

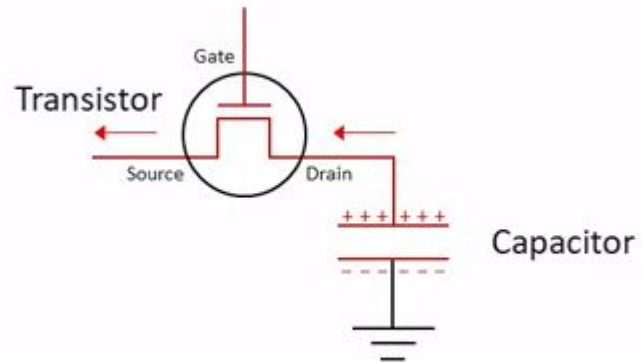


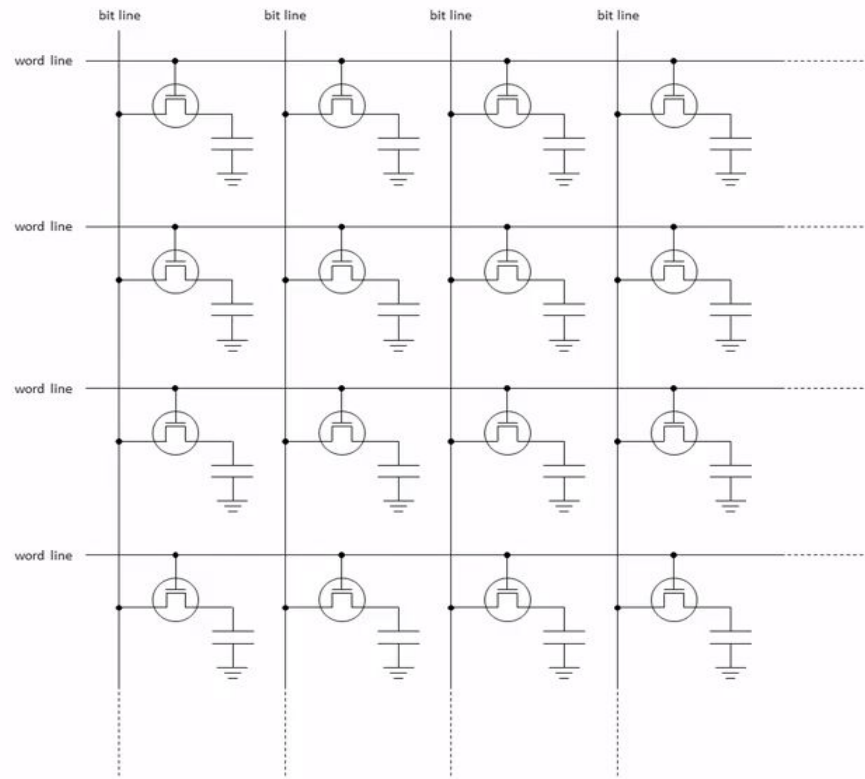
## Main memory simulator for multi-bank system

