DRAMSim2 - Quick Reference Guide

I. ABSTRACT

This document briefly decribes DRAMSim2, the differences between DRAMSim and DRAMSim2, and the rationale used to justify the development of DRAMSim2.

II. WHY DRAMSIM2 ?

DRAMSim began as a set of code written to simulate DRAM device operations on a phase-by-phase level. That is, each DRAM command goes through different phases of operation, and no two DRAM commands can use the same shared resource at the same time. DRAMsim's modeling of pipelining DRAM device thus depends on accurate modelling of the phase operation and resource contention of the shared resources by DRAM commands. DRAMsim then simulated the usage and release of the resources on the DRAM devices on a cycle-by-cycle basis. Figure 1

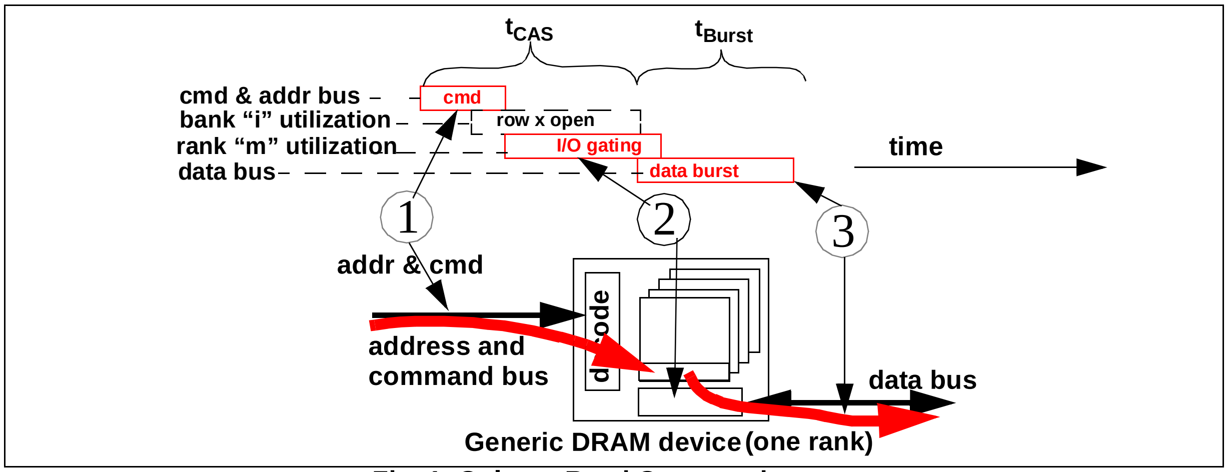

Fig. 1: Column Read Command

illustrates the progression of a column read command and shows that the column read command goes through three basic phases. Any other command that follows the column read command cannot make use of the command bus, the I/O gating resources or the data bus at the same time as the column read command. The belief was that with proper simulation at the resource level, DRAM command interactions would reveal themselves naturally. Unfortunately, modification of the DRAM simulator required the programmer to have a deep and fundamental understanding of DRAM device operations. Otherwise, the addition of a new type of DRAM devices (DDR2, DDR3, FCRAM etc.) became difficult tasks as timing had to be validated across different system configurations and scheduling algorithms. The primary motivation of DRAMSim2 is to move away from the phase-by-phase simulation of the DRAM command in DRAMsim, and move to a protocol-based simulation of DRAM commands. We believe that the use of protocol-table driven simulation ensures that DRAM memory system timing constraints are accurate by design, and simulation on a command-by-command basis enables far faster simulation speeds over that of cycle-by-cycle simulation.

III. WHAT ARE THE DIFFERENCES?

At the present time, DRAMsim is a fully featured DRAM memory system simulator, including Fully Buffered DIMM memory system simulation, DRAM power computation as well as a host of statistics collection mechanisms. In contrast, DRAMsim2 is at this time a minimal simulation core that has none of these features. The current Plan of Record (POR) is to re-integrate all of the supported features currently present in DRAMsim into DRAMsim2. As a result, we recommend that those that are interested in processor-memory system interaction continue to use DRAMsim, while programmers that are interested in developing highly accurate and fast DRAM memory system simulations, testing different assumptions of scheduling algorithms and support of new type of DRAM devices such as RLDRAM and XDR should collaborate closely with the DRAMsim2 code.

Aside from the differences in how DRAM commands are simulated, some other minor differences exist in the structural assumption of DRAMsim and DRAMsim2. Those differences and the rationale in changing the structure are described in the following sections.

