CSE530 Project 1

Name- Niramay Vaidya
PSU ID- 939687597
Date- 11/10/21
Time- 01:00 PM

Table of Contents

1. Mat Mul CUDA Kernel-

The only major change as compared to the mat add kernel is the need of a loop for the k index in order to perform the mat mul operation c[i][j] += a[i][k] * b[k][j], where i and j are set to thread ids within blocks. The distribution of work between threads across blocks is such that there is one thread responsible for one (i, j) pair i.e. the total number of threads is equal to the number of values within one matrix.

2. GTX480 Floor Plan-

1. Layer 0

Since a 3D stacked die is to be simulated, usually how this is done in GPUs is that HBMs (High Bandwidth Memories), the equivalent of DRAMs, are stacked across multiple silicon layers with thermal interfacing layers in between, whereas all the rest of the components are kept on the bottom layer i.e. the 1st silicon layer. In this case, considering layer 2 as the bottom silicon layer, layer 2 is where the stacking of DRAMs should start, and layer 0 should also have them. Hence, DRAMs have been placed on layer 0 but have been named MEMs instead.

2. Layer 2

According to the explanation provided above, all the rest of the components along with the DRAMs have been placed on layer 2. This includes the L2 cache and the components within each SM (Streaming Multiprocessor). Certain components within a single SM have been combined to design a floorplan with a coarser granularity (for example, the cores, SFUs (Special Function Units), the warp schedulers, etc.). Also, some components within the SM are left out of the floorplan since there are no power values generated from gpgpusim that can be linked to them (for example, LD/ST (Load/Store Units)).

3. Power Report Log to Ptrace Converter-

The python script reads the power log, extracts all the power values from it (avg, max, min) for all the components, and combines certain components' powers to match the components that are actually present in this GPU (this is done so that there is no power value that is left unaccounted for). Since for smaller matrix sizes, the power values for certain components show anomalous behavior where the avg value is smaller than the max value, the script determines the true ordering between these min, avg and max values for each component. Since these power values are accumulated for multiple instances of the same component, they are divided by the appropriate scaling factor depending upon how many instances of each component there are in the floorplan.

4. Automation Script-

1. Required Flow

The required flow the end-to-end automation script must provide is that it takes in the power log file and the layer id, and then generates the corresponding heat map image. Along with these inputs, this script also takes in another input which is the tensor dimension (this is used in order to save the heat map image in an appropriate location within the created folder structure).

```
Usage: source ./automate.sh <tensor_dimension> <layer_id>
<power_report_log_file>
tensor_dimension > 0
layer_id = 0/2
```

Note- See prerequisites for running the script mentioned in the comments at the beginning of the script file itself.

2. Extra Flow

Apart from the required flow for the automation script, another script has been provided which extends the end-to-end functionality by taking in just the tensor dimension and the layer id, and then generating the corresponding heat map image, which means that it also handles the execution of the mat mul kernel for the given tensor dimension and generation of the gpgpusim power log file. Beyond this, the flow is the same as that supported by the script explained before. One important point to note for this script is that it tracks the tensor dimension used for the previous execution, so if it is the same for the current execution, the mat mul kernel need not be re-executed. Accordingly, the script skips doing this and directly moves on to the heat map image generation.

```
Usage: source ./automate.sh <tensor_dimension> <layer_id>
tensor_dimension > 0
layer_id = 0/2
```

Note- See prerequisites for running the script mentioned in the comments at the beginning of the script file itself.