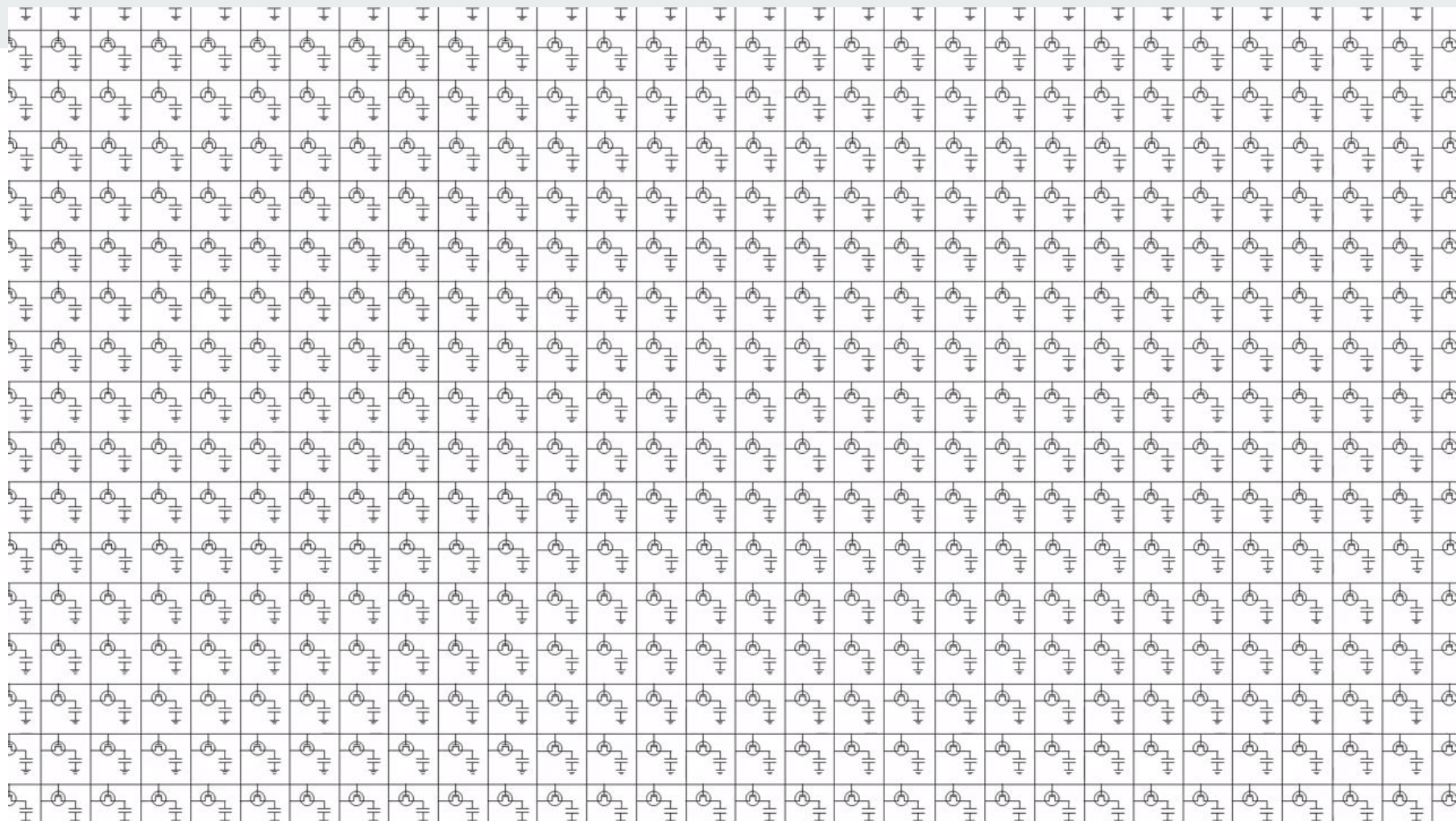
CS20M001

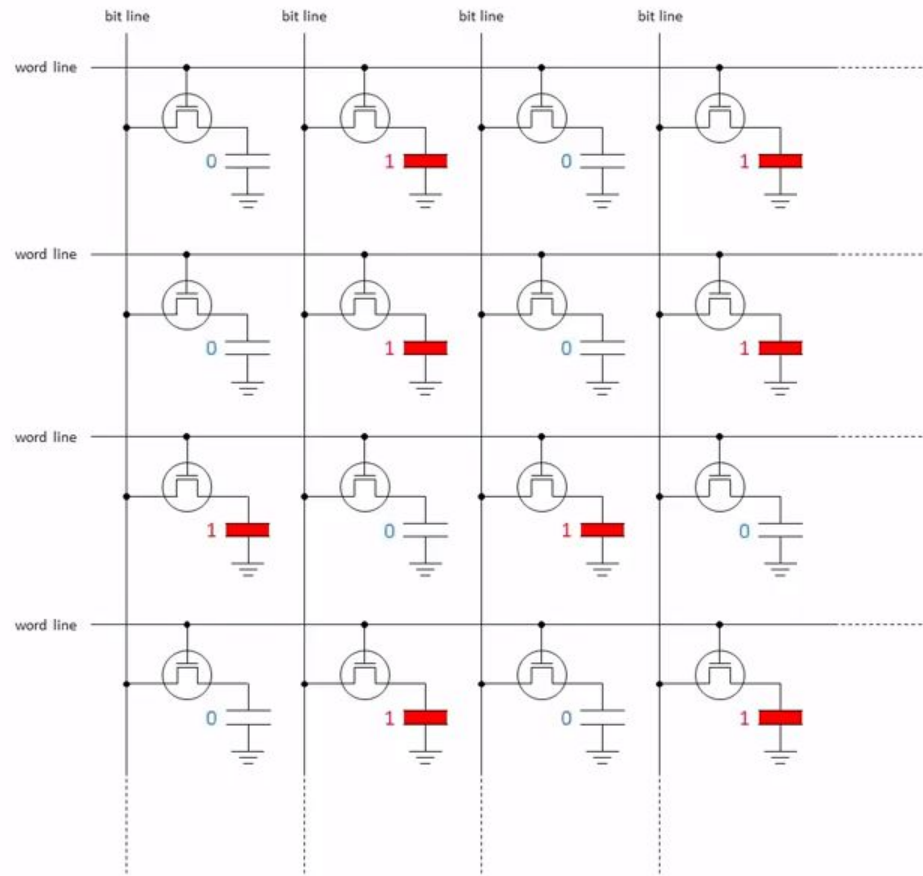
CS20M012

# Memory Cell

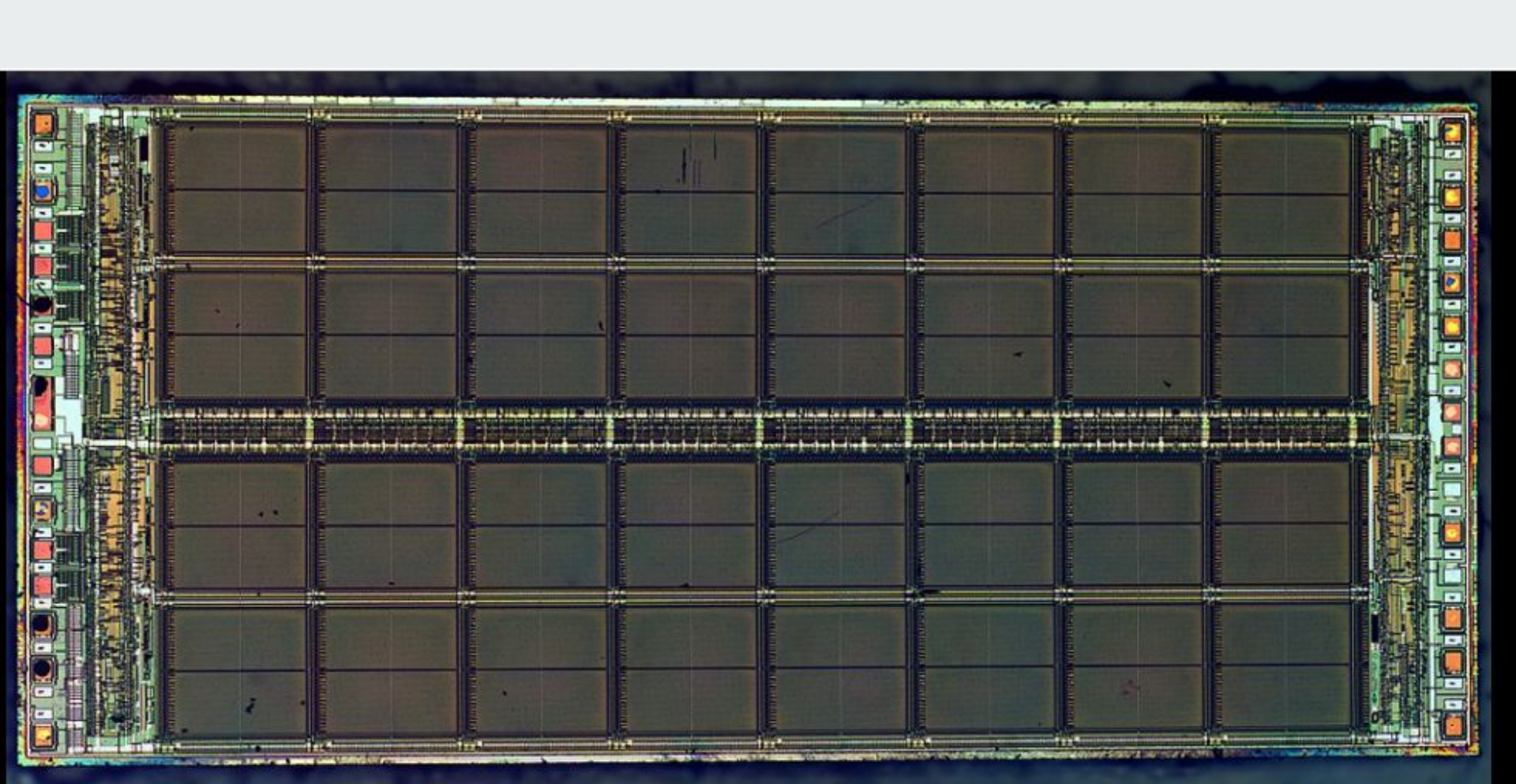










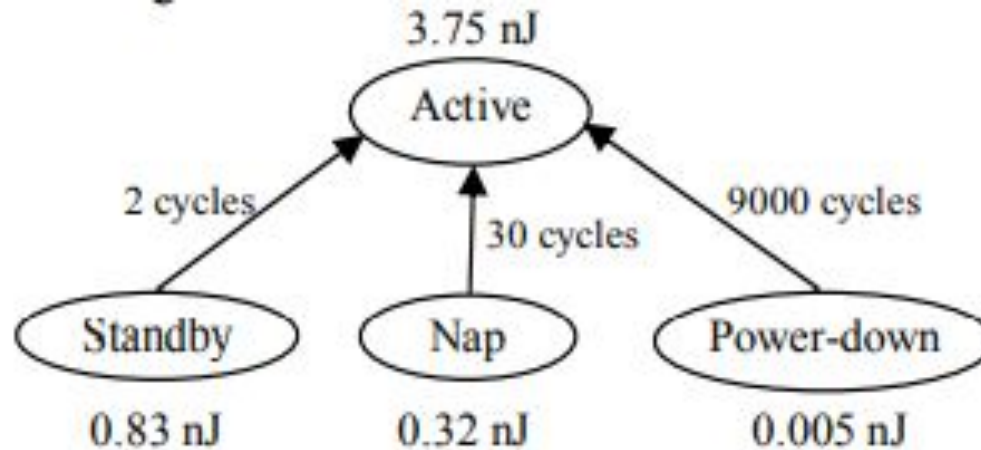




## Multi-Bank Memory Allocation

The main memory is consuming an increasing proportion of the power budget and thus motivates efforts to improve DRAM energy efficiency. On the other hand, memories with multiple banks appeared in several architectures. This kind of memory architecture was recently exploited to reduce energy dissipation by operating banks at different modes (Active, Standby, Nap, Power-Down...) for example RAMBUS-DRAM technology (RDRAM). To service a memory request (read or write), a bank must be in active mode which consumes most of the power.

