

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 [1, 2]. 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	2
I-A	Accounting for read-write asymmetric memories	2
II	Illustration of Typical Input and Output of DESTINY	2
II-A	Understanding DESTINY output	7
III	Use of DESTINY to model assist structures	7
IV	Advantage of comprehensiveness of DESTINY	7

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

```
1 -DesignTarget: cache
2 -ProcessNode: 32
3 -Capacity (MB): 16
4 -WordWidth (bit): 512
5 -Associativity (for cache only): 16
6
7 -PrintLevel: 1
8
9 -DeviceRoadmap: HP
10
11 -LocalWireType: LocalConservative
12 -LocalWireRepeaterType: RepeatedNone
13 -LocalWireUseLowSwing: No
14
15 -GlobalWireType: GlobalConservative
16 -GlobalWireRepeaterType: RepeatedNone
17 -GlobalWireUseLowSwing: No
18
19 -Routing: H-tree
20 // -Routing: Non-H-tree
21 // -InternalSensing: false
22
23 -MemoryCellInputFile: sample_RRAM.cell
24
25 -Temperature (K): 350
26
27 // -OptimizationTarget: WriteLatency
28 // -OptimizationTarget: Full
```

```

29 -OptimizationTarget: WriteEDP
31 -BufferDesignOptimization: balanced
33 // -ForceBankA (Total AxB): 64x128
34 // -ForceBank (Total AxB, Active CxD): 8x16x4, 8x4
35 // -ForceMatA (Total AxB): 1x1
36 // -ForceMat (Total AxB, Active CxD): 2x2, 1x2
37 // -ForceMuxSenseAmp: 32
38 // -ForceMuxOutputLev1: 1
39 // -ForceMuxOutputLev2: 1
41 -StackedDieCount: 2
42 -MonolithicStackCount: 1

```

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

```

User-defined configuration file (Input_3DReRAM.cfg) is loaded
2 =====
DESIGN SPECIFICATION
4 =====
Design Target: Cache
6 Capacity : 16MB
Cache Line Size: 64Bytes
8 Cache Associativity: 16 Ways

10 Searching for the best solution that is optimized for write energy-delay-product ...
Using cell file: sample_RRAM.cell
12 numSolutions = 123506 7 numDesigns = 15604974

14 =====
CACHE DESIGN — SUMMARY
16 =====
Access Mode: Normal
18 Area:
- Total Area = 2.195mm^2
20 |--- Data Array Area = 946.864um x 539.400um = 1.913mm^2
|--- Tag Array Area = 477.882um x 379.131um = 0.282mm^2
22 Timing:
- Cache Hit Latency = 11.692ns
24 - Cache Miss Latency = 2.028ns
- Cache Write Latency = 14.650ns
26 Power:
- Cache Hit Dynamic Energy = 0.445nJ per access
28 - Cache Miss Dynamic Energy = 0.445nJ per access
- Cache Write Dynamic Energy = 0.164nJ per access
30 - Cache Total Leakage Power = 135.741mW
|--- Cache Data Array Leakage Power = 57.675mW
32 |--- Cache Tag Array Leakage Power = 78.066mW

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

```

48 - Reset Voltage: 2.000V
 - Reset Pulse: 4.000ns
Set Mode: Voltage
50 - Set Voltage: 2.000V
 - Set Pulse: 4.000ns
52 Access Type: Diode

54 =====

55 CONFIGURATION

56 =====

Bank Organization: 4 x 1 x 2
58 - Row Activation : 2 / 4 x 1
 - Column Activation: 1 / 1 x 1
60 Mat Organization: 2 x 2
 - Row Activation : 2 / 2
62 - Column Activation: 2 / 2
 - Subarray Size : 1024 Rows x 4096 Columns
64 Mux Level:
 - Senseamp Mux : 4
66 - Output Level-1 Mux: 1
 - Output Level-2 Mux: 1
68 - One set is partitioned into 1 rows

Local Wire:

70 - Wire Type : Local Conservative
 - Repeater Type: No Repeaters
72 - Low Swing : No

Global Wire:

74 - Wire Type : Global Conservative
 - Repeater Type: No Repeaters
76 - Low Swing : No

Buffer Design Style: Balanced

78 =====

80 RESULT

80 =====

Area:

82 - Total Area = 946.864um x 539.400um = 1.913mm²
 |--- Mat Area = 236.716um x 539.400um = 127684.538um² (107.639%)
84 |--- Subarray Area = 118.358um x 265.604um = 31436.314um² (109.300%)
 |--- TSV Area = 900.000um²
86 - Area Efficiency = 28.739%

Timing:

88 - Read Latency = 6.766ns
 |--- TSV Latency = 0.010ps
90 |--- H-Tree Latency = 50.185ps
 |--- Mat Latency = 6.715ns
92 |--- Predecoder Latency = 84.055ps
 |--- Subarray Latency = 6.631ns
94 |--- Row Decoder Latency = 5.421ns
 |--- Bitline Latency = 115.197ps
96 |--- Senseamp Latency = 1.073ns
 |--- Mux Latency = 22.311ps
98 |--- Precharge Latency = 525.993ps
 - Write Latency = 14.650ns
100 |--- TSV Latency = 0.005ps
 |--- H-Tree Latency = 25.093ps
102 |--- Mat Latency = 14.625ns
 |--- Predecoder Latency = 84.055ps
104 |--- Subarray Latency = 14.541ns
 |--- Row Decoder Latency = 5.421ns
106 |--- Charge Latency = 807.650ps
 - Read Bandwidth = 36.864GB/s
108 - Write Bandwidth = 4.401GB/s

Power:

```

110 - Read Dynamic Energy = 377.269pJ
111 |--- TSV Dynamic Energy = 29.955pJ
112 |--- H-Tree Dynamic Energy = 52.428pJ
113 |--- Mat Dynamic Energy = 147.443pJ per mat
114 |---- Predecoder Dynamic Energy = 0.238pJ
115 |---- Subarray Dynamic Energy = 36.801pJ per active subarray
116 |---- Row Decoder Dynamic Energy = 0.397pJ
117 |---- Mux Decoder Dynamic Energy = 0.792pJ
118 |---- Bitline & Cell Read Energy = -0.185pJ
119 |---- Senseamp Dynamic Energy = 32.337pJ
120 |---- Mux Dynamic Energy = 0.529pJ
121 |---- Precharge Dynamic Energy = 2.930pJ
122 - Write Dynamic Energy = 135.726pJ
123 |--- TSV Dynamic Energy = 29.955pJ
124 |--- H-Tree Dynamic Energy = 52.428pJ
125 |--- Mat Dynamic Energy = 26.671pJ per mat
126 |---- Predecoder Dynamic Energy = 0.238pJ
127 |---- Subarray Dynamic Energy = 6.608pJ per active subarray
128 |---- Row Decoder Dynamic Energy = 0.397pJ
129 |---- Mux Decoder Dynamic Energy = 0.792pJ
130 |---- Mux Dynamic Energy = 0.529pJ
131 - Leakage Power = 57.675mW
132 |--- TSV Leakage = 0.000pW
133 |--- H-Tree Leakage Power = 0.000pW
134 |--- Mat Leakage Power = 7.209mW per mat

```

136 CACHE TAG ARRAY DETAILS

137 =====

138 SUMMARY

139 =====

```

140 Optimized for: Write Energy-Delay-Product
141 Memory Cell: RRAM (Memristor)
142 Cell Area (F^2) : 4.000 (2.000Fx2.000F)
143 Cell Aspect Ratio : 1.000
144 Cell Turned-On Resistance : 1.000Mohm
145 Cell Turned-Off Resistance: 10.000Mohm
146 Read Mode: Current-Sensing
147 - Read Voltage: 0.300V
148 Reset Mode: Voltage
149 - Reset Voltage: 2.000V
150 - Reset Pulse: 4.000ns
151 Set Mode: Voltage
152 - Set Voltage: 2.000V
153 - Set Pulse: 4.000ns
154 Access Type: Diode

```

155 =====

156 CONFIGURATION

157 =====

```

158 Bank Organization: 2 x 1 x 2
159 - Row Activation : 1 / 2 x 1
160 - Column Activation: 1 / 1 x 1
161 Mat Organization: 2 x 2
162 - Row Activation : 2 / 2
163 - Column Activation: 1 / 2
164 - Subarray Size : 256 Rows x 1920 Columns
165 Mux Level:
166 - Senseamp Mux : 8
167 - Output Level-1 Mux: 1
168 - Output Level-2 Mux: 1
169 - One set is partitioned into 1 rows
170 Local Wire:
171 - Wire Type : Local Aggressive

```

```

174 - Repeater Type:      No Repeaters
175 - Low Swing :       No
Global Wire:
176 - Wire Type :       Global Aggressive
177 - Repeater Type:    No Repeaters
178 - Low Swing :       Yes
Buffer Design Style:    Latency-Optimized

```

===== RESULT =====

Area:

```

184 - Total Area = 477.882um x 379.131um = 281979.865um^2
185 |--- Mat Area      = 238.941um x 379.131um = 90589.933um^2    (17.779%)
186 |--- Subarray Area = 119.180um x 184.128um = 21944.375um^2    (18.349%)
187 |--- TSV Area      = 900.000um^2
188 - Area Efficiency = 11.424%

```

Timing:

```

190 - Read Latency = 2.028ns
191 |--- TSV Latency   = 0.010ps
192 |--- H-Tree Latency = 379.767ps
193 |--- Mat Latency   = 1.648ns
194 |--- Predecoder Latency = 98.405ps
195 |--- Subarray Latency = 1.526ns
196 |--- Row Decoder Latency = 436.013ps
197 |--- Bitline Latency = 5.708ps
198 |--- Senseamp Latency = 1.073ns
199 |--- Mux Latency    = 11.370ps
200 |--- Precharge Latency = 121.199ps
201 |--- Comparator Latency = 24.237ps
202 - Write Latency = 8.787ns
203 |--- TSV Latency   = 0.005ps
204 |--- H-Tree Latency = 189.883ps
205 |--- Mat Latency   = 8.597ns
206 |--- Predecoder Latency = 98.405ps
207 |--- Subarray Latency = 8.499ns
208 |--- Row Decoder Latency = 436.013ps
209 |--- Charge Latency = 62.965ps
210 - Read Bandwidth = 3.097GB/s
211 - Write Bandwidth = 441.230MB/s

