Exploring Memory Technology Simulators

Reza Adinepour

Computer Engineering Department, Tehran Ploytechnic

adinepour@aut.ac.ir

May 9, 2024

Memory Technologies - Spring 2024

Amirkabir University of Technology
(Tehran Polytechnic)

Agenda

Introduction

Why we should use simulators?

1. Simulators are vital for understanding computer architecture
2. Two main categories:
   1. Memory simulators
      => focus solely on memory components
   2. Full-system simulators
      => emulate all computer components
3. Efficient design relies on effective simulation tools
4. Comprehensive insights through full-system simulation
5. Maximize performance with accurate simulators

Full-system simulators

Full-system simulators emulate the entire computer system, providing a holistic view. For example, we can refer to the following simulators:

1. GEM5
2. QEMU
3. Bochs
4. SimpleScalar

It's worth noting that a notable and highly regarded emulator in this field is **GEM5**.

Memory simulators

Memory simulators focus on simulating specific memory components. Examples include:

1. **CACTI**
2. **NVSIM**
3. **DRAMSim**
4. DiskSim
5. Ramulator
6. OpenRAM
7. HSPICE

In this talk, we will review the first 3 cases and **SimpleScalar** in the category of Full-system simulators.

CACTI

"In 1993, Dr. Jupi and Dr. Wilton pioneered the first simulation, and CACTI was subsequently developed through HP company tests."

1. Although this simulator simulates all memory levels, its main use is in the analysis of **Caches**
2. This simulator takes a set of memory parameters as input
3. It calculates various parameters such as **Access time**, **Power**, **Cycle time**, and **Area**
4. CACTI is available in two varieties: Web version and C++ Source code

Next, we will explain how to install and work with the uncompiled version of this emulator

CACTI (Cont.)

Advantages:

1. Open source
2. To be general
3. High speed
4. High flexibility in personalization

Disadvantages:

1. Approximate calculations
2. Productivity gap
3. Not real time
4. It doesn't have a strong community

CACTI (Cont.)

How install and compile CACTI?

In first we should install dependencies.

Install dependencies

```
$ sudo apt-get update
$ sudo apt-get install build-essential
```

After install dependencies we should clone repository.

Clone repository

```
$ git clone
https://github.com/HewlettPackard/cacti.git
```

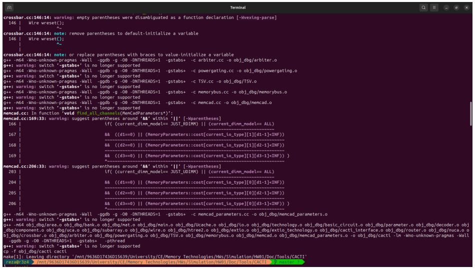
CACTI (Cont.)

Now we build CACTI:

Build

```
$ cd CACTI
$ make
```

After the build is completed, you will see an output like Figure 1

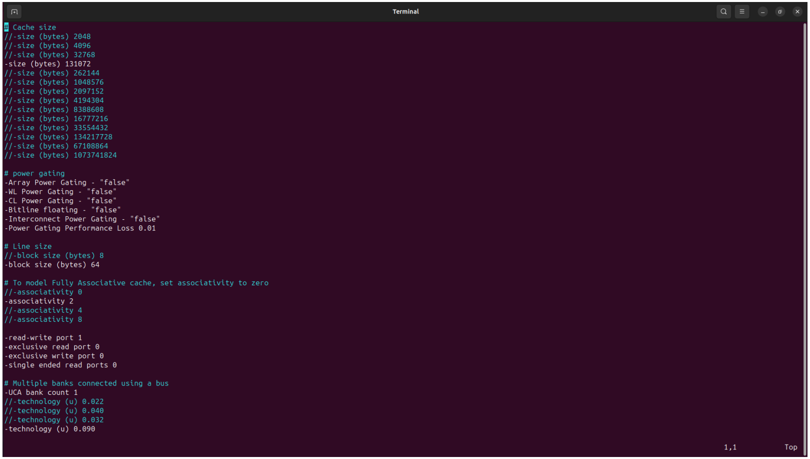


```
crossbar.cc:146:14: warning: empty parentheses were disambiguated as a function declaration [-Wvexing-parse]
  146 |   Wire wreset();
      |              ^~
crossbar.cc:146:14: note: remove parentheses to default-initialize a variable
  146 |   Wire wreset();
      |              ^~
      |              --
crossbar.cc:146:14: note: or replace parentheses with braces to value-initialize a variable
g++ -m64 -Wno-unknown-pragmas -Wall  -ggdb -g -O0 -DNTHREADS=1  -gstabs+  -c arbiter.cc -o obj_dbg/arbiter.o
g++: warning: switch '-gstabs+' is no longer supported
g++ -m64 -Wno-unknown-pragmas -Wall  -ggdb -g -O0 -DNTHREADS=1  -gstabs+  -c powergating.cc -o obj_dbg/powergating.o
g++: warning: switch '-gstabs+' is no longer supported
g++ -m64 -Wno-unknown-pragmas -Wall  -ggdb -g -O0 -DNTHREADS=1  -gstabs+  -c TSV.cc -o obj_dbg/TSV.o
g++: warning: switch '-gstabs+' is no longer supported
g++ -m64 -Wno-unknown-pragmas -Wall  -ggdb -g -O0 -DNTHREADS=1  -gstabs+  -c memorybus.cc -o obj_dbg/memorybus.o
g++: warning: switch '-gstabs+' is no longer supported
g++ -m64 -Wno-unknown-pragmas -Wall  -ggdb -g -O0 -DNTHREADS=1  -gstabs+  -c memcad.cc -o obj_dbg/memcad.o
g++: warning: switch '-gstabs+' is no longer supported
memcad.cc: In function 'void find_all_channels(MemCadParameters*)':
memcad.cc:169:33: warning: suggest parentheses around '&&' within '||' [-Wparentheses]
  166 |                                if( (current_dimm_model== JUST_RDIMM) || (current_dimm_model== ALL)
      |                                                                         ~~~~~~~~~~~~~~~~~~~~~~~~~~
  167 |                                &&  ((d1==0) || (MemoryParameters::cost[current_io_type][1][d1-1]<INF))
      |                                ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  168 |                                &&  ((d2==0) || (MemoryParameters::cost[current_io_type][1][d2-1]<INF))
      |                                ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  169 |                                &&  ((d3==0) || (MemoryParameters::cost[current_io_type][1][d3-1]<INF)) )
      |                                ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
memcad.cc:206:33: warning: suggest parentheses around '&&' within '||' [-Wparentheses]
  203 |                                if( (current_dimm_model== JUST_UDIMM) || (current_dimm_model== ALL)
      |                                                                         ~~~~~~~~~~~~~~~~~~~~~~~~~~
  204 |                                &&  ((d1==0) || (MemoryParameters::cost[current_io_type][0][d1-1]<INF))
      |                                ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  205 |                                &&  ((d2==0) || (MemoryParameters::cost[current_io_type][0][d2-1]<INF))
      |                                ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  206 |                                &&  ((d3==0) || (MemoryParameters::cost[current_io_type][0][d3-1]<INF)) )
      |                                ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
g++ -m64 -Wno-unknown-pragmas -Wall  -ggdb -g -O0 -DNTHREADS=1  -gstabs+  -c memcad_parameters.cc -o obj_dbg/memcad_parameters.o
g++: warning: switch '-gstabs+' is no longer supported
g++ -m64 obj_dbg/area.o obj_dbg/bank.o obj_dbg/mat.o obj_dbg/main.o obj_dbg/Ucache.o obj_dbg/io.o obj_dbg/technology.o obj_dbg/basic_circuit.o obj_dbg/parameter.o obj_dbg/decoder.o obj
_dbg/component.o obj_dbg/uca.o obj_dbg/subarray.o obj_dbg/wire.o obj_dbg/htree2.o obj_dbg/extio.o obj_dbg/extio_technology.o obj_dbg/cacti_interface.o obj_dbg/router.o obj_dbg/nuca.o o
bj_dbg/crossbar.o obj_dbg/arbiter.o obj_dbg/powergating.o obj_dbg/TSV.o obj_dbg/memorybus.o obj_dbg/memcad.o obj_dbg/memcad_parameters.o -o obj_dbg/cacti -lm -Wno-unknown-pragmas -Wall
  -ggdb -g -O0 -DNTHREADS=1  -gstabs+   -pthread
g++: warning: switch '-gstabs+' is no longer supported
cp -f obj_dbg/cacti cacti
make[1]: Leaving directory '/mnt/9636D17436D15639/University/CE/Memory Technologies/HWs/Simulation/HW01/Doc/Tools/CACTI'
 reza@r3z4  /mnt/9636D17436D15639/University/CE/Memory Technologies/HWs/Simulation/HW01/Doc/Tools/CACTI  master ?
```

Figure 1: Successful build

CACTI (Cont.)

