User Manual of MATLAB Thermal Modeling Framework V1.0

Developers: Ankit Kaul, Yang Zhang, and Madison Manley

PI: Dr. Muhannad Bakir, Georgia Institute of Technology

December 2, 2022

1 CONTENTS

2 INTRODUCTION

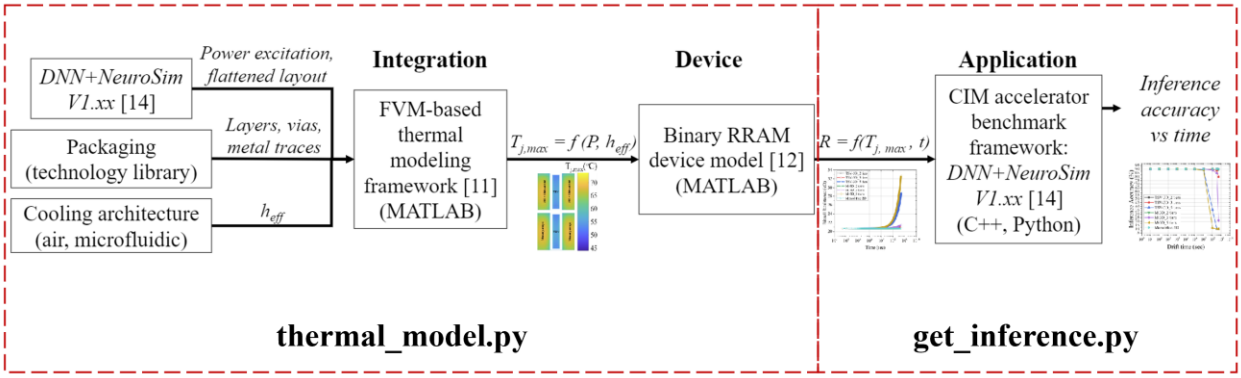

Fig. 1. Device-integration-application-level 3D CIM reliability evaluation flow. (Note: $T_{j,max}$ = Memory tier maximum junction temperature (^oC); P=Total package power (W); h_{eff}:=Effective heat transfer coefficient of heat sink ($W/m^2.\ ^oC$); R=RRAM device resistance (Ω), t=time (sec).

This thermal modeling framework, which is developed in MATLAB and wrapped by python scripts, is used to quantify the impact of 3D integration design parameters on CIM inference accuracy [1]. This flow combines a finite volume method (FVM)-based thermal analysis framework with a measurement-calibrated binary RRAM retention model and a CIM inference

accuracy estimation framework, to perform a device-integration-application-level reliability evaluation. Details of each flow component are described as follows:

2.1 THERMAL ANALYSIS

A modified version of the FVM-based thermal modeling framework described in [2] was used to model the 3D heterogeneous integration structures and perform steady state thermal simulations. Inputs for this step include a flattened layout of the modeled CIM IP (such as memory array, ADCs, etc.), power excitation maps for each active tier/die based on the flattened layout, and a description of the die stack-up, i.e. the bulk material, interconnects, and dielectrics along with their thermal properties (thermal conductivity, specific heat capacity, etc.). For validation, the memory tier maximum junction temperatures $T_{j,max}$ from our thermal models were compared with finite-element ANSYS Mechanical APDL models (ver. 2021 R1). The maximum deviation in $T_{j,max}$ and $T_{j,min}$ between our models and ANSYS was 3.3%.

2.2 RRAM RETENTION

The memory tier $T_{j,max}$, as a function of integration design parameters such as architecture, number of tiers, input power, boundary conditions, etc., is an input to the second part of this flow that consists of a measurement-calibrated HfO_2-based binary RRAM device model [3]. This analytical model is used to estimate device resistance variation over time, which is converted to a resistance drift ratio. The drift ratio is calculated as $\frac{R_{10}-R_{on}}{R_{on}}$, where R_{on} is the device low resistance state (LRS) resistance and R_{10} is the device resistance at 10 years operating at a specific junction temperature.

