Multi-Banking system in Main memory

Alok Kumar Sharma(cs20m001@iittp.ac.in)

Saurav Kumar Singh(cs20m012@iittp.ac.in)

Guide:
Dr. Jayanarayan Thakurdas Tudu (jtt@iittp.ac.in)

Abstract--

In this paper we are present about the DRAM simulation. Our focus is DDR2/DDR3 ram. The simulator which we used, works on x86. In this paper our focus is to visualize and compare the memory system. We worked on the performance metrics such as bandwidth latency and power.

1. Introduction

There are many simulators like simplistic models of memory and many others. The main reason to select DRAMSim2 is because of the accurate memory cycle. We have also seen that many simulators fail during the high complexity behaviour of the memory system. DRAMSim2 has a very simple programming interface with an object-oriented design which provides a verification tool that can be used to validate the result. In this paper we also understand the structure of the memory system and how the multibank is configured. How the data is transferred from the memory bank to the cache buffer. And the main part which we worked on are different kinds of memory latency.

2. DRAMSim2 Architecture

In this section we will briefly discuss how the DRAM simulator 2 structure is designed and how it works. The structure of the DRAM simulator 2 is given in Figure 1.

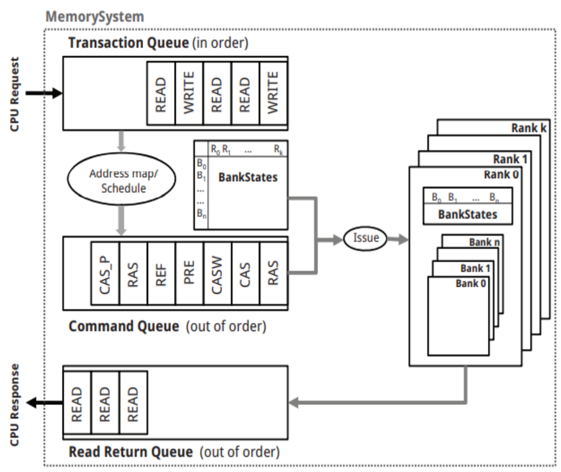


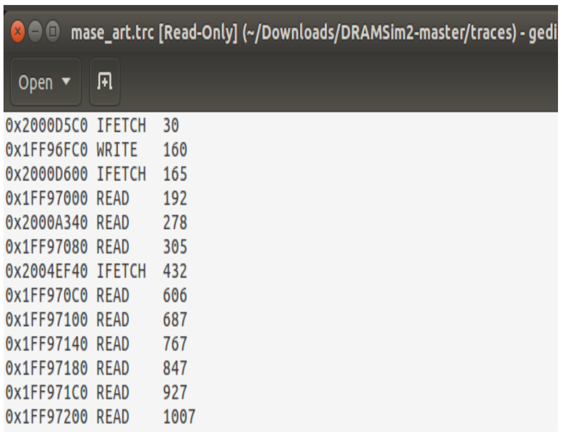
Fig. 1. Overview of DRAMSim2 components. The MemorySystem wrapper provides a simple user interface for any front end driver.

To run the simulator, we need two ini files. 1st device ini file and 2nd system ini file. A device ini file contains parameters that describe a specific DRAM device such as the timing constraints and power consumption of the device. Now this parameter which is documented in the device ini file can we get from the website of the device.

system ini file consists of parameters such as the number of ranks, the address mapping scheme, debug options, row buffer policies, memory controller queue structures, and other simulation details.

3. Experiment

We have used the tracer file named as "mase_art.trc.gz". In the tracer file it contains the instructions to be executed. The instructions contain the register, memory address and operation to be done.

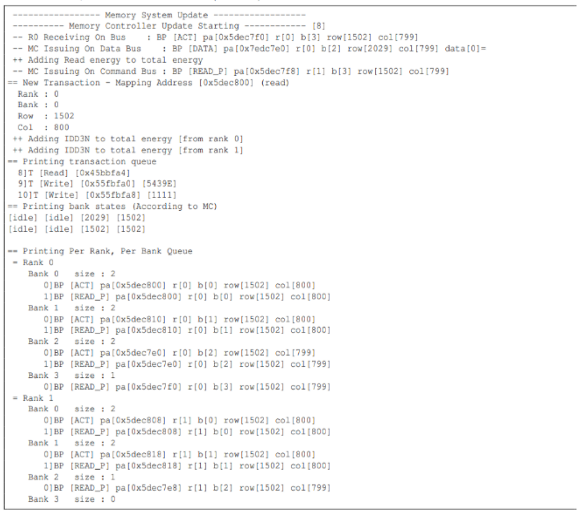

mase_art.trc [Read-Only] (~/Downloads/DRAMSim2-master/traces) - gedi
Open

```
0x2000D5C0 IFETCH  30
0x1FF96FC0 WRITE   160
0x2000D600 IFETCH  165
0x1FF97000 READ    192
0x2000A340 READ    278
0x1FF97080 READ    305
0x2004EF40 IFETCH  432
0x1FF970C0 READ    606
0x1FF97100 READ    687
0x1FF97140 READ    767
0x1FF97180 READ    847
0x1FF971C0 READ    927
0x1FF97200 READ    1007
```