```

Power:

```

212 - Read Dynamic Energy = 68.054pJ
213 |--- TSV Dynamic Energy = 2.540pJ
214 |--- H-Tree Dynamic Energy = 0.581pJ
215 |--- Mat Dynamic Energy = 64.933pJ per mat
216 |--- Predecoder Dynamic Energy = 1.032pJ
217 |--- Subarray Dynamic Energy = 31.950pJ per active subarray
218 |--- Row Decoder Dynamic Energy = 0.208pJ
219 |--- Mux Decoder Dynamic Energy = 0.435pJ
220 |--- Bitline & Cell Read Energy = -0.695pJ
221 |--- Senseamp Dynamic Energy = 30.316pJ
222 |--- Mux Dynamic Energy = 0.302pJ
223 |--- Precharge Dynamic Energy = 1.384pJ
224 - Write Dynamic Energy = 28.312pJ
225 |--- TSV Dynamic Energy = 2.540pJ
226 |--- H-Tree Dynamic Energy = 0.581pJ
227 |--- Mat Dynamic Energy = 25.191pJ per mat
228 |--- Predecoder Dynamic Energy = 1.032pJ
229 |--- Subarray Dynamic Energy = 12.079pJ per active subarray
230 |--- Row Decoder Dynamic Energy = 0.208pJ
231 |--- Mux Decoder Dynamic Energy = 0.435pJ
232 |--- Mux Dynamic Energy = 0.302pJ
233 - Leakage Power = 78.066mW
234 |--- TSV Leakage = 0.000pW

```

```

236 | --- H-Tree Leakage Power      = 1.396mW
    | --- Mat Leakage Power       = 18.819mW per mat
238 |
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

```

1  =====
  CACHE DESIGN — SUMMARY
3  =====
  Access Mode: Normal
5  Area:
  - Total Area = 2.195mm^2
7  Timing:
  - Cache Hit Latency   = 11.692ns
  - Cache Miss Latency  = 2.028ns
  - Cache Write Latency = 14.650ns
11 Power:
  - Cache Hit Dynamic Energy = 0.445nJ per access
13 - Cache Miss Dynamic Energy = 0.445nJ per access
  - Cache Write Dynamic Energy = 0.164nJ per access
15 - Cache Total Leakage Power = 135.741mW

```

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

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

$$\text{Cache Miss Latency} = \text{Tag lookup latency} \quad (2)$$

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

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 [3, 4], victim cache [5], 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 useful for gaining more insight and seeing the fractional contribution of both arrays.

IV. ADVANTAGE OF COMPREHENSIVENESS OF DESTINY

Study and comparison of different memory technologies (e.g. SRAM, NVMs) is desirable for finding the most suitable one for a given workload or usage scenario. In absence of a comprehensive modeling tool that can model all memory technologies simultaneously, researchers have different tools for modeling them, for example, CACTI for

modeling SRAM and NVSim for modeling NVMs and in-house tools for modeling eDRAM etc. (e.g. [6]). However, there are several challenges and pitfalls in such an approach:

- Different tools have different modeling assumptions, output format and even different output values for same input configuration [1].
- Over time, these tools undergo revisions (due to more accurate modeling, bug fixes and other improvements), and in such a case, their cross-comparison becomes difficult.
- Use of in-house tools precludes reproducing the results.

Due to these factors, the conclusions derived from these studies may be inaccurate and misleading.

By comparison, DESTINY is a comprehensive tool which can model 5 memory technologies (and we plan to include other memory technologies soon). This allows meaningful comparison and a single code-base for improvement. It also reduces the effort on part of researchers/users to configure the tool properly.

REFERENCES

- [1] S. Mittal *et al.*, “Exploring Design Space of 3D NVM and eDRAM Caches Using DESTINY Tool,” Oak Ridge National Laboratory, USA, Tech. Rep. ORNL/TM-2014/636, 2014.
- [2] M. Poremba *et al.*, “DESTINY: A Tool for Modeling Emerging 3D NVM and eDRAM caches,” in *Design Automation and Test in Europe*, 2015.
- [3] S. Mittal *et al.*, “EnCache: Improving cache energy efficiency using a software-controlled profiling cache,” in *IEEE International Conference On Electro/Information Technology*, 2012.
- [4] 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.
- [5] 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.
- [6] S.-M. Syu *et al.*, “High-endurance hybrid cache design in CMP architecture with cache partitioning and access-aware policy,” in *23rd ACM international conference on Great lakes symposium on VLSI*, 2013, pp. 19–24.