# Energy consumption and resynchronization times for different operating modes.







## Memory Consumption Parameters

The energy consumption monotonically increases with memory size. For the multi-bank main memory, several papers consider that the energy values given in the figure (Active, Standby, Nap and Power-down) increases by  $\tau_1 = 30\%$  when bank size is doubled [1, 3]. In our approach we consider that the size  $S_{bj}$  of bank  $b_j$  is the sum of the size of all tasks  $T_i$  allocated to this bank:

$$S_{bj} = \sum_{T_i / \varphi(T_i) = b_j} S_{T_i}$$

$$E_x = E_{0x} \times (1.3)^{\log(\frac{S_{bj}}{8})}$$



## Number of banks.

The multi-bank energy consumption also depends on the number of banks in the memory architecture. When a new bank is added, the sizes of banks decrease as well as the energy values. However, we assume that the energy consumption for communication increases by  $\tau_2 = 20\%$  when we add a new bank to the architecture. So for main memory architecture with  $k$  banks, the communication energy is described by

$$E_{bus} = E_{0bus} (1.2)^{k-1}$$

$E_{0bus}$ : The bus consumption for one bank main memory architecture (monolithic memory).



## Successivity and Preemption.

We call successivity between task  $T_i$  and task  $T_j$ , noted  $\sigma_{ij}$ , when  $T_j$  begins its execution just after the end of  $T_i$  or when the higher priority task ( $T_i$  or  $T_j$ ) preempts the other one. The successivity parameters are deduced from the application scheduling during the hyperperiod. They are exploited to minimize the resynchronization number of memory banks and making the idle period of banks as long as possible. The resynchronization number of a bank  $b_j$  is computed as follows, where  $N_{exeT_i}$  is the number of times the task  $T_i$  was executed during the hyperperiod.

$$N_{resynchronization\_b_j} = \sum_{T_i / \varphi(T_i)=b_j} N_{exeT_i} - \sum_{T_i, T_j / (\varphi(T_i), \varphi(T_j))=(b_j, b_j)} \sigma_{ij}$$

# Energy Models for a Multi-Bank Memory



The energy consumption of a memory composed of  $k$  banks and a given allocation of  $N$  tasks to these banks can be evaluated with

$$E_{\text{memory}} = E_{\text{access}} + E_{\text{nonaccess}} + E_{\text{lpmode}} + E_{\text{resynchronization}} + E_{\text{preemption}} + E_{\text{bus}}$$

We separate the Active mode into two different operating modes: the read/write mode (access) and active but idle mode (non-access).  $E_{\text{access}}$  is the energy due to read or write accesses to the memory banks while  $E_{\text{nonaccess}}$  is the energy consumption when the memory banks are active but not servicing any read or write operation.