You can set cache configs in `cache.cfg` file like figure



```
# Cache size
//-size (bytes) 2048
//-size (bytes) 4096
//-size (bytes) 32768
-size (bytes) 131072
//-size (bytes) 262144
//-size (bytes) 1048576
//-size (bytes) 2097152
//-size (bytes) 4194304
//-size (bytes) 8388608
//-size (bytes) 16777216
//-size (bytes) 33554432
//-size (bytes) 134217728
//-size (bytes) 67108864
//-size (bytes) 1073741824

# power gating
-Array Power Gating - "false"
-WL Power Gating - "false"
-CL Power Gating - "false"
-Bitline floating - "false"
-Interconnect Power Gating - "false"
-Power Gating Performance Loss 0.01

# Line size
//-block size (bytes) 8
-block size (bytes) 64

# To model Fully Associative cache, set associativity to zero
//-associativity 0
-associativity 2
//-associativity 4
//-associativity 8

-read-write port 1
-exclusive read port 0
-exclusive write port 0
-single ended read ports 0

# Multiple banks connected using a bus
-UCA bank count 1
//-technology (u) 0.022
//-technology (u) 0.040
//-technology (u) 0.032
-technology (u) 0.090
                                                            1,1           Top
```

Figure 2: cache config file

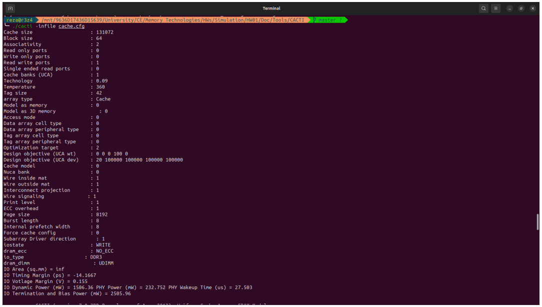
CACTI (Cont.)

Run simulation with this command:

Run

```
$ ./cacti -infile cache.cfg
```

The simulation output is as follows:



```
reza@r3z4  /mnt/9636D17436D15639/University/CE/Memory Technologies/HWs/Simulation/HW01/Doc/Tools/CACTI  master ?
└ ./cacti -infile cache.cfg
Cache size                    : 131072
Block size                    : 64
Associativity                 : 2
Read only ports               : 0
Write only ports              : 0
Read write ports              : 1
Single ended read ports       : 0
Cache banks (UCA)             : 1
Technology                    : 0.09
Temperature                   : 360
Tag size                      : 42
array type                    : Cache
Model as memory               : 0
Model as 3D memory               : 0
Access mode                   : 0
Data array cell type          : 0
Data array peripheral type    : 0
Tag array cell type           : 0
Tag array peripheral type     : 0
Optimization target           : 2
Design objective (UCA wt)     : 0 0 0 100 0
Design objective (UCA dev)    : 20 100000 100000 100000 100000
Cache model                   : 0
Nuca bank                     : 0
Wire inside mat               : 1
Wire outside mat              : 1
Interconnect projection       : 1
Wire signaling               : 1
Print level                   : 1
ECC overhead                  : 1
Page size                     : 8192
Burst length                  : 8
Internal prefetch width       : 8
Force cache config            : 0
Subarray Driver direction       : 1
iostate                       : WRITE
dram_ecc                      : NO_ECC
io_type                     : DDR3
dram_dimm                       : UDIMM
IO Area (sq.mm) = inf
IO Timing Margin (ps) = -14.1667
IO Votlage Margin (V) = 0.155
IO Dynamic Power (mW) = 1506.36 PHY Power (mW) = 232.752 PHY Wakeup Time (us) = 27.503
IO Termination and Bias Power (mW) = 2505.96
```

Figure 3: Output report

NVSIM

1. NVSIM simulator is a tool for analyzing and simulating non-volatile memories
2. It is primarily used for analyzing and estimating the area, power, and energy consumed
3. Unlike CACTI simulator, NVSIM simulator supports the simulation and analysis of new emerging memories like:
    1. PCM (Phase Change Memory)
    2. STT RAM (Spin Torque Transfer RAM)
    3. ReRAM (Resistive RAM)
    4. FBDRAM (Floating Body Dynamic RAM)
    5. eDRAM
4. Developed with C++

NVSIM (Cont.)