2.3 CIM INFERENCE ACCURACY ESTIMATION

Device retention change in RRAM-based CIM inference or training accelerators can correspond to a change in locally stored DNN weights/inputs/outpus (for weight-, input- or output-stationary dataflows [4], respectively), thus affecting the accuracy of an inference operation. The device retention drift ratio obtained from the previous step is used to adjust the drift coefficient of a retention model within DNN+NeuroSimV1.xx (a popular framework to benchmark CIM accelerators [5])**,** to estimate the variation of CIM inference accuracy.

3 SYSTEM REQUIREMENTS (LINUX)

The tool is expected to run in Linux with the required system dependencies installed. These include MATLAB, the matlab.engine python library, and dependencies needed for the DNN+NeuroSim Framework V1.3. We have tested the compatibility of the tool with the Linux environment Ubuntu 22 using MATLAB v2022b with python v3.8.15 and DNN+NeuroSim Framework V1.3.

The matlab.engine library is dependent on the MATLAB and python version. Please make sure that the MATLAB and python versions are compatible or else the matlab.engine library will not be supported.

4 INSTALLATION

Step 1: Clone repo from Github

```
git clone https://github.com/i3dsystems/3D_CIM_thermal_v1.0.git
```

Step 2: Clone DNN+NeuroSim Framework V1.3 into the same directory as the Thermal Modeling software.

```
git clone https://github.com/neurosim/DNN_NeuroSim_V1.3.git
```

Step 3: Download the MATLAB engine

- Directions can be found here.
- Make sure that the MATLAB and python version are compatible here.

5 HOW TO RUN

1. Run thermal_model.py

After setting up the parameters, the user will run the thermal model. In this file, the user can choose to do a design space exploration. All dimensions are in micrometers. An explanation of the areas the user can explore:

0 - Single run: One simulation with the current design.
1 – chip thickness: Changes the chip thickness of all dies.
2 – chip size: Changes the chip size in the x and y dimension for all dies.
3 – TSV Diameter: Changes the TSV diameter for all dies with TSVs.
4 – TSV Pitch: Changes the chip pitch in the x and y dimension for all dies with TSVs.
5 – Bump Diameter: Changes the bump diameter for all dies.
6 – Bump Pitch: Changes the bump pitch in the x and y dimension for all dies.

Example run: `python thermal_model.py 1 200 400`

- Run the thermal model for die thickness at 200um and 400um

The program will print out the results for each simulation, which will include the $T_{j,min}$ and $T_{j,max}$ for each chip and will calculate the drift coefficient based on the device model. The screenshot below shows an example output. A **run_summary.txt** file is created that saves the last design space exploration the user has run.