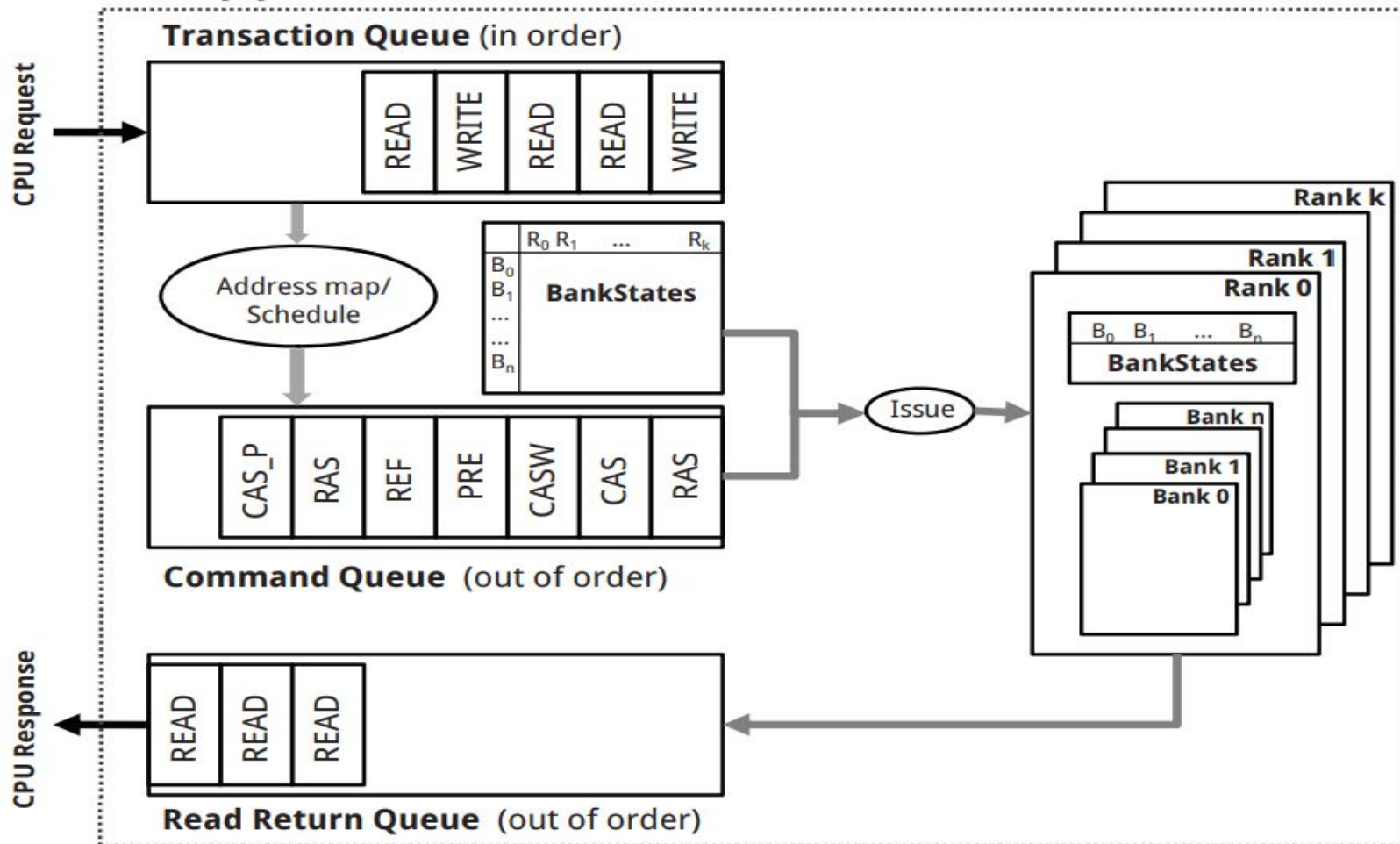


# EVALUATION - 3

CS20M001

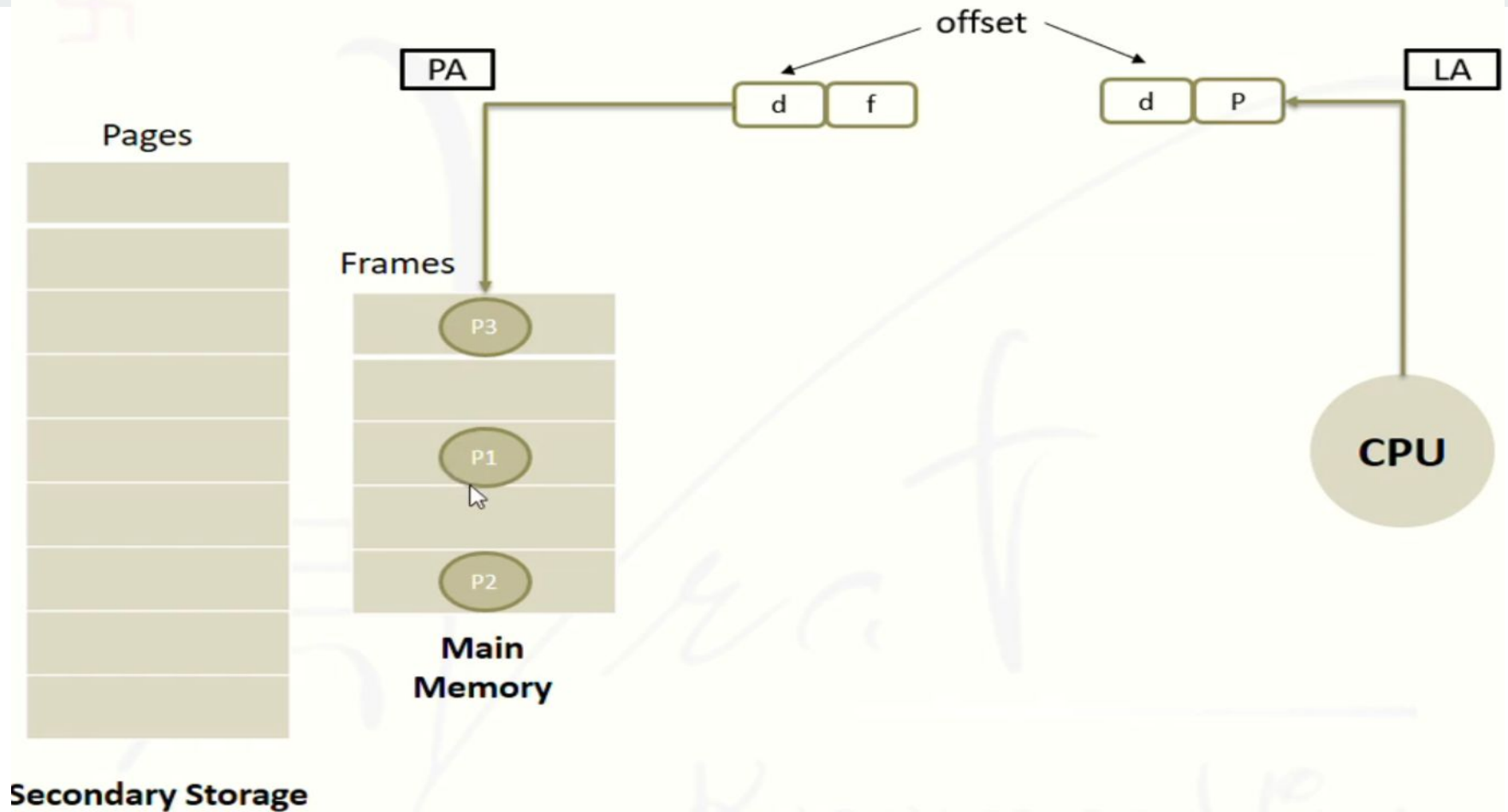
CS20M012

## MemorySystem

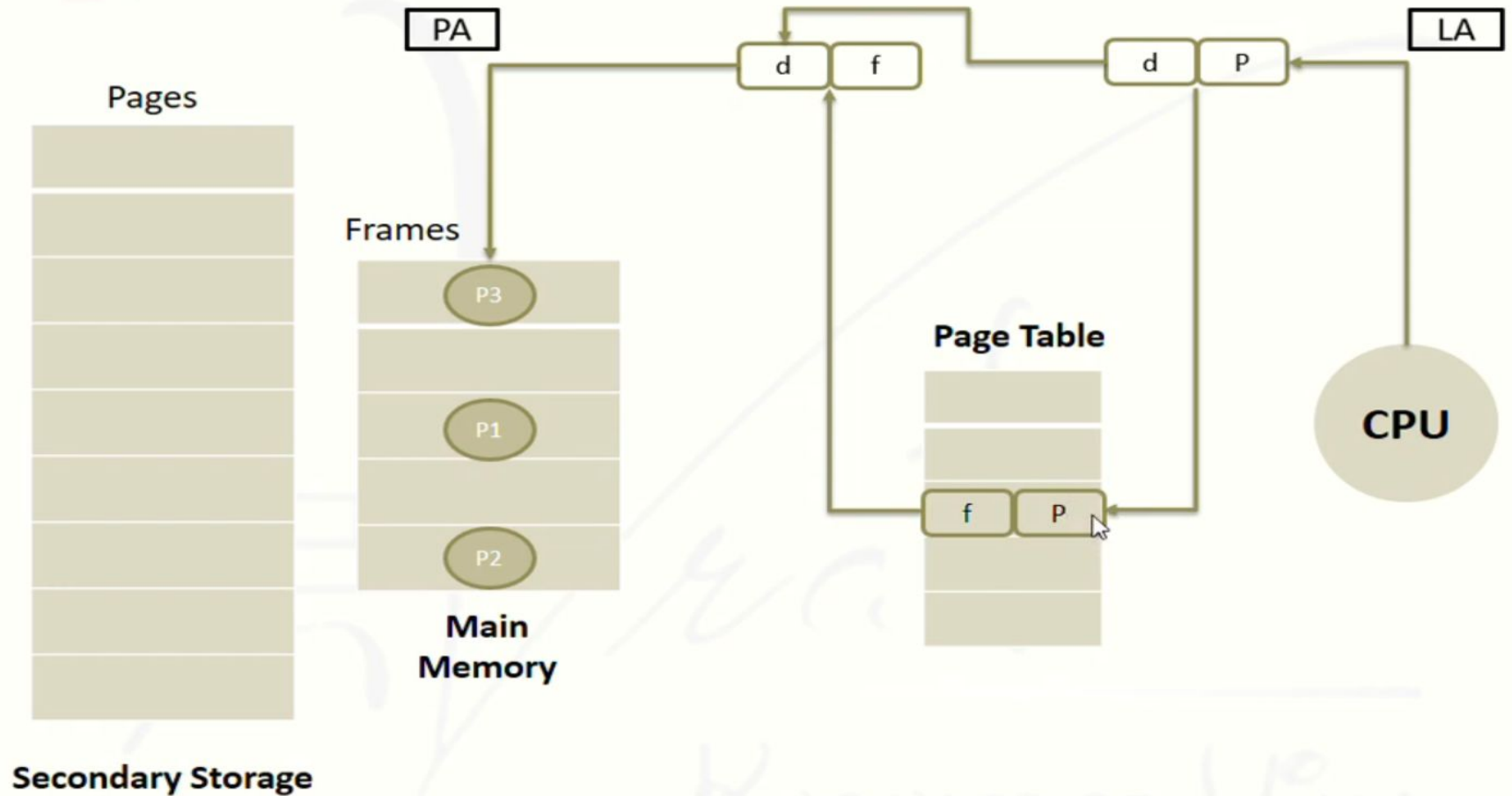




# Paging



# Paging



# DRAM Bank