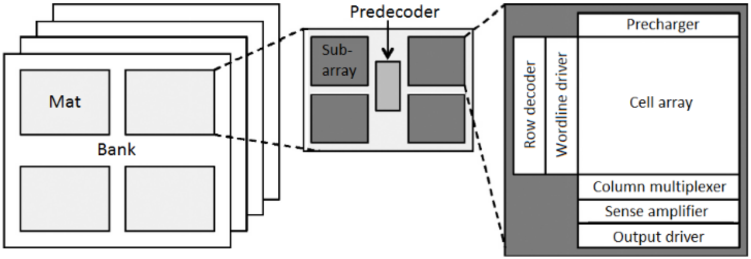


Figure 4: Memory hierarchy in NVSIM

NVSIM (Cont.)

Advantages:

1. Open source
2. Support for the simulation of emerging memories
3. low level Changeability and personalization

Disadvantages:

1. Not real time
2. There is no official version (In this talk i use modified version of simulator)

NVSIM (Cont.)

How install and compile CACTI?

In first clone repository:

clone repository

```
$ git clone
https://github.com/lpentecost/nvsim-merged
```

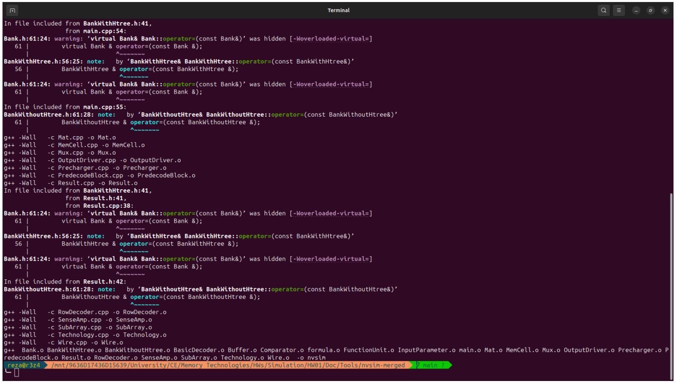
go to repository directory and make it:

build

```
$ cd nvsim-merged
$ make
```

NVSIM (Cont.)

If the build is successful, your terminal output will look like this:

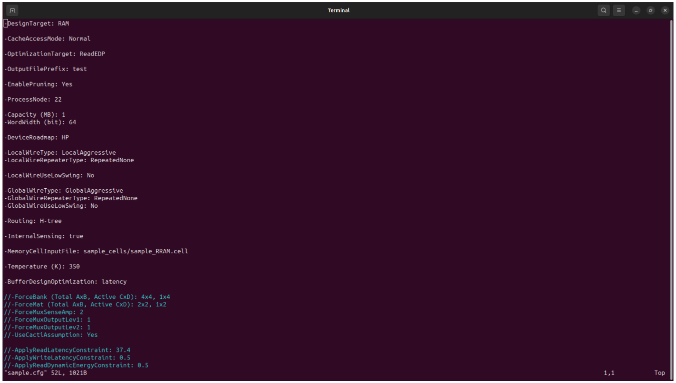


```
In file included from BankWithHtree.h:41,
                 from main.cpp:54:
Bank.h:61:24: warning: ‘virtual Bank& Bank::operator=(const Bank&)’ was hidden [-Woverloaded-virtual=]
   61 |         virtual Bank & operator=(const Bank &);
      |                        ^~~~~~~~
BankWithHtree.h:56:25: note:   by ‘BankWithHtree& BankWithHtree::operator=(const BankWithHtree&)’
   56 |         BankWithHtree & operator=(const BankWithHtree &);
      |                         ^~~~~~~~
Bank.h:61:24: warning: ‘virtual Bank& Bank::operator=(const Bank&)’ was hidden [-Woverloaded-virtual=]
   61 |         virtual Bank & operator=(const Bank &);
      |                        ^~~~~~~~
In file included from main.cpp:55:
BankWithoutHtree.h:61:28: note:   by ‘BankWithoutHtree& BankWithoutHtree::operator=(const BankWithoutHtree&)’
   61 |         BankWithoutHtree & operator=(const BankWithoutHtree &);
      |                            ^~~~~~~~
g++ -Wall   -c Mat.cpp -o Mat.o
g++ -Wall   -c MemCell.cpp -o MemCell.o
g++ -Wall   -c Mux.cpp -o Mux.o
g++ -Wall   -c OutputDriver.cpp -o OutputDriver.o
g++ -Wall   -c Precharger.cpp -o Precharger.o
g++ -Wall   -c PredecodeBlock.cpp -o PredecodeBlock.o
g++ -Wall   -c Result.cpp -o Result.o
In file included from BankWithHtree.h:41,
                 from Result.h:41,
                 from Result.cpp:38:
Bank.h:61:24: warning: ‘virtual Bank& Bank::operator=(const Bank&)’ was hidden [-Woverloaded-virtual=]
   61 |         virtual Bank & operator=(const Bank &);
      |                        ^~~~~~~~
BankWithHtree.h:56:25: note:   by ‘BankWithHtree& BankWithHtree::operator=(const BankWithHtree&)’
   56 |         BankWithHtree & operator=(const BankWithHtree &);
      |                         ^~~~~~~~
Bank.h:61:24: warning: ‘virtual Bank& Bank::operator=(const Bank&)’ was hidden [-Woverloaded-virtual=]
   61 |         virtual Bank & operator=(const Bank &);
      |                        ^~~~~~~~
In file included from Result.h:42:
BankWithoutHtree.h:61:28: note:   by ‘BankWithoutHtree& BankWithoutHtree::operator=(const BankWithoutHtree&)’
   61 |         BankWithoutHtree & operator=(const BankWithoutHtree &);
      |                            ^~~~~~~~
g++ -Wall   -c RowDecoder.cpp -o RowDecoder.o
g++ -Wall   -c SenseAmp.cpp -o SenseAmp.o
g++ -Wall   -c SubArray.cpp -o SubArray.o
g++ -Wall   -c Technology.cpp -o Technology.o
g++ -Wall   -c Wire.cpp -o Wire.o
g++  Bank.o BankWithHtree.o BankWithoutHtree.o BasicDecoder.o Buffer.o Comparator.o formula.o FunctionUnit.o InputParameter.o main.o Mat.o MemCell.o Mux.o OutputDriver.o Precharger.o P
redecodeBlock.o Result.o RowDecoder.o SenseAmp.o SubArray.o Technology.o Wire.o  -o nvsim
 reza@r3z4  /mnt/9636D17436D15639/University/CE/Memory Technologies/HWs/Simulation/HW01/Doc/Tools/nvsim-merged  main ? 
╰─
```

Figure 5: NVSIM successful build

NVSIM (Cont.)

Now we should set the config file like CACTI in `.cfg` file. for simulate simple design we use `samole.cfg` which the config of a 64 bit memristor.



```
-DesignTarget: RAM

-CacheAccessMode: Normal

-OptimizationTarget: ReadEDP

-OutputFilePrefix: test

-EnablePruning: Yes

-ProcessNode: 22

-Capacity (MB): 1
-WordWidth (bit): 64

-DeviceRoadmap: HP

-LocalWireType: LocalAggressive
-LocalWireRepeaterType: RepeatedNone

-LocalWireUseLowSwing: No

-GlobalWireType: GlobalAggressive
-GlobalWireRepeaterType: RepeatedNone
-GlobalWireUseLowSwing: No

-Routing: H-tree

-InternalSensing: true

-MemoryCellInputFile: sample_cells/sample_RRAM.cell

-Temperature (K): 350

-BufferDesignOptimization: latency

//-ForceBank (Total AxB, Active CxD): 4x4, 1x4
//-ForceMat (Total AxB, Active CxD): 2x2, 1x2
//-ForceMuxSenseAmp: 2
//-ForceMuxOutputLev1: 1
//-ForceMuxOutputLev2: 1
//-UseCactiAssumption: Yes

//-ApplyReadLatencyConstraint: 37.4
//-ApplyWriteLatencyConstraint: 0.5
//-ApplyReadDynamicEnergyConstraint: 0.5
"sample.cfg" 52L, 1021B                                                                                                    1,1           Top
```

Figure 6: `sample.cfg` config file

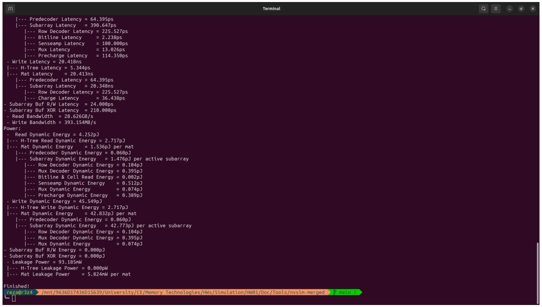
NVSIM (Cont.)