A. STRUCTURAL ASSUMPTIONS OF THE DRAM CONTROLLER

Figure 2 illustrates the basic structural assumptions of a single channel memory controller in DRAMSim and

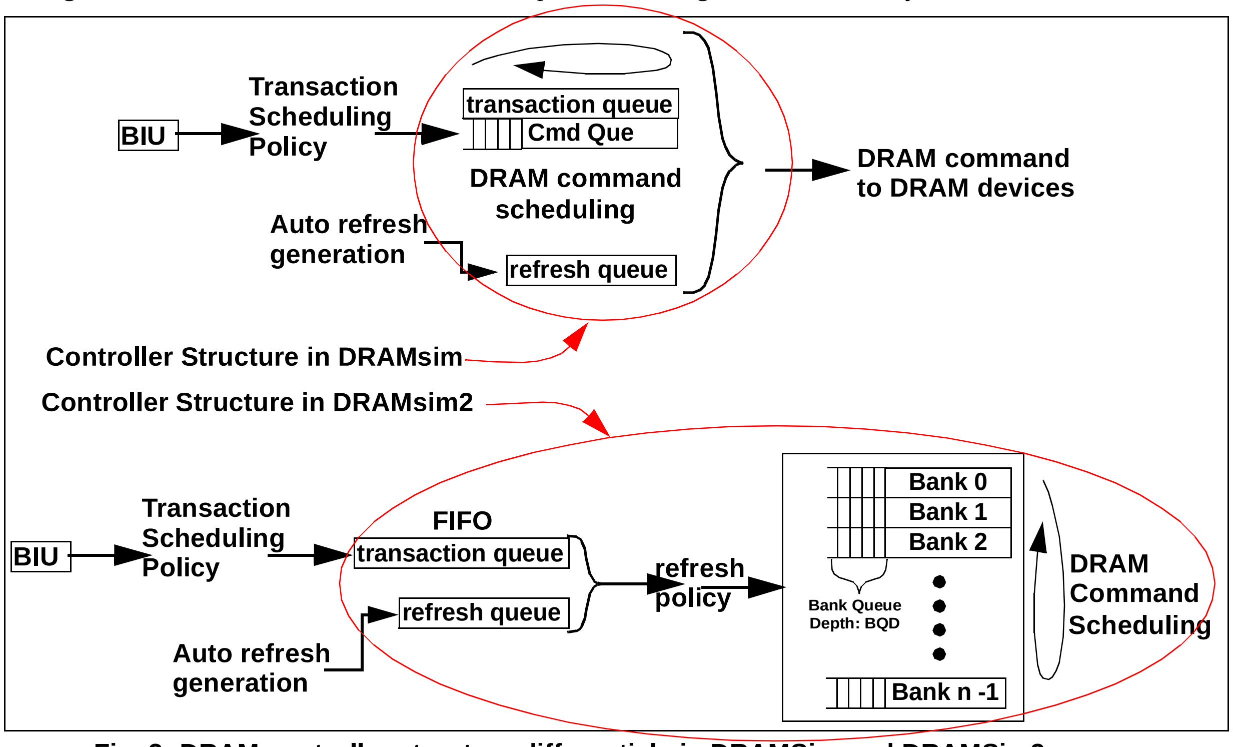

Fig. 2: DRAM controller structure differentials in DRAMSim and DRAMSim2

DRAMsim2. Figure 2 illustrates that in DRAMSim, transactions are translated into DRAM commands, then placed into a unified queue that contained each transaction and a sequence of DRAM commands. In contrast, DRAMSim2 uses separate queues to hold DRAM commands destined for each bank. (One variable that can be easily adjusted in DRAMSim2 is the depth of the per-bank queues.) The default assumption in DRAMSim2 is that each per-bank queue holds all of the DRAM commands destined for each bank, and aside from precharge commands that can be deferred in favor of column access commands in open-page memory systems, DRAM commands are executed in FIFO order within each per-bank queue. In this manner, DRAMSim2 allows for the implementation of very aggressive memory controller designs that intelligently selects between DRAM commands present at the top of each per-bank queue without having to worry about read-or-write ordering issues. That is, since reads and writes destined for a given bank are simulated in-order, read commands that follows write commands to the same address location cannot be erroneously scheduled ahead of the write command. Moreover, the separate FIFO queues ensure that the likelihood of an erroneously coded algorithm leaving out critical row access and precharge comands is greatly reduced.