Operation

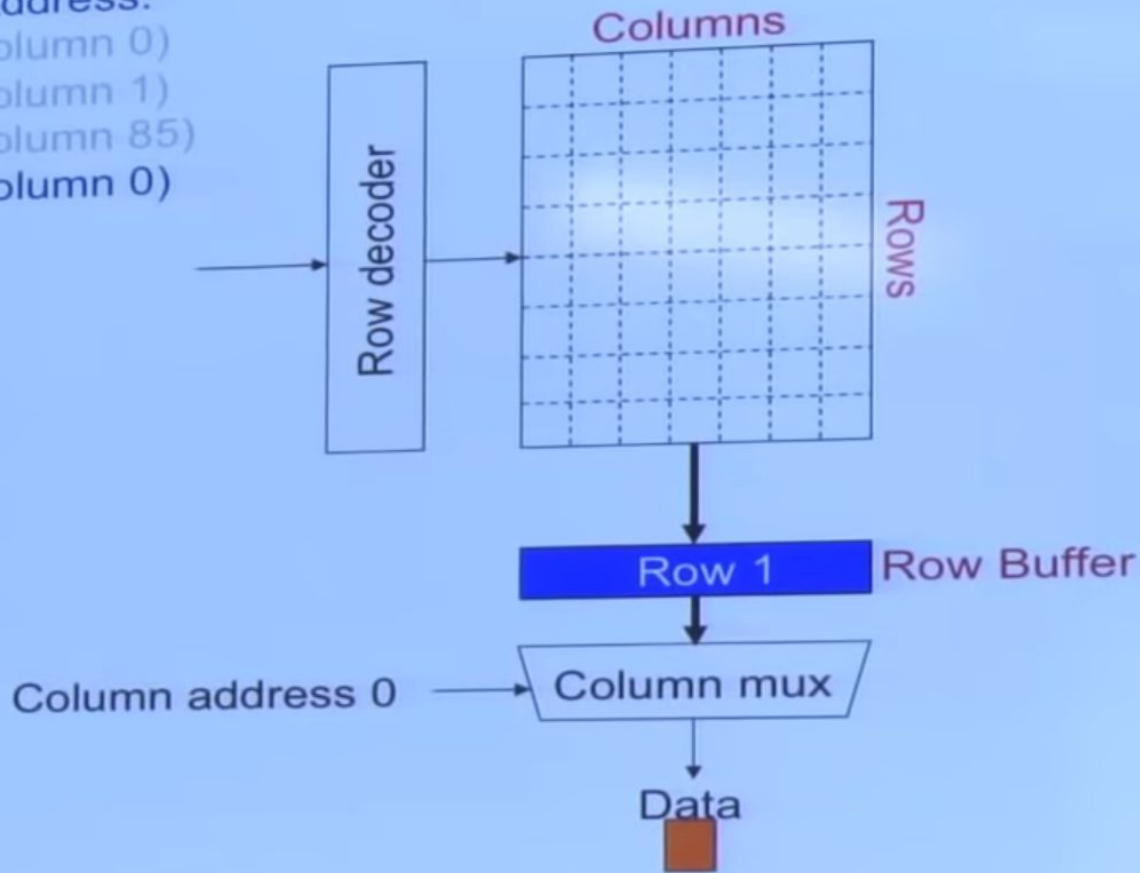
Access Address:

(Row 0, Column 0)

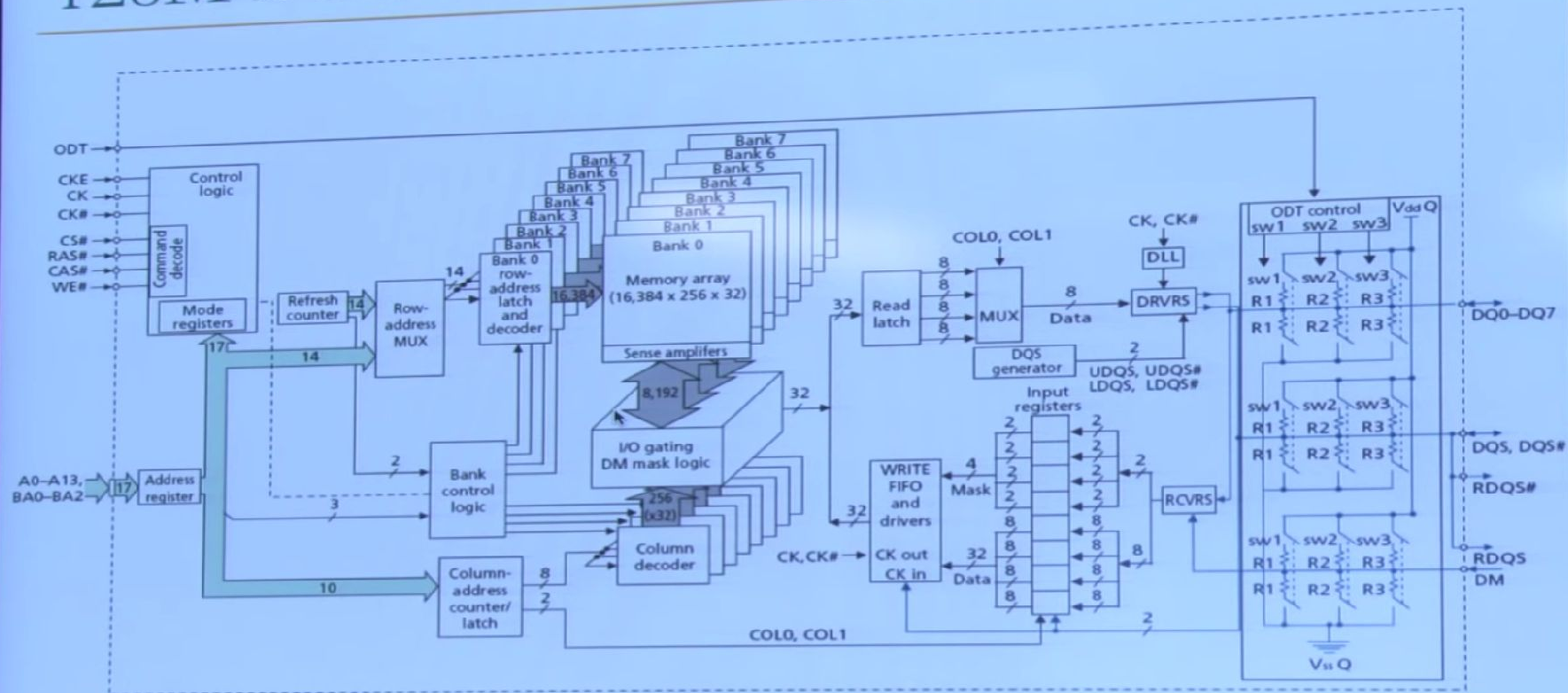
(Row 0, Column 1)

(Row 0, Column 85)

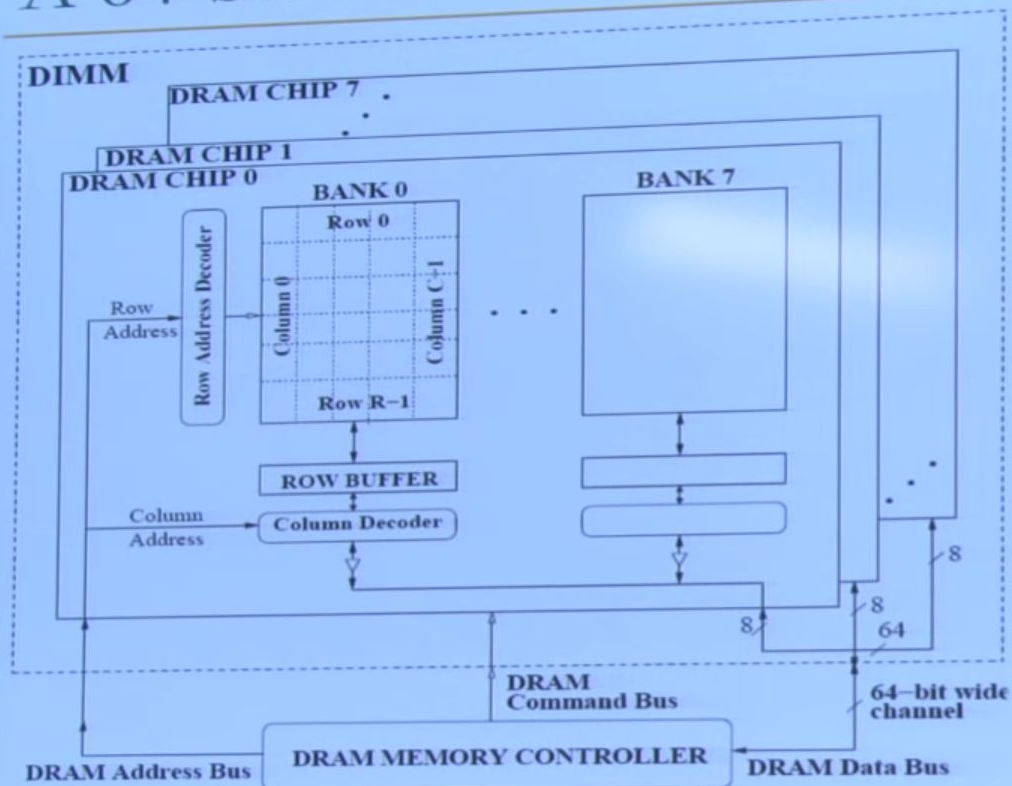
(Row 1, Column 0)