Run simulation:

Run

```
$ ./nvsim sample.cfg
```

output as follow:



```
   |--- Predecoder Latency = 64.395ps
   |--- Subarray Latency   = 390.647ps
      |--- Row Decoder Latency = 225.527ps
      |--- Bitline Latency     = 2.238ps
      |--- Senseamp Latency    = 100.000ps
      |--- Mux Latency         = 13.026ps
      |--- Precharge Latency   = 114.350ps
 - Write Latency = 20.418ns
 |--- H-Tree Latency = 5.344ps
 |--- Mat Latency    = 20.413ns
   |--- Predecoder Latency = 64.395ps
   |--- Subarray Latency   = 20.348ns
      |--- Row Decoder Latency = 225.527ps
      |--- Charge Latency      = 36.438ps
- Subarray Buf R/W Latency  = 24.000ps
- Subarray Buf XOR Latency  = 210.000ps
 - Read Bandwidth  = 28.626GB/s
 - Write Bandwidth = 393.154MB/s
Power:
 -  Read Dynamic Energy = 4.252pJ
 |--- H-Tree Read Dynamic Energy = 2.717pJ
 |--- Mat Dynamic Energy    = 1.536pJ per mat
   |--- Predecoder Dynamic Energy = 0.060pJ
   |--- Subarray Dynamic Energy   = 1.476pJ per active subarray
      |--- Row Decoder Dynamic Energy = 0.104pJ
      |--- Mux Decoder Dynamic Energy = 0.395pJ
      |--- Bitline & Cell Read Energy = 0.002pJ
      |--- Senseamp Dynamic Energy    = 0.512pJ
      |--- Mux Dynamic Energy         = 0.074pJ
      |--- Precharge Dynamic Energy   = 0.389pJ
 - Write Dynamic Energy = 45.549pJ
 |--- H-Tree Write Dynamic Energy = 2.717pJ
 |--- Mat Dynamic Energy    = 42.832pJ per mat
   |--- Predecoder Dynamic Energy = 0.060pJ
   |--- Subarray Dynamic Energy   = 42.773pJ per active subarray
      |--- Row Decoder Dynamic Energy = 0.104pJ
      |--- Mux Decoder Dynamic Energy = 0.395pJ
      |--- Mux Dynamic Energy         = 0.074pJ
- Subarray Buf R/W Energy = 0.000pJ
- Subarray Buf XOR Energy = 0.000pJ
 - Leakage Power = 93.185mW
 |--- H-Tree Leakage Power = 0.000pW
 |--- Mat Leakage Power    = 5.824mW per mat

Finished!
reza@r3z4 > /mnt/9636D17436D15639/University/CE/Memory Technologies/HWs/Simulation/HW01/Doc/Tools/nvsim-merged > main ?
╰ 
```

Figure 7: Output of simulation

References

Back to start

[1] S. Senni, *Exploration of non-volatile magnetic memory for processor architecture*, 2015.

[2] N. Muralimanohar, R. Balasubramonian, and N. P. Jouppi, "Cacti 6.0: A tool to understand large caches," *University of Utah and Hewlett Packard Laboratories, Tech. Rep*, vol. 147, 2009.

[3] P. Rosenfeld, E. Cooper-Balis, and B. Jacob, "Dramsim2: A cycle accurate memory system simulator," *IEEE computer architecture letters*, vol. 10, no. 1, pp. 16–19, 2011.

[4] S. Li, Z. Yang, D. Reddy, A. Srivastava, and B. Jacob, "Dramsim3: A cycle-accurate, thermal-capable dram simulator," *IEEE Computer Architecture Letters*, vol. 19, no. 2, pp. 106–109, 2020.

To be continued

Questions? Comments?

You can find this slides here:

github.com/M-Sc-AUT/M.Sc-Computer-Architecture/Memory Technologies