The program will read the tracer file and generate the per clock cycle statistics.

4. Result

The experiment carried out generated the result file in ".vas" format. The detailed report is attached on the next page. In DRAMSim2 we can customize the "system.ini" file by turning the various debug flags on or off. Some of the points in the report are explained below.



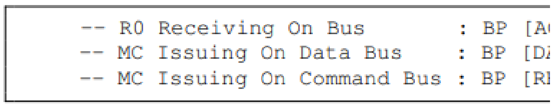
```
 ----------------- Memory System Update ------------------
 ---------- Memory Controller Update Starting ------------ [8]
 -- R0 Receiving On Bus    : BP [ACT] pa[0x5dec7f0] r[0] b[3] row[1502] col[799]
 -- MC Issuing On Data Bus    : BP [DATA] pa[0x7edc7e0] r[0] b[2] row[2029] col[799] data[0]=
 ++ Adding Read energy to total energy
 -- MC Issuing On Command Bus : BP [READ_P] pa[0x5dec7f8] r[1] b[3] row[1502] col[799]
== New Transaction - Mapping Address [0x5dec800] (read)
  Rank : 0
  Bank : 0
  Row  : 1502
  Col  : 800
 ++ Adding IDD3N to total energy [from rank 0]
 ++ Adding IDD3N to total energy [from rank 1]
== Printing transaction queue
  8]T [Read] [0x45bbfa4]
  9]T [Write] [0x55fbfa0] [5439E]
  10]T [Write] [0x55fbfa8] [1111]
== Printing bank states (According to MC)
[idle] [idle] [2029] [1502]
[idle] [idle] [1502] [1502]

== Printing Per Rank, Per Bank Queue
 - Rank 0
    Bank 0   size : 2
       0]BP [ACT] pa[0x5dec800] r[0] b[0] row[1502] col[800]
       1]BP [READ_P] pa[0x5dec800] r[0] b[0] row[1502] col[800]
    Bank 1   size : 2
       0]BP [ACT] pa[0x5dec810] r[0] b[1] row[1502] col[800]
       1]BP [READ_P] pa[0x5dec810] r[0] b[1] row[1502] col[800]
    Bank 2   size : 2
       0]BP [ACT] pa[0x5dec7e0] r[0] b[2] row[1502] col[799]
       1]BP [READ_P] pa[0x5dec7e0] r[0] b[2] row[1502] col[799]
    Bank 3   size : 1
       0]BP [READ_P] pa[0x5dec7f0] r[0] b[3] row[1502] col[799]
 = Rank 1
    Bank 0   size : 2
       0]BP [ACT] pa[0x5dec808] r[1] b[0] row[1502] col[800]
       1]BP [READ_P] pa[0x5dec808] r[1] b[0] row[1502] col[800]
    Bank 1   size : 2
       0]BP [ACT] pa[0x5dec818] r[1] b[1] row[1502] col[800]
       1]BP [READ_P] pa[0x5dec818] r[1] b[1] row[1502] col[800]
    Bank 2   size : 1
       0]BP [READ_P] pa[0x5dec7e8] r[1] b[2] row[1502] col[799]
    Bank 3   size : 0
```

Anything sent on the bus is encapsulated in a Bus Packet (BP) object. When printing, they display the following information:

BP [ACT] pa[0x5dec818] r[1] b[1] row[1502] col[800]

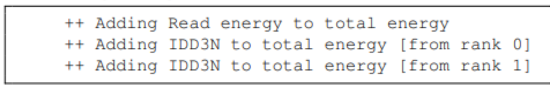
The information displayed is (in order): command type, physical address, rank #, bank #, row #, and column #.

Lines beginning with " – " indicate bus traffic.