# 128M x 8-bit DRAM Chip



# A 64-bit Wide DIMM (One Rank)



## Advantages:

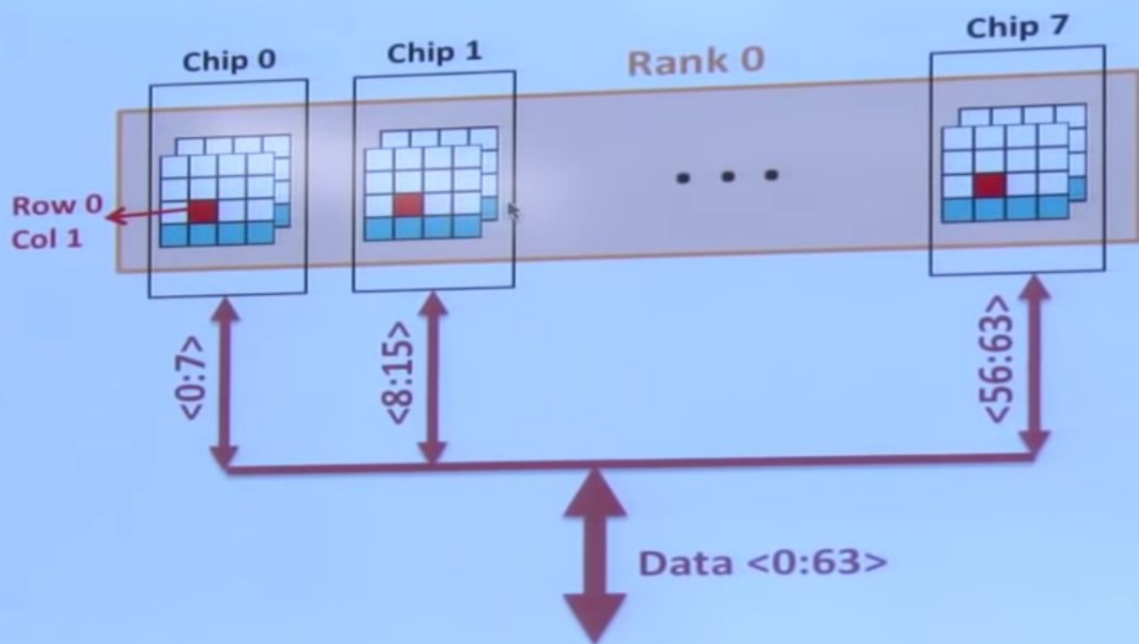
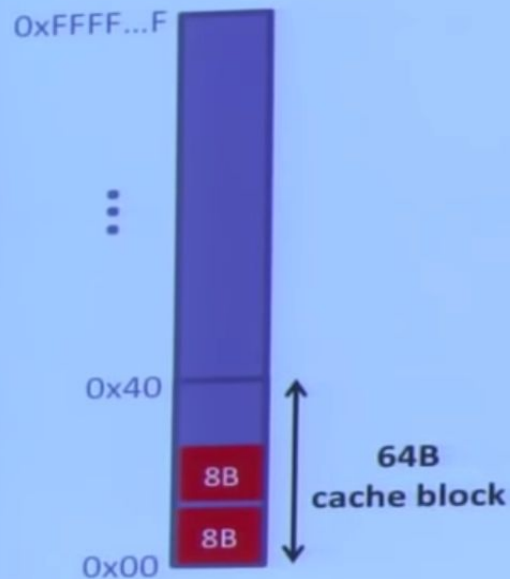
- Acts like a **high-capacity DRAM chip** with a **wide interface**
- **Flexibility:** memory controller does not need to deal with individual chips

## Disadvantages:

- **Granularity:** Accesses cannot be smaller than the interface width

# Example: Transferring a cache block

Physical memory space



A 64B cache block takes 8 I/O cycles to transfer.

During the process, 8 columns are read sequentially.