B. BASIC TIMING PARAMETERS

In any DRAM memory-access protocol, a set of timing parameters is used to characterize various command durations and latencies. The timing parameters used in the simulation framework are summarized in Table 1. This set of timing parameters is used to model SDRAM, DDR SDRAM, DDR2 SDRAM and DDR3 SDRAM devices in both DRAMSim and DRAMsim2.

Parameter	Description
t_{AL}	**A**dded **L**atency to column accesses, used in DDRx SDRAM devices for Posted CAS commands.
t_{BURST}	Data **burst** duration. The time period that data burst occupies on the data bus. Typically 4 or 8 beats of data. In DDR SDRAM, 4 beats of data occupies 2 full clock cycles
t_{CAS}	**C**olumn **A**ccess **S**trobe latency. Time interval between column access command and the start of data return by DRAM device(s). Also known as t_{CL}.
t_{CMD}	**C**o**m**man**d** transport duration. Time period that a command occupies on the command bus as it is transported from the DRAM controller to the DRAM devices
t_{CWD}	**C**olumn **W**rite **D**elay. Time interval between issuance of column-write command and placement of data on data bus by the DRAM controller.
t_{FAW}	**F**our (row) bank **A**ctivation **W**indow. A rolling time frame in which a maximum of four bank activation can be engaged. Limits peak current profile in DDR2 and DDR3 devices with more than 4 banks.
$t_{INT\text{-}BURST}$	**Int**ernal **burst** duration. The internal burst length of the DRAM device. Multiple internal bursts are often used to form one longer continuous burst for column read commands. $t_{int\text{-}burst}$ is 2 for DDR, 4 for DDR2.
t_{RAS}	**R**ow **A**ccess **S**trobe. Time interval between row access command and data restoration in DRAM array. DRAM bank cannot be precharged until at least t_{RAS} time after the previous bank activation.
t_{RC}	**R**ow **C**ycle. Time interval between accesses to different rows in a bank. $t_{RC} = t_{RAS} + t_{RP}$
t_{RCD}	**R**ow to **C**olumn command **D**elay. Time interval between row access and data ready at sense amplifiers.
t_{RFC}	**R**e**f**resh **C**ycle Time. Time interval between Refresh and Activation command
t_{RP}	**R**ow **P**recharge. Time interval that it takes for a DRAM array to be precharged for another row access.
t_{RRD}	**R**ow activation to **R**ow activation **D**elay. Minimum time interval between two row activation commands to same DRAM device. Limits peak current profile.
t_{RTP}	**R**ead **t**o **P**recharge. Time interval between a read and a precharge command. Can be approximated by $t_{CAS} - t_{CMD}$
t_{RTRS}	**R**ank **t**o **r**ank **s**witching time. Used in DDR and DDR2 SDRAM memory systems. Not used in SDRAM or Direct RDRAM memory systems. 1 full cycle in DDR SDRAM
t_{WR}	**W**rite **R**ecovery time. Minimum time interval between end of write data burst and the start of a precharge command. Allows sense amplifiers to restore data to cells
t_{WTR}	**W**rite **T**o **R**ead delay time. Minimum time interval between end of write data burst and the start of a column-read command. Allows I/O gating to overdrive sense amplifiers before read command starts

Table 1: Summary of timing parameters used in generic DRAM access protocol

prev	next	rank	bank	Minimum Timing	Notes
A	A	s	s	t_{RC}	
A	A	s	d	t_{RRD}	Plus t_{FAW} for 5th RAS to same rank
P	A	s	d	t_{RP}	
F	A	s	s	t_{RFC}	
A	R	s	s	$t_{RCD} - t_{AL}$	t_{AL} = 0 without posted CAS command
R	R	s	a	t_{BURST}	
R	R	d	a	$t_{BURST} + t_{RTRS}$	
W	R	s	a	$t_{CWD} + t_{BURST} + t_{WTR}$	
W	R	d	a	$t_{CWD} + t_{BURST} + t_{RTRS} - t_{CAS}$	
A	W	s	s	$t_{RCD} - t_{AL}$	
R	W	a	a	$t_{CAS} + t_{BURST} + t_{RTRS} - t_{CWD}$	
W	W	a	a	t_{BURST}	
A	P	s	s	t_{RAS}	
R	P	s	s	$t_{AL} + t_{BURST} + t_{RTP} - t_{INT\text{-}BURST}$	
W	P	s	s	$t_{AL} + t_{CWD} + t_{BURST} + t_{WR}$	
F	F	s	a	t_{RFC}	
P	F	s	a	t_{RP}	

Legend

F = re**F**resh	s = **s**ame
A = row **A**ccess	d = **d**ifferent
R = column-**R**ead	a = **a**ny
W = column-**W**rite	
RP = **R**ead-and-**P**rec	
WP = **W**rite-and-**P**rec	
P = **P**recharge	