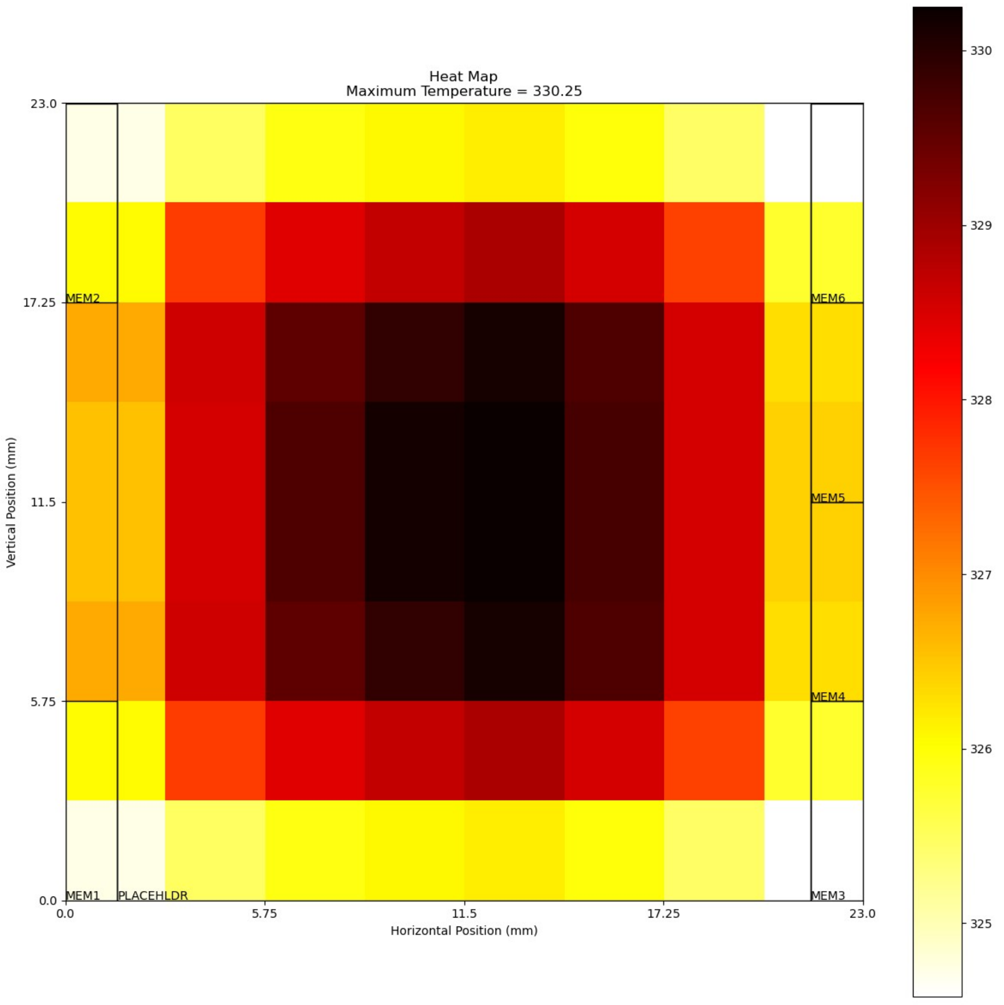
5. Results-

1. Original L2 Size

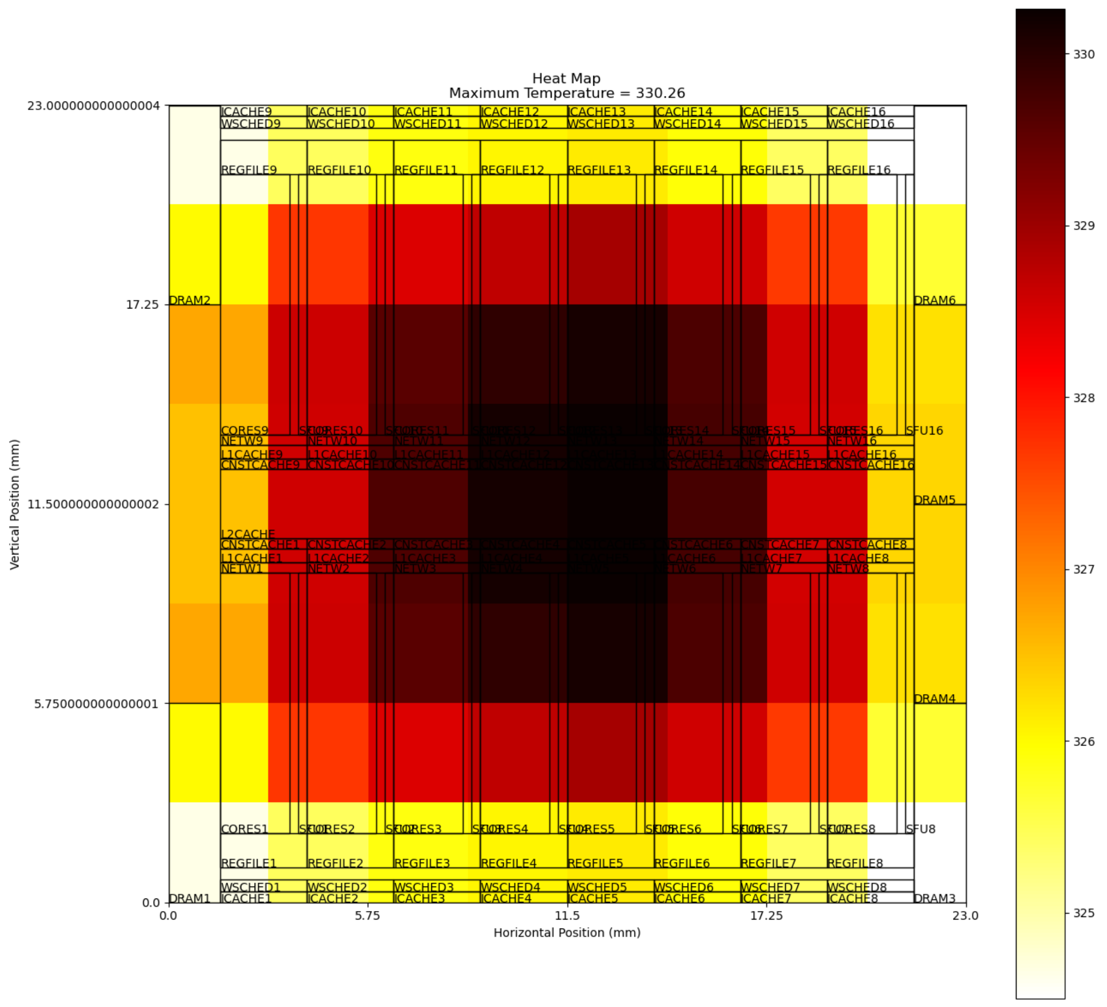
Tensor dimensions 10, 50, 100, 200, 300 and 400 (matrix sizes) were simulated and for all these simulations, the L2 cache (having size 768 KB) accesses weren't overflowing to the DRAM since the entire input for all these tensor dimensions was fitting into L2 itself.