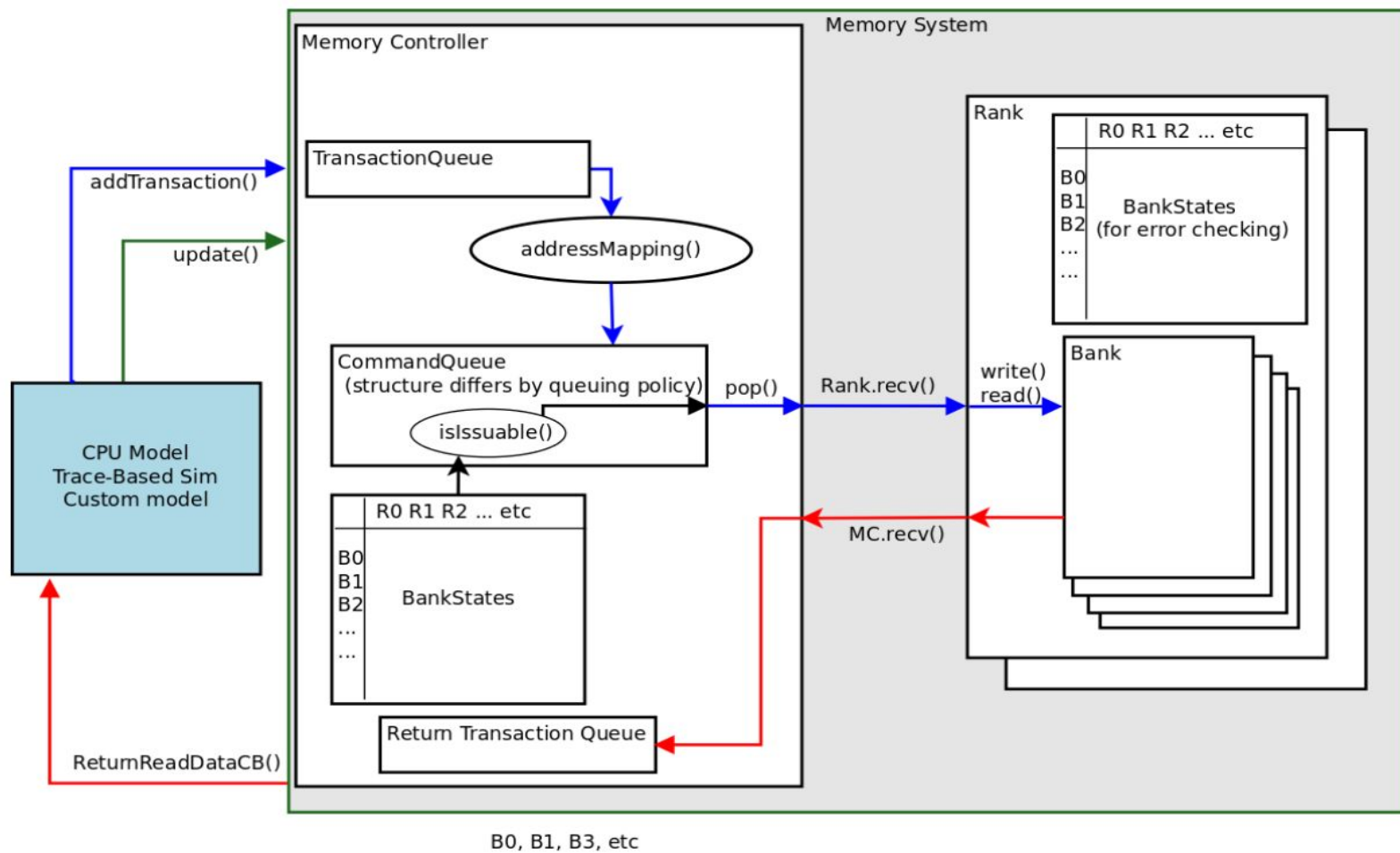
## Memory Timings Explained

1. **CAS Latency** ( $t_{CL}$ ) - This is the most important memory timing. CAS stands for Column Address Strobe. If a row has already been selected, it tells us how many clock cycles we'll have to wait for a result (after sending a column address to the RAM controller).
2. **Row Address (RAS) to Column Address (CAS) Delay** ( $t_{RCD}$ ) - Once we send the memory controller a row address, we'll have to wait this many cycles before accessing one of the row's columns. So, if a row hasn't been selected, this means we'll have to wait  $t_{RCD} + t_{CL}$  cycles to get our result from the RAM.
3. **Row Precharge Time** ( $t_{RP}$ ) - If we already have a row selected, we'll have to wait this number of cycles before selecting a different row. This means it will take  $t_{RP} + t_{RCD} + t_{CL}$  cycles to access the data in a different row.
4. **Row Active Time** ( $t_{RAS}$ ) - This is the minimum number of cycles that a row has to be active for to ensure we'll have enough time to access the information that's in it. This usually needs to be greater than or equal to the sum of the previous three latencies ( $t_{RAS} = t_{CL} + t_{RCD} + t_{RP}$ ).

# Latency Components: Basic DRAM Operation

---

- CPU → controller transfer time
- Controller latency
  - Queuing & scheduling delay at the controller
  - Access converted to basic commands
- Controller → DRAM transfer time
- DRAM bank latency
  - Simple CAS (column address strobe) if row is “open” OR
  - RAS (row address strobe) + CAS if array precharged OR
  - PRE + RAS + CAS (worst case)
- DRAM → Controller transfer time
  - Bus latency (BL)
- Controller to CPU transfer time



Block diagram of DRAMSim2. The `recv()` functions are actually called `receiveFromBus()` but were abbreviated to save space.

# Processing of Running the Simulator

- We need to unpack the simulator.
- We need to go in the folder name “DRAMim2-master” :  
\$Cd DRAMim2-master
- “\$ Make”

```
alok@alok-VirtualBox:~/Downloads/DRAMSIm2-master$ make clean
rm -f AddressMapping.o CommandQueue.o BusPacket.o Bank.o SimulatorObject.o ClockDomain.o Rank.o MemoryController.o MultiChannelMemorySystem.o BankState.o TraceBasedSim.o MemorySystem.o PrintMacros.po Transaction.po Rank.po BusPacket.po BankState.po CommandQueue.po MemorySystem.po ClockDomain.po MemoryController.po MultiChannelMemorySystem.po IniReader.po DRAMSIm libdramsim.so libdramsim.a *.dep *.deppo
alok@alok-VirtualBox:~/Downloads/DRAMSIm2-master$ make
g++ -DNO_STORAGE -Wall -DDEBUG_BUILD -O3 -o AddressMapping.o -c AddressMapping.cpp
g++ -DNO_STORAGE -Wall -DDEBUG_BUILD -O3 -o PrintMacros.o -c PrintMacros.cpp
g++ -DNO_STORAGE -Wall -DDEBUG_BUILD -O3 -o Transaction.o -c Transaction.cpp
g++ -DNO_STORAGE -Wall -DDEBUG_BUILD -O3 -o BusPacket.o -c BusPacket.cpp
g++ -DNO_STORAGE -Wall -DDEBUG_BUILD -O3 -o Rank.o -c Rank.cpp
g++ -DNO_STORAGE -Wall -DDEBUG_BUILD -O3 -o BankState.o -c BankState.cpp
g++ -DNO_STORAGE -Wall -DDEBUG_BUILD -O3 -o CommandQueue.o -c CommandQueue.cpp
g++ -DNO_STORAGE -Wall -DDEBUG_BUILD -O3 -o MemorySystem.o -c MemorySystem.cpp
g++ -DNO_STORAGE -Wall -DDEBUG_BUILD -O3 -o TraceBasedSim.o -c TraceBasedSim.cpp
g++ -DNO_STORAGE -Wall -DDEBUG_BUILD -O3 -o Bank.o -c Bank.cpp
g++ -DNO_STORAGE -Wall -DDEBUG_BUILD -O3 -o ClockDomain.o -c ClockDomain.cpp
g++ -DNO_STORAGE -Wall -DDEBUG_BUILD -O3 -o SimulatorObject.o -c SimulatorObject.cpp
g++ -DNO_STORAGE -Wall -DDEBUG_BUILD -O3 -o MemoryController.o -c MemoryController.cpp
g++ -DNO_STORAGE -Wall -DDEBUG_BUILD -O3 -o MultiChannelMemorySystem.o -c MultiChannelMemorySystem.cpp
g++ -DNO_STORAGE -Wall -DDEBUG_BUILD -O3 -o IniReader.o -c IniReader.cpp
g++ -DNO_STORAGE -Wall -DDEBUG_BUILD -O3 -o DRAMSIm AddressMapping.o PrintMacros.o Transaction.o BusPacket.o Rank.o BankState.o CommandQueue.o SimulatorObject.o MemoryController.o MultiChannelMemorySystem.o IniReader.o
Built DRAMSIm successfully
```



# To build the DRAMSim2 library

- “\$ make libdramsim.so”

```
alok@alok-VirtualBox:~/Downloads/DRAMSim2-master$ make libdramsim.so
g++ -DNO_STORAGE -Wall -DDEBUG_BUILD -O3 -DLOG_OUTPUT -fPIC -o AddressMapping.po -c AddressMapping.cpp
g++ -DNO_STORAGE -Wall -DDEBUG_BUILD -O3 -DLOG_OUTPUT -fPIC -o PrintMacros.po -c PrintMacros.cpp
g++ -DNO_STORAGE -Wall -DDEBUG_BUILD -O3 -DLOG_OUTPUT -fPIC -o Transaction.po -c Transaction.cpp
g++ -DNO_STORAGE -Wall -DDEBUG_BUILD -O3 -DLOG_OUTPUT -fPIC -o Rank.po -c Rank.cpp
g++ -DNO_STORAGE -Wall -DDEBUG_BUILD -O3 -DLOG_OUTPUT -fPIC -o BusPacket.po -c BusPacket.cpp
g++ -DNO_STORAGE -Wall -DDEBUG_BUILD -O3 -DLOG_OUTPUT -fPIC -o BankState.po -c BankState.cpp
g++ -DNO_STORAGE -Wall -DDEBUG_BUILD -O3 -DLOG_OUTPUT -fPIC -o CommandQueue.po -c CommandQueue.cpp
g++ -DNO_STORAGE -Wall -DDEBUG_BUILD -O3 -DLOG_OUTPUT -fPIC -o MemorySystem.po -c MemorySystem.cpp
g++ -DNO_STORAGE -Wall -DDEBUG_BUILD -O3 -DLOG_OUTPUT -fPIC -o Bank.po -c Bank.cpp
g++ -DNO_STORAGE -Wall -DDEBUG_BUILD -O3 -DLOG_OUTPUT -fPIC -o ClockDomain.po -c ClockDomain.cpp
g++ -DNO_STORAGE -Wall -DDEBUG_BUILD -O3 -DLOG_OUTPUT -fPIC -o SimulatorObject.po -c SimulatorObject.cpp
g++ -DNO_STORAGE -Wall -DDEBUG_BUILD -O3 -DLOG_OUTPUT -fPIC -o MemoryController.po -c MemoryController.cpp
g++ -DNO_STORAGE -Wall -DDEBUG_BUILD -O3 -DLOG_OUTPUT -fPIC -o MultiChannelMemorySystem.po -c MultiChannelMemorySystem.cpp
g++ -DNO_STORAGE -Wall -DDEBUG_BUILD -O3 -DLOG_OUTPUT -fPIC -o IniReader.po -c IniReader.cpp
g++ -g -shared -Wl,-soname,libdramsim.so -o libdramsim.so AddressMapping.po PrintMacros.po Transaction.po Rank.po BusPacket.p
ockDomain.po SimulatorObject.po MemoryController.po MultiChannelMemorySystem.po IniReader.po
Built libdramsim.so successfully
```