Table 2: Unified SDRAM, DDR, DDR2 and DDR3 Protocol Table

C. PROTOCOL TABLE

The major innovation in DRAMSim2 is the use of the protocol table. as seen in Table 2. The use of the cycle-by-cycle simulation technique in DRAMsim dictated that DRAM command timing must be separately validated after the phase-by-phase relationship for each DRAM command is coded into the simulator. The use of the protocol table means that DRAM command timing is correct-by-design. That is, the timing equations used in Table 2 can be independently verfied against timing specifications given in various DRAM device datasheets, and the literal implemetation of the protocol table ensures that the timing relationships as specified in DRAM device datasheets are observed. Table 2 summarizes the minimum timing equations for basic DRAM command interactions between the row access, column-read, column-write, column -read-and-precharge, column-write-and-precharge, precharge, and refresh commands. The DRAM command interactions described in Table 2 is fully applicable to all commodity DRAM memory systems: SDRAM, DDR SDRAM, DDR2 SDRAM as well as DDR3 SDRAM memory systems. Table 2 is organized by each DRAM command, the possible commands that can precede each command, and the respective minimum timing constraints that must be met in between each command combination before the second command can be issued. The timing equations contained in Table 2 is not fully illustrated for all command combinations in this manual. The complete protocol description will be published in a book to be released in 2007. For validation of the timing equations, please refer to any available SDRAM, DDR SDRAM or DDR2 SDRAM device datasheet. If you have more questions, please contact the authors of this DRAM memory system simulator.

IV. CURRENT STATE OF DRAMSIM2 AND PLAN OF DEVELOPMENT

DRAMSim2 is currently a simple standalone code, and efforts have been devoted to ensure that the implementation of the protocol table and the multiple per-bank queue structures are bullet-proof. Code reviews have been conducted to ensure that the code is read-able and easily portable to different simulation environments. The plan is to seamlessly replace the core of DRAMSim2 with DRAMsim, retaining all of the features present in DRAMSim such as FB-DIMM support and DRAM Power simulations.

DRAMSim2 simulations are based on simulation of discrete DRAM commands that are essentially discrete events. However, DRAMsim2 is not fully event-driven at this time. As a result, the primary use of DRAMSim2 at this time is to examine DRAM memory system throughput with fully saturated queues. Work is currently on-going to convert DRAMsim2 to an event-driven model, then to couple DRAMSim2 to various processor/system simulators. (AlphaSim, MASE, GEMS)

Figure 3 shows the type of study that DRAMSim2 is capable of doing at this time. Figure 3 shows that the the use