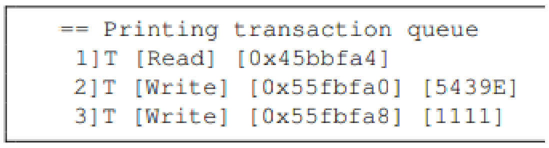
```
 -- R0 Receiving On Bus        : BP [A
 -- MC Issuing On Data Bus     : BP [D
 -- MC Issuing On Command Bus  : BP [R
```

Lines beginning with "++" indicate power calculations



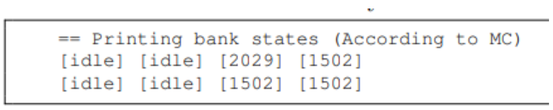
```
 ++ Adding Read energy to total energy
 ++ Adding IDD3N to total energy [from rank 0]
 ++ Adding IDD3N to total energy [from rank 1]
```

If a pending transaction is in the transaction queue, it will be printed as



```
== Printing transaction queue
  1]T [Read] [0x45bbfa4]
  2]T [Write] [0x55fbfa0] [5439E]
  3]T [Write] [0x55fbfa8] [1111]
```

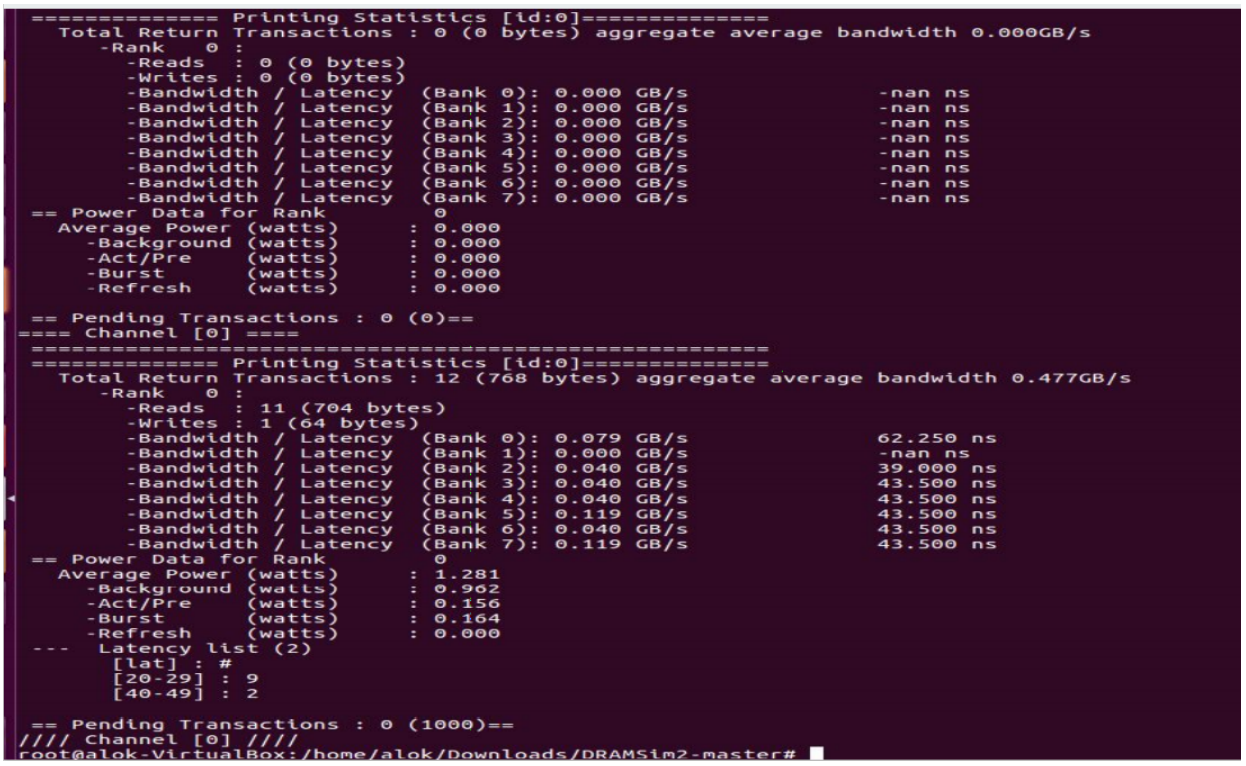
The state of each bank in the system is also displayed



```
== Printing bank states (According to MC)
[idle] [idle] [2029] [1502]
[idle] [idle] [1502] [1502]
```

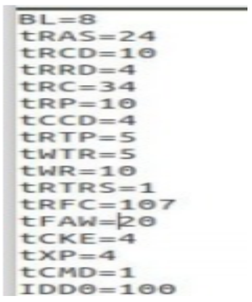
The output of the file also gives information about the various clock cycles. And the detailed information about the bandwidth and latency.

