virtual Bank Architecture for Random Row Access, US Patent No. 6,925,028 B2 Aug. 2, 2005

[4] Bruce L. Jacob, Synchronous DRAM Architectures, Organizations, and Alternative Technologies 2002-12-10, Electrical & Computer Engineering Dept. University of Maryland College Park, MD 20742 http://www.ece.umd.edu/ blj/

[5] Sally A. McKee and Robert H. Klenke, Smarter Memory: Improving Bandwidth for streamed References, *IEEE Computer* , pp 54-63, July 1998.

[6] Nagi Mekhiel, Reconfigurable Row DRAM, US Patent 9,734,889 B2, Aug 15, 2017.

[7] Samuel Moore, Multicore Is Bad News For Supercomputers, *IEEE* Spectrum, Nov 2008.

[8] R. Murphy, On the Effects of Memory Latency and Bandwidth on Supercomputer Application Performance Workload Characterization, 2007. IISWC 2007. IEEE 10th International Symposium, pp.35-43, Sept 27-29, 2007.

[9] RAMBUS XDR DRAM, www.rambus.com, 2007 .

[10] SPEC CPU2000 Benchmark https://www.spec.org/cpu2000

[11] Wm. A. Wulf, and Sally A. McKee, Hiting the Memory Wall: Implication of the Obvious, *ACM Computer Architecture News* Vol. 23, No. 1, pp. 2024, 1995..

Nagi Mekhiel is a Professor in the Department of Electrical, Computer and Biomedical Engineering, Ryerson University, Toronto. His research interests are computer architecture, parallel processing, high performance memory systems, advanced processors, VLSI, and performance evaluation of computer systems. He holds many U.S. and World patents in memory and multiprocessors. He is conducting research to solve the fundamental problems facing computer industry, including scalability of parallel processors, and processor/memory speed gap.