A Manual on use of DESTINY Tool for Architectural Studies

Abstract

DESTINY is a tool for modeling 3D caches designed with SRAM, eDRAM, STT-RAM, PCM and ReRAM. This manual explains some of the internals and parameters of DESTINY. It shows a typical configuration file and the corresponding output, which may be especially useful for a user who wants to get an overview of DESTINY even before installing it. This manual also provides ideas and suggestions for integrating DESTINY into architectural simulators and points towards future work.

This manual is expected to be useful for researchers to assess the scope and potential of DESTINY and help them in using DESTINY to drive architectural studies. This manual is also expected to answer some of the "frequently asked questions" related to DESTINY.

CONTENTS

I. Integrating DESTINY into an Architectural Simulator

Note that DESTINY is NOT a timing simulator like Sniper, GEM5, SimpleScalar, MARSS etc. Rather, DESTINY is analogous to CACTI, NVSim etc. DESTINY provides area, timing and energy parameters for the cache.

There are at least three ways of integrating DESTINY into architectural simulators.

1) The code of DESTINY can be interfaced directly with that of a simulator. The inputs which are generally given to DESTINY as .cfg file can instead be passed through suitable function calls. Although this option allows highest flexibility, it is likely to be difficult and time-consuming.
2) DESTINY can be integrated in a tool like McPAT to model caches.
3) By separately running DESTINY code, the required parameters can be found, which be stored in a file. Later on, the simulator can read these values as a lookup table and use them for timing simulation. The energy values can be computed, for example, by multiplying number of reads/writes with corresponding read/write access energy values. This is a simple option, but does not allow as much flexibility as the previous options.

Note that DESTINY only models cache energy and to model the energy consumption of other components such as main memory and core, other tools may be required.

A. Accounting for read-write asymmetric memories

Non-volatile memories such as STT-RAM, ReRAM and PCM have different latencies and access energy values for reads and writes. For such memories, due care needs to be taken while using DESTINY output in an architectural simulator, since the values for read and write need to be separately replaced.

II. Illustration of Typical Input and Output of DESTINY

The first Listing below shows a typical input configuration file for DESTINY and the second Listing below shows the output of DESTINY for this configuration file.

Listing 1: Configuration (input) file for a 3D ReRAM cache

```
-DesignTarget: cache
-ProcessNode: 32
-Capacity (MB): 16
-WordWidth (bit): 512
-Associativity (for cache only): 16

-PrintLevel: 1

-DeviceRoadmap: HP

-LocalWireType: LocalConservative
-LocalWireRepeaterType: RepeatedNone
-LocalWireUseLowSwing: No

-GlobalWireType: GlobalConservative
-GlobalWireRepeaterType: RepeatedNone
-GlobalWireUseLowSwing: No

-Routing: H-tree
//-Routing: Non-H-tree
//-InternalSensing: false

-MemoryCellInputFile: sample_RRAM.cell

-Temperature (K): 350

//-OptimizationTarget: WriteLatency
//-OptimizationTarget: Full
```

```
-OptimizationTarget: WriteEDP

-BufferDesignOptimization: balanced

//-ForceBankA (Total AxB): 64x128
//-ForceBank (Total AxB, Active CxD): 8x16x4, 8x4
//-ForceMatA (Total AxB): 1x1
//-ForceMat (Total AxB, Active CxD): 2x2, 1x2
//-ForceMuxSenseAmp: 32
//-ForceMuxOutputLev1: 1
//-ForceMuxOutputLev2: 1

-StackedDieCount: 2
-MonolithicStackCount: 1
```