The screen shot of the output is attached below.



```
============== Printing Statistics [id:0]==============
   Total Return Transactions : 0 (0 bytes) aggregate average bandwidth 0.000GB/s
      -Rank   0 :
        -Reads  : 0 (0 bytes)
        -Writes : 0 (0 bytes)
        -Bandwidth / Latency  (Bank 0): 0.000 GB/s                -nan ns
        -Bandwidth / Latency  (Bank 1): 0.000 GB/s                -nan ns
        -Bandwidth / Latency  (Bank 2): 0.000 GB/s                -nan ns
        -Bandwidth / Latency  (Bank 3): 0.000 GB/s                -nan ns
        -Bandwidth / Latency  (Bank 4): 0.000 GB/s                -nan ns
        -Bandwidth / Latency  (Bank 5): 0.000 GB/s                -nan ns
        -Bandwidth / Latency  (Bank 6): 0.000 GB/s                -nan ns
        -Bandwidth / Latency  (Bank 7): 0.000 GB/s                -nan ns
 == Power Data for Rank        0
   Average Power (watts)     : 0.000
     -Background (watts)     : 0.000
     -Act/Pre    (watts)     : 0.000
     -Burst      (watts)     : 0.000
     -Refresh    (watts)     : 0.000

 == Pending Transactions : 0 (0)==
==== Channel [0] ====
 =================================================
 ============== Printing Statistics [id:0]==============
   Total Return Transactions : 12 (768 bytes) aggregate average bandwidth 0.477GB/s
      -Rank   0 :
        -Reads  : 11 (704 bytes)
        -Writes : 1 (64 bytes)
        -Bandwidth / Latency  (Bank 0): 0.079 GB/s                62.250 ns
        -Bandwidth / Latency  (Bank 1): 0.000 GB/s                -nan ns
        -Bandwidth / Latency  (Bank 2): 0.040 GB/s                39.000 ns
        -Bandwidth / Latency  (Bank 3): 0.040 GB/s                43.500 ns
        -Bandwidth / Latency  (Bank 4): 0.040 GB/s                43.500 ns
        -Bandwidth / Latency  (Bank 5): 0.119 GB/s                43.500 ns
        -Bandwidth / Latency  (Bank 6): 0.040 GB/s                43.500 ns
        -Bandwidth / Latency  (Bank 7): 0.119 GB/s                43.500 ns
 == Power Data for Rank        0
   Average Power (watts)     : 1.281
     -Background (watts)     : 0.962
     -Act/Pre    (watts)     : 0.156
     -Burst      (watts)     : 0.164
     -Refresh    (watts)     : 0.000
 ---  Latency list (2)
       [lat] : #
       [20-29] : 9
       [40-49] : 2

 == Pending Transactions : 0 (1000)==
//// Channel [0] ////
root@alok-VirtualBox:/home/alok/Downloads/DRAMSim2-master#
```

Now we can see that bandwidth/Latency is given. If we see the output, the latency is nearly the same for the last 5 banks, which is “43.5”. but the bandwidth varies. If we can somehow make the bandwidth same for all then we can optimise the latency. For this we must use scheduling techniques that will distribute the data across all the banks evenly. We can also come up with techniques such as out-of-order speculative execution, vector, stream, and massive multithreading, or multi level caches that lower the average memory access time. There is detailed information of cycles used in various activities.



```
BL=8
tRAS=24
tRCD=10
tRRD=4
tRC=34
tRP=10
tCCD=4
tRTP=5
tWTR=5
tWR=10
tRTRS=1
tRFC=107
tFAW=20
tCKE=4
tXP=4
tCMD=1
IDD0=100
```

tRAS uses 24 cycles and tRCD is 10 cycles. tRAS is minimum number of clock cycles required between a row active command and issuing the pre-charge command. tRCD is the number of clock cycles required when it is an open row and needs to access a column. Now if we have a good scheduling algorithm which can minimize the **Miss rate** and increase the **Hit rate** then we can reduce the number of tRAS cycles. And if we can reduce the tRAS then we can reduce the overall cycle and the latency can be decreased.

5. Future Work

After completing this project now, we want to explore more in this area and want to know how this can be used in a Distributed System, where we can use a multibank for a multiprocessor environment. And I want to start building the simulator from scratch. It will give more flexibility in terms of checking the performance.

References

- Journals & magazines>IEEE Computer Architecture Le…>Volume:10 Issue :1

  DRAMSim2: A cycle Accurate Memory System Simulator