# Trace-Based Simulation

We have two file for trace simulation

- K6\_aoe\_02\_short.trc.
- Mase\_art.trc

We will we working on the 2nd file for the simulation.

We need to go in the folder name traces



# To run the preprocessor



Now we need to run the file “traceParse.py”

- `$ chmod 755 traceParse.py`

The .gz traces file should first be preprocessed before running through the simulator

- `$ cd traces`
- `$ ./traceParse.py mase_art.trc.gz`

This will create file by the name mase\_art.trc in the traces folder.



## Running the trace based simulator:

We need to run the below command to generate the result

- `./DRAMSim -t traces/k6_aoe_02_short.trc -s system.ini -d ini/DDR3_micron_64M_8B_x4_sg15.ini -c 1000`

1000 means it will run a 1000 cycle simulation

```

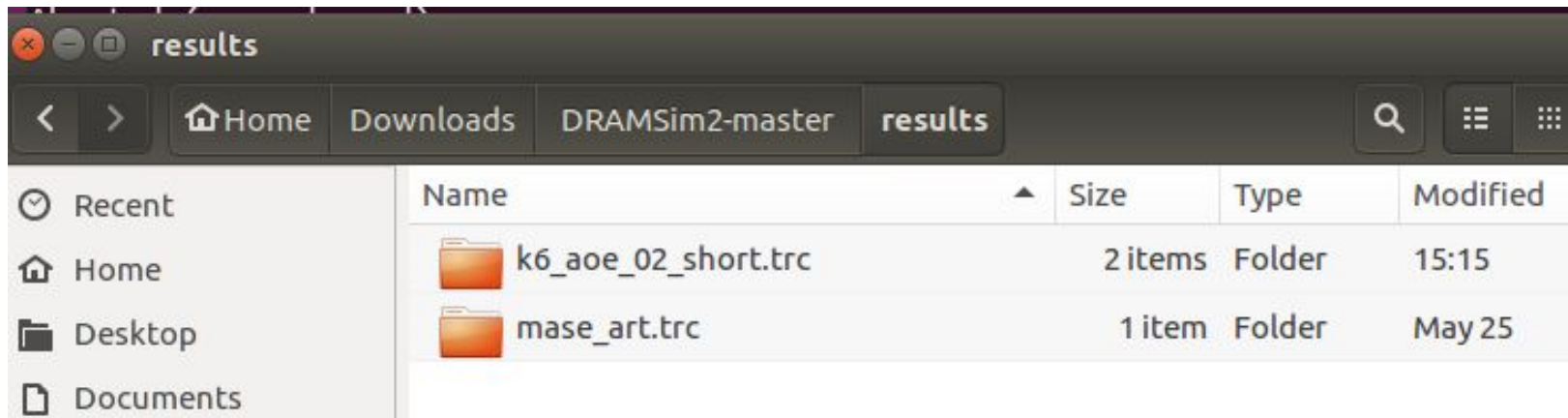
===== Printing Statistics [id:0]=====
Total Return Transactions : 0 (0 bytes) aggregate average bandwidth 0.000GB/s
-Rank 0 :
-Reads : 0 (0 bytes)
-Writes : 0 (0 bytes)
-Bandwidth / Latency (Bank 0): 0.000 GB/s -nan ns
-Bandwidth / Latency (Bank 1): 0.000 GB/s -nan ns
-Bandwidth / Latency (Bank 2): 0.000 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#

```

# Result



Name	Size	Type	Modified
 k6_aoe_02_short.trc	2 items	Folder	15:15
 mase_art.trc	1 item	Folder	May 25

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  
IDD1=130  
IDD2P=10  
IDD2Q=70  
IDD2N=70  
IDD3Pf=60  
IDD3Ps=60  
IDD3N=90  
IDD4W=255  
IDD4R=230  
IDD5=305  
IDD6=9  
IDD6L=12  
IDD7=415  
Vdd=1.5  
!!EPOCH\_DATA  
ms,Background\_Power[0][0],ACT\_PRE\_Power[0][0],Burst\_Power[0][0],Refresh\_Power[0][0],Bandwidth[0][0][0],Average\_Latency[0][0][0],Bandwidth[0][0][1],Average\_Latency[0][0][1],Bandwidth[0][0][2],Average\_Latency[0][0][2],Bandwidth[0][0][3],Average\_Latency[0][0][3],Bandwidth[0][0][4],Average\_Latency[0][0][4],Bandwidth[0][0][5],Average\_Latency[0][0][5],Bandwidth[0][0][6],Average\_Latency[0][0][6],Bandwidth[0][0][7],Average\_Latency[0][0][7],Rank\_Aggregate\_Bandwidth[0][0],Rank\_Average\_Bandwidth[0][0],Aggregate\_Bandwidth[0],Average\_Bandwidth[0],  
0.0015,0.96192,0.15552,0.16368,0,0.0794729,62.25,0,-  
nan,0.0397364,39,0.0397364,43.5,0.0397364,43.5,0.119209,43.5,0.0397364,43.5,0.119209,43.5,0.476837,0.476837,0.476837,0.0596046,!!HISTOGRAM\_DATA  
20=9  
40=2



## Learning from this:

Able to get more information about the internal structure of memory and how it works.

How banks are mapped.

How address mapping is done.

Depending on the status of bank How much energy is used by banks.

And most importantly it gave a good amount of technical knowledge about the simulator. About it's functionality, about it working and its performance.





## What Work will be continued in future?

- After completing this project now I want to explore more it this area. And want to know how this can be used in distributed system. Where we can use multibank for multiprocessor environment.
- And also want to start building the simulator from scratch.
- Which will give more flexibility in term of checking the performance.

# Reference



- Journals & Magazines >IEEE Computer Architecture Le... >Volume: 10 Issue: 1  
DRAMSim2: A Cycle Accurate Memory System Simulator
- Hanene BEN FRADJ, Cécile BELLEUDY, Michel auguin Laboratoire d'informatique, Signaux et Systèmes de Sophia-Antipolis, Les Algorithmes-bat. Euclide 2000, route des Lucioles-BP 121, 06903 Sophia-Antipolis cedex. France {benfradj, belleudy, auguin}@i3s.unice.fr

Multi-Bank Main Memory Architecture with Dynamic Voltage Frequency Scaling for System Energy Optimization



# Thankyou

For More Reference: Please visit Github page by clicking image below