Listing 2: Parameters (output) obtained from DESTINY for above configuration file

```
User-defined configuration file (Input_3DReRAM.cfg) is loaded
====================
DESIGN SPECIFICATION
====================
Design Target: Cache
Capacity    : 16MB
Cache Line Size: 64Bytes
Cache Associativity: 16 Ways

Searching for the best solution that is optimized for write energy-delay-product ...
Using cell file: sample_RRAM.cell
numSolutions = 123506 / numDesigns = 15604974

=======================
CACHE DESIGN -- SUMMARY
=======================
Access Mode: Normal
Area:
 - Total Area = 2.195mm^2
 |--- Data Array Area = 946.864um x 539.400um = 1.913mm^2
 |--- Tag Array Area  = 477.882um x 379.131um = 0.282mm^2
Timing:
 - Cache Hit Latency    = 11.692ns
 - Cache Miss Latency   = 2.028ns
 - Cache Write Latency = 14.650ns
Power:
 - Cache Hit Dynamic Energy    = 0.445nJ per access
 - Cache Miss Dynamic Energy   = 0.445nJ per access
 - Cache Write Dynamic Energy = 0.164nJ per access
 - Cache Total Leakage Power   = 135.741mW
 |--- Cache Data Array Leakage Power = 57.675mW
 |--- Cache Tag Array Leakage Power  = 78.066mW

CACHE DATA ARRAY DETAILS
    =============
       SUMMARY
    =============
    Optimized for: Write Energy-Delay-Product
    Memory Cell: RRAM (Memristor)
    Cell Area (F^2)     : 4.000 (2.000Fx2.000F)
    Cell Aspect Ratio   : 1.000
    Cell Turned-On Resistance : 1.000Mohm
    Cell Turned-Off Resistance: 10.000Mohm
    Read Mode: Current-Sensing
      - Read Voltage: 0.300V
    Reset Mode: Voltage
```

```
  - Reset Voltage: 2.000V
  - Reset Pulse: 4.000ns
Set Mode: Voltage
  - Set Voltage: 2.000V
  - Set Pulse: 4.000ns
Access Type: Diode

=============
CONFIGURATION
=============
Bank Organization: 4 x 1 x 2
 - Row Activation    : 2 / 4 x 1
 - Column Activation: 1 / 1 x 1
Mat Organization: 2 x 2
 - Row Activation    : 2 / 2
 - Column Activation: 2 / 2
 - Subarray Size     : 1024 Rows x 4096 Columns
Mux Level:
 - Senseamp Mux       : 4
 - Output Level-1 Mux: 1
 - Output Level-2 Mux: 1
 - One set is partitioned into 1 rows
Local Wire:
 - Wire Type :      Local Conservative
 - Repeater Type:      No Repeaters
 - Low Swing :      No
Global Wire:
 - Wire Type :      Global Conservative
 - Repeater Type:      No Repeaters
 - Low Swing :      No
Buffer Design Style:      Balanced
=============
   RESULT
=============
Area:
 - Total Area = 946.864um x 539.400um = 1.913mm^2
 |--- Mat Area       = 236.716um x 539.400um = 127684.538um^2   (107.639%)
 |--- Subarray Area = 118.358um x 265.604um = 31436.314um^2   (109.300%)
 |--- TSV Area       = 900.000um^2
 - Area Efficiency = 28.739%
Timing:
 -  Read Latency = 6.766ns
 |--- TSV Latency     = 0.010ps
 |--- H-Tree Latency = 50.185ps
 |--- Mat Latency     = 6.715ns
    |--- Predecoder Latency = 84.055ps
    |--- Subarray Latency   = 6.631ns
       |--- Row Decoder Latency = 5.421ns
       |--- Bitline Latency     = 115.197ps
       |--- Senseamp Latency    = 1.073ns
       |--- Mux Latency         = 22.311ps
       |--- Precharge Latency   = 525.993ps
 - Write Latency = 14.650ns
 |--- TSV Latency     = 0.005ps
 |--- H-Tree Latency = 25.093ps
 |--- Mat Latency     = 14.625ns
    |--- Predecoder Latency = 84.055ps
    |--- Subarray Latency   = 14.541ns
       |--- Row Decoder Latency = 5.421ns
       |--- Charge Latency      = 807.650ps
 - Read Bandwidth  = 36.864GB/s
 - Write Bandwidth = 4.401GB/s
Power:
```

```
      -  Read Dynamic Energy = 377.269pJ
      |--- TSV Dynamic Energy     = 29.955pJ
      |--- H-Tree Dynamic Energy = 52.428pJ
      |--- Mat Dynamic Energy     = 147.443pJ per mat
         |--- Predecoder Dynamic Energy = 0.238pJ
         |--- Subarray Dynamic Energy   = 36.801pJ per active subarray
            |--- Row Decoder Dynamic Energy = 0.397pJ
            |--- Mux Decoder Dynamic Energy = 0.792pJ
            |--- Bitline & Cell Read Energy = -0.185pJ
            |--- Senseamp Dynamic Energy    = 32.337pJ
            |--- Mux Dynamic Energy         = 0.529pJ
            |--- Precharge Dynamic Energy   = 2.930pJ
      - Write Dynamic Energy = 135.726pJ
      |--- TSV Dynamic Energy     = 29.955pJ
      |--- H-Tree Dynamic Energy = 52.428pJ