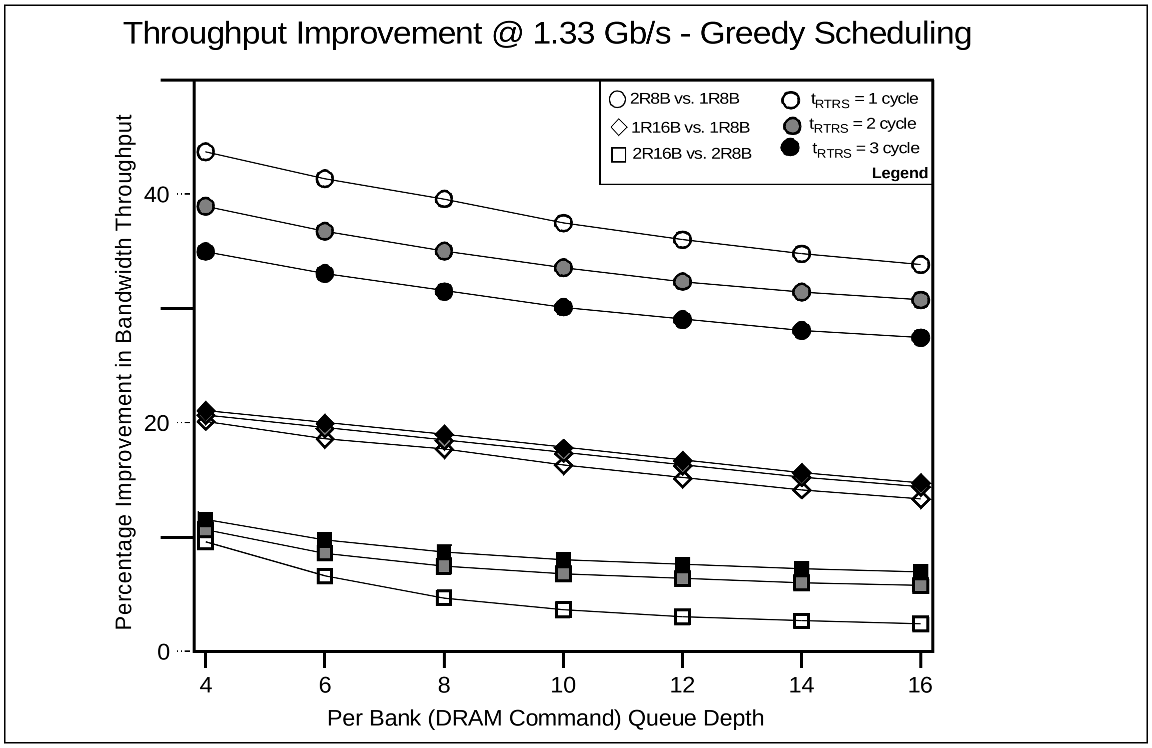

Fig. 3: DRAM Bandwidth throughput study with DRAMSim2

of two ranks of memory can improve bandwidth throughput by as much as forty-plus precent over that of a single rank of memory at 1.33 Gb/s (DDR3). However, the bandwidth improvement greatly lessens in the case of deeper queues and larger rank-to-rank switching penalties.

DRAMSim2 Options

This section has been included to provide a summary of available options currently supported in DRAMSim2.

-config:channel *?channel_count?*

Specifies the number of channels in the DRAM memory system. Overrides value specified in .spd file.

-config:rank *?rank_count?*

Specifies the number of ranks per channel in the DRAM memory system. Overrides value specified in .spd file.

-config:bank *?bank_count?*

Specifies the number of banks per rank in the DRAM memory system. Overrides value specified in .spd file.

-config:ordering_algorithm *?algorithm?*

Specifies how to select between different per-bank queues. The basic algorithms are ***strict_order****,* ***rank_round_robin*** *and* ***bank_round_robin****.*

-config:queue_depth *?queue_depth?*

Specifies the depth of the per-bank DRAM command queues.

-config:read_percentage *?read_percentage?*

Specifies the percentage of read requests in a random input stream.

-debug

This switch turns on certain debug messages.

-dram:spd_input *?input_filename?*

Since it gets tedious to specify 20 different parameters to specify a memory system, the preferred way to specify DRAM memory system configuration and timing information is with a configuration file. Numerous timing parameters and DRAM system configurations can be specified with a .spd file. A sample .spd file is shown below. Comments are allowed after // Sample .spd files are defined under the subdirectory of /mem_system_def/

```
//  DDR2 800 Mbps memory system
//  Composed of 512 Mbit chips.  2 ranks, each rank has 8 2 Gbit (x8) chips.
//  This is a 64 bit wide interface.
//  Total is 1 GB
//  Bandwidth is  6.4 GB/s
//  The parameters contained in this file are from Micron datasheet for MT47H64M8 -25E
//
type            ddr2 // ddr2
datarate        800
clock_granularity 2 // 2 half cycles per cycle (timing listed in half cycles)
channel_count   1 // Logical channel
channel_width   8 // Byte width
PA_mapping_policyclose_page_baseline // Comments are allowed here
row_buffer_policy close_page
rank_count 2
bank_count4 // 4 banks per chip. larger chips have 8 banks.
row_count 16384 // 14 bits row address space
col_count 1024 // 10 bits col address space
t_burst 8 // burst length of 4 or 8
t_cas 10 // 5 * 2
t_faw 30 // 22.5 ns
t_ras 36 // 30 ns
t_rc 46 // 45 ns
t_rcd 10 // 12.5 ns
t_rfc 102 // takes 127 ns to do one refresh
t_rrd 6 // 7.5 ns
t_rp 10 // 12.5 ns
t_rtp           6 // 3 * 2
t_rtrs          2 // 1 * 2
t_wr            12        // 10 ns
t_wtr           6         //
posted_cas TRUE // posted CAS
t_al 10 // 4 * 2
auto_refresh     FALSE // Enable auto refresh
auto_refresh_policy refresh_one_chan_all_rank_all_bank
refresh_time  64000 // 64 ms refresh time
```

Fig. 4: DRAM Configuration File for DDR2 SDRAM Memory System with 512 Mbit Devices

-request_count *?request_count?*

specifies the number of requests to be simulated.

-tracefile *?trace_filename?*

DRAMSim2 can be driven with a trace input file or random number generated input. The trace flle specifies a trace input.