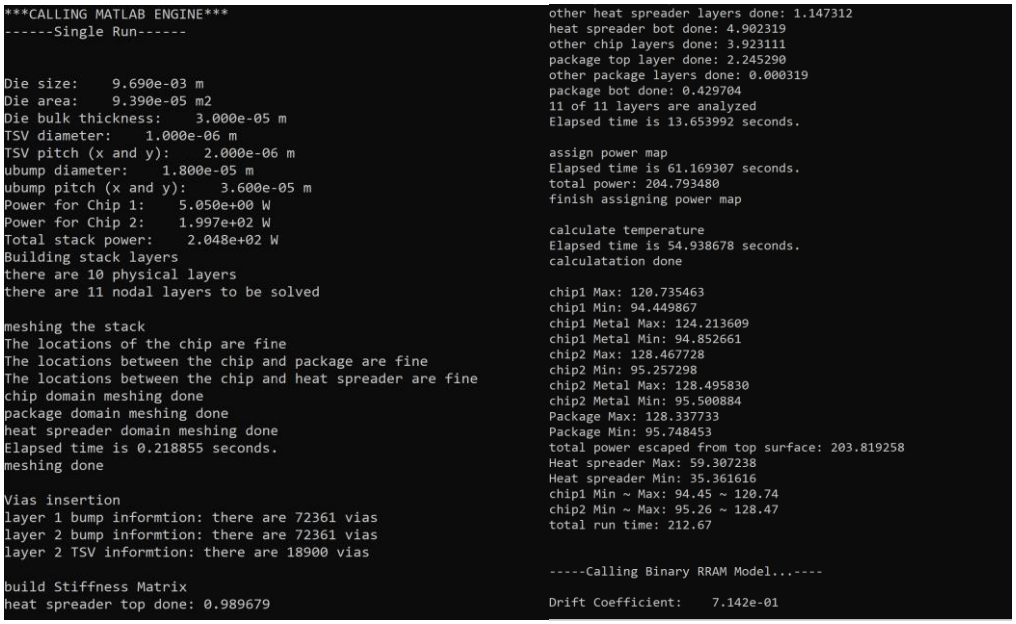

Fig. 2. Sample output from thermal_model.py

2. Run get_inference.py

After running the thermal model, the user needs to run the DNN+NeuroSim program. If the user did not place the DNN+NeuroSim folder into the thermal modeling working directory, the user will need to go into the neurosim.sh shell and change the path to the DNN+NeuroSim folder.

```
# Path to NeuroSim
cd "DNN_NeuroSim_V1.3/Inference_pytorch"
```

Fig. 3. Change the path to neurosim in neurosim.sh shell

Once get_inference.py is called, the script will get the inference accuracy from the DNN+NeuroSim program for the last design space exploration the user has run. The console will be printed to the folder **AccuracyOutput/console/<name.txt>** where **<name.txt>** is the last option the user has run with the thermal_model.py.

Last thermal_model.py run by user: `python thermal_model.py 1 200 400`

- Runs the thermal model for die thickness at 200um and 400um
- `python get_inference.py`
- Text files created in **AccuracyOutput/console/**
  - **1_200.txt**
  - **1_400.txt**

The console will output information in the NeuroSim simulation. The inference accuracy will be printed to the text files found in **AccuracyOutput/console/**

An example of the console output is below:

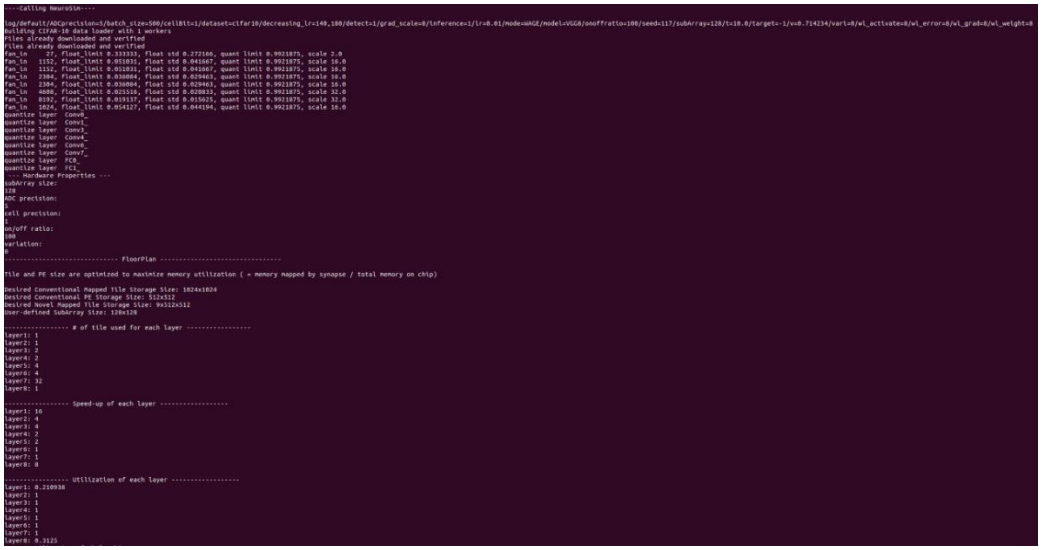

Fig. 4. Sample console output

An example of the <name.txt> output is below:

```
1 ==================FLAGS==================
2 dataset: cifar10
3 model: VGG8
4 mode: WAGE
5 batch_size: 500
6 epochs: 200
7 grad_scale: 8
8 seed: 117
9 log_interval: 100
10 test_interval: 1
11 logdir: log/default/ADCprecision=5/batch_size=500/cellBit=1/dataset=cifar10/decreasing_lr=140,180/detect=1/grad_scale=8/
   inference=1/lr=0.01/mode=WAGE/model=VGG8/onoffratio=100/seed=117/subArray=128/t=10.0/target=-1/v=0.714234/vari=0/
   wl_activate=8/wl_error=8/wl_grad=8/wl_weight=8
12 lr: 0.01
13 decreasing_lr: 140,180
14 wl_weight: 8
15 wl_grad: 8
16 wl_activate: 8
17 wl_error: 8
18 gpu: 0
19 inference: 1
20 subArray: 128
21 ADCprecision: 5
22 cellBit: 1
23 onoffratio: 100
24 vari: 0
25 t: 10.0
26 v: 0.714234
27 detect: 1
28 target: -1
29 =======================================
30 Test set: Average loss: 2.3026, Accuracy: 1000/10000 (10%)
31 ==================FLAGS==================
32 dataset: cifar10
33 model: VGG8
34 mode: WAGE
35 batch_size: 500
36 epochs: 200
37 grad_scale: 8
38 seed: 117
39 log_interval: 100
40 test_interval: 1
41 logdir: log/default/ADCprecision=5/batch_size=500/cellBit=1/dataset=cifar10/decreasing_lr=140,180/detect=1/grad_scale=8/
   inference=1/lr=0.01/mode=WAGE/model=VGG8/onoffratio=100/seed=117/subArray=128/t=100.0/target=-1/v=0.714234/vari=0/
   wl_activate=8/wl_error=8/wl_grad=8/wl_weight=8
42 lr: 0.01
43 decreasing_lr: 140,180
44 wl_weight: 8
45 wl_grad: 8
46 wl_activate: 8
```

Fig. 5. <name.txt> sample output. The inference accuracy is highlighted in red.

6 REFERENCES

[1] A. Kaul, Y. Luo, X. Peng, S. Yu, and M. S. Bakir, “Thermal reliability considerations of resistive synaptic devices for 3d cim system performance,” in 2021 IEEE International 3D Systems Integration Conference (3DIC), 2021, pp. 1–5. [Online]. Available: 10.1109/3DIC52383.2021.9687612

[2] Y. Zhang, Y. Zhang, and M. S. Bakir, “Thermal design and constraints for heterogeneous integrated chip stacks and isolation technology using air gap and thermal bridge,” IEEE Transactions on Components, Packaging and Manufacturing Technology, vol. 4, no. 12, pp. 1914–1924, 2014. [Online]. Available: 10.1109/TCPMT.2014.2364742

[3] P.-Y. Chen and S. Yu, “Compact modeling of rram devices and its applications in 1t1r and 1s1r array design,” IEEE Trans. Electron Devices, vol. 62, no. 12, pp. 4022–4028, 2015. [Online]. Available: 10.1109/TED.2015.2492421

[4] Y.-H. Chen, J. Emer, and V. Sze, “Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks,” in Proceedings of the 43rd International Symposium on Computer Architecture, ser. ISCA ’16. IEEE Press, 2016, p. 367–379. [Online]. Available: https://doi.org/ 10.1109/ISCA.2016.40

[5] X. Peng, S. Huang, Y. Luo, X. Sun, and S. Yu, “Dnn+neurosim: An end-to-end benchmarking framework for compute-in-memory accelerators with versatile device technologies,” in Proc. IEEE Int. Elec. Devices Meeting, 2019, pp. 32.5.1–32.5.4. [Online]. Available: 10.1109/IEDM19573.2019.8993491