      |--- Mat Dynamic Energy     = 26.671pJ per mat
         |--- Predecoder Dynamic Energy = 0.238pJ
         |--- Subarray Dynamic Energy   = 6.608pJ per active subarray
            |--- Row Decoder Dynamic Energy = 0.397pJ
            |--- Mux Decoder Dynamic Energy = 0.792pJ
            |--- Mux Dynamic Energy         = 0.529pJ
      - Leakage Power = 57.675mW
      |--- TSV Leakage               = 0.000pW
      |--- H-Tree Leakage Power      = 0.000pW
      |--- Mat Leakage Power         = 7.209mW per mat

CACHE TAG ARRAY DETAILS
     =============
        SUMMARY
     =============
     Optimized for: Write Energy-Delay-Product
     Memory Cell: RRAM (Memristor)
     Cell Area (F^2)     : 4.000 (2.000Fx2.000F)
     Cell Aspect Ratio   : 1.000
     Cell Turned-On Resistance : 1.000Mohm
     Cell Turned-Off Resistance: 10.000Mohm
     Read Mode: Current-Sensing
       - Read Voltage: 0.300V
     Reset Mode: Voltage
       - Reset Voltage: 2.000V
       - Reset Pulse: 4.000ns
     Set Mode: Voltage
       - Set Voltage: 2.000V
       - Set Pulse: 4.000ns
     Access Type: Diode

     =============
     CONFIGURATION
     =============
     Bank Organization: 2 x 1 x 2
      - Row Activation    : 1 / 2 x 1
      - Column Activation: 1 / 1 x 1
     Mat Organization: 2 x 2
      - Row Activation    : 2 / 2
      - Column Activation: 1 / 2
      - Subarray Size     : 256 Rows x 1920 Columns
     Mux Level:
      - Senseamp Mux       : 8
      - Output Level-1 Mux: 1
      - Output Level-2 Mux: 1
      - One set is partitioned into 1 rows
     Local Wire:
      - Wire Type :      Local Aggressive
```

```
 - Repeater Type:      No Repeaters
 - Low Swing :      No
Global Wire:
 - Wire Type :      Global Aggressive
 - Repeater Type:      No Repeaters
 - Low Swing :      Yes
Buffer Design Style:      Latency-Optimized
==============
   RESULT
==============
Area:
 - Total Area = 477.882um x 379.131um = 281979.865um^2
 |--- Mat Area       = 238.941um x 379.131um = 90589.933um^2    (17.779%)
 |--- Subarray Area = 119.180um x 184.128um = 21944.375um^2    (18.349%)
 |--- TSV Area       = 900.000um^2
 - Area Efficiency = 11.424%
Timing:
 -  Read Latency = 2.028ns
 |--- TSV Latency     = 0.010ps
 |--- H-Tree Latency = 379.767ps
 |--- Mat Latency     = 1.648ns
    |--- Predecoder Latency = 98.405ps
    |--- Subarray Latency   = 1.526ns
       |--- Row Decoder Latency = 436.013ps
       |--- Bitline Latency     = 5.708ps
       |--- Senseamp Latency    = 1.073ns
       |--- Mux Latency         = 11.370ps
       |--- Precharge Latency   = 121.199ps
    |--- Comparator Latency  = 24.237ps
 - Write Latency = 8.787ns
 |--- TSV Latency     = 0.005ps
 |--- H-Tree Latency = 189.883ps
 |--- Mat Latency     = 8.597ns
    |--- Predecoder Latency = 98.405ps
    |--- Subarray Latency   = 8.499ns
       |--- Row Decoder Latency = 436.013ps
       |--- Charge Latency      = 62.965ps
 - Read Bandwidth  = 3.097GB/s
 - Write Bandwidth = 441.230MB/s
Power:
 -  Read Dynamic Energy = 68.054pJ
 |--- TSV Dynamic Energy     = 2.540pJ
 |--- H-Tree Dynamic Energy = 0.581pJ
 |--- Mat Dynamic Energy     = 64.933pJ per mat
    |--- Predecoder Dynamic Energy = 1.032pJ
    |--- Subarray Dynamic Energy   = 31.950pJ per active subarray
       |--- Row Decoder Dynamic Energy = 0.208pJ
       |--- Mux Decoder Dynamic Energy = 0.435pJ
       |--- Bitline & Cell Read Energy = -0.695pJ
       |--- Senseamp Dynamic Energy    = 30.316pJ
       |--- Mux Dynamic Energy         = 0.302pJ
       |--- Precharge Dynamic Energy   = 1.384pJ
 - Write Dynamic Energy = 28.312pJ
 |--- TSV Dynamic Energy     = 2.540pJ
 |--- H-Tree Dynamic Energy = 0.581pJ
 |--- Mat Dynamic Energy     = 25.191pJ per mat
    |--- Predecoder Dynamic Energy = 1.032pJ
    |--- Subarray Dynamic Energy   = 12.079pJ per active subarray
       |--- Row Decoder Dynamic Energy = 0.208pJ
       |--- Mux Decoder Dynamic Energy = 0.435pJ
       |--- Mux Dynamic Energy         = 0.302pJ
 - Leakage Power = 78.066mW
 |--- TSV Leakage               = 0.000pW
```

```
      |--- H-Tree Leakage Power      = 1.396mW
      |--- Mat Leakage Power         = 18.819mW per mat

Finished!
```

A. *Understanding DESTINY output*

Listing 2 above provides the detailed output. It first provides the summary for whole cache and then its individual breakdown for data array and tag array, respectively. Listing 3 below shows the selected output from Listing 2. Most researchers who don't want to worry about breakdown of results into tag/array will just be interested in the results shown in Listing 3.

Listing 3: Selected output from Listing 2

```
=======================
CACHE DESIGN -- SUMMARY
=======================
Access Mode: Normal
Area:
 - Total Area = 2.195mm^2
Timing:
 - Cache Hit Latency    = 11.692ns
 - Cache Miss Latency   = 2.028ns
 - Cache Write Latency = 14.650ns
Power:
 - Cache Hit Dynamic Energy    = 0.445nJ per access
 - Cache Miss Dynamic Energy   = 0.445nJ per access
 - Cache Write Dynamic Energy = 0.164nJ per access
 - Cache Total Leakage Power   = 135.741mW
```

To use timing parameters of DESTINY, note that in sequetial cache access mode, we have

$$\text{Cache Hit Latency} = \text{Tag lookup latency + data read access latency} \tag{1}$$

$$\text{Cache Miss Latency} = \text{Tag lookup latency} \tag{2}$$

$$\text{Cache Write Latency} = \text{Tag lookup latency + data write access latency} \tag{3}$$

III. Use of DESTINY to model assist structures

DESTINY is useful to model not only the regular caches, but 'assist structures' also. For example, some works use profiling cache [1, 2], victim cache [3], write buffer etc. These additional structures are typically small-sized and are useful for some profiling work, prediction work (e.g. dead-block prediction), keeping the evicted blocks, etc. These structures are used as part of a broader architectural technique which may be proposed to improve cache lifetime, performance, energy etc. To comprehensively account for the overhead of that technique, it is important to account for area, timing and energy overhead of these assist structures.

By either modifying the DESTINY code suitably or by specifying their parameters as such, a user may be able to model these structures in DESTINY. Also, DESTINY provides separate parameters for tag and data arrays and this is userful for gaining more insight and seeing the fractional contribution of both arrays.

References

[1] S. Mittal *et al.*, "EnCache: Improving cache energy efficiency using a software-controlled profiling cache," in *IEEE International Conference On Electro/Information Technology*, 2012.

[2] S. Mittal *et al.*, "EnCache: A Dynamic Profiling Based Reconfiguration Technique for Improving Cache Energy Efficiency," *Journal of Circuits, Systems, and Computers*, vol. 23, no. 10, 2014.

[3] N. P. Jouppi, "Improving direct-mapped cache performance by the addition of a small fully-associative cache and prefetch buffers," in *Computer Architecture, 1990. Proceedings., 17th Annual International Symposium on*, 1990, pp. 364–373.