1. Matrix size 10*10-

Layer 0

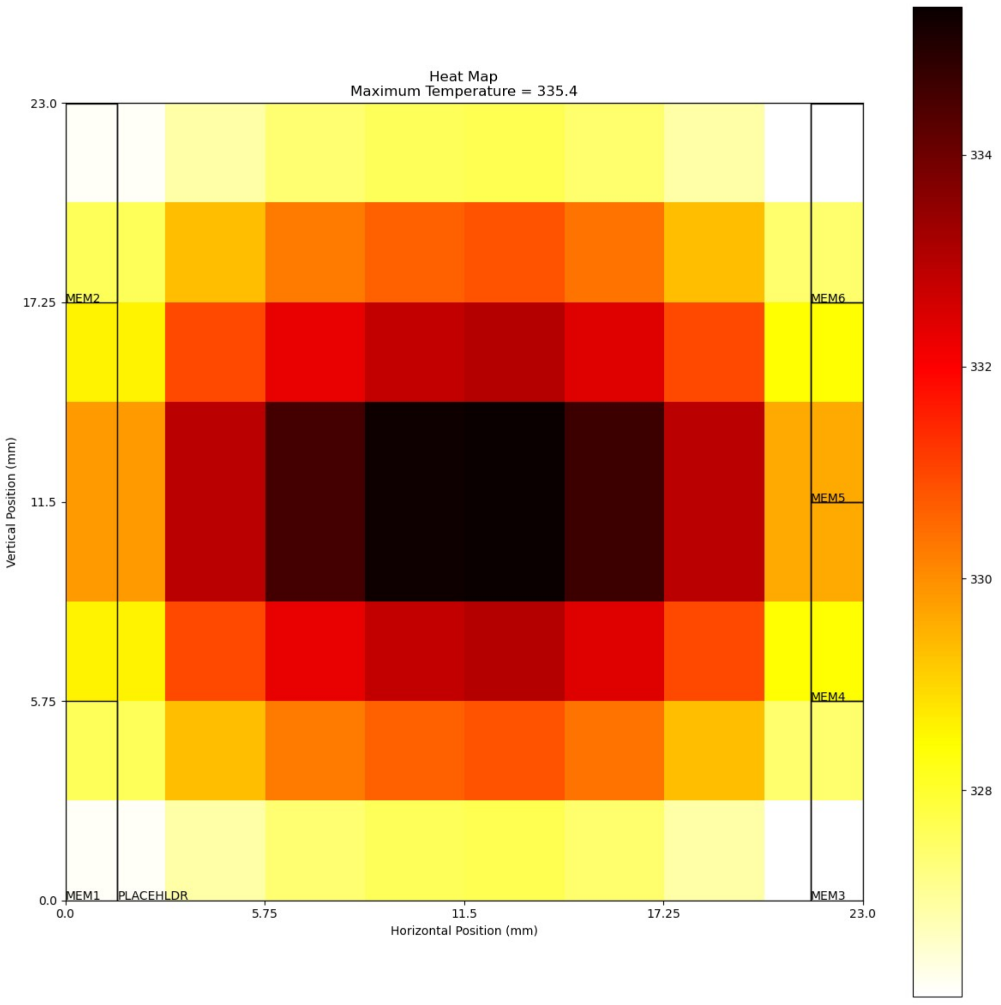

Heat Map
Maximum Temperature = 330.25
Vertical Position (mm)
Horizontal Position (mm)
0.0
5.75
11.5
17.25
23.0
MEM1
MEM2
MEM3
MEM4
MEM5
MEM6
PLACEHLDR
325
326
327
328
329
330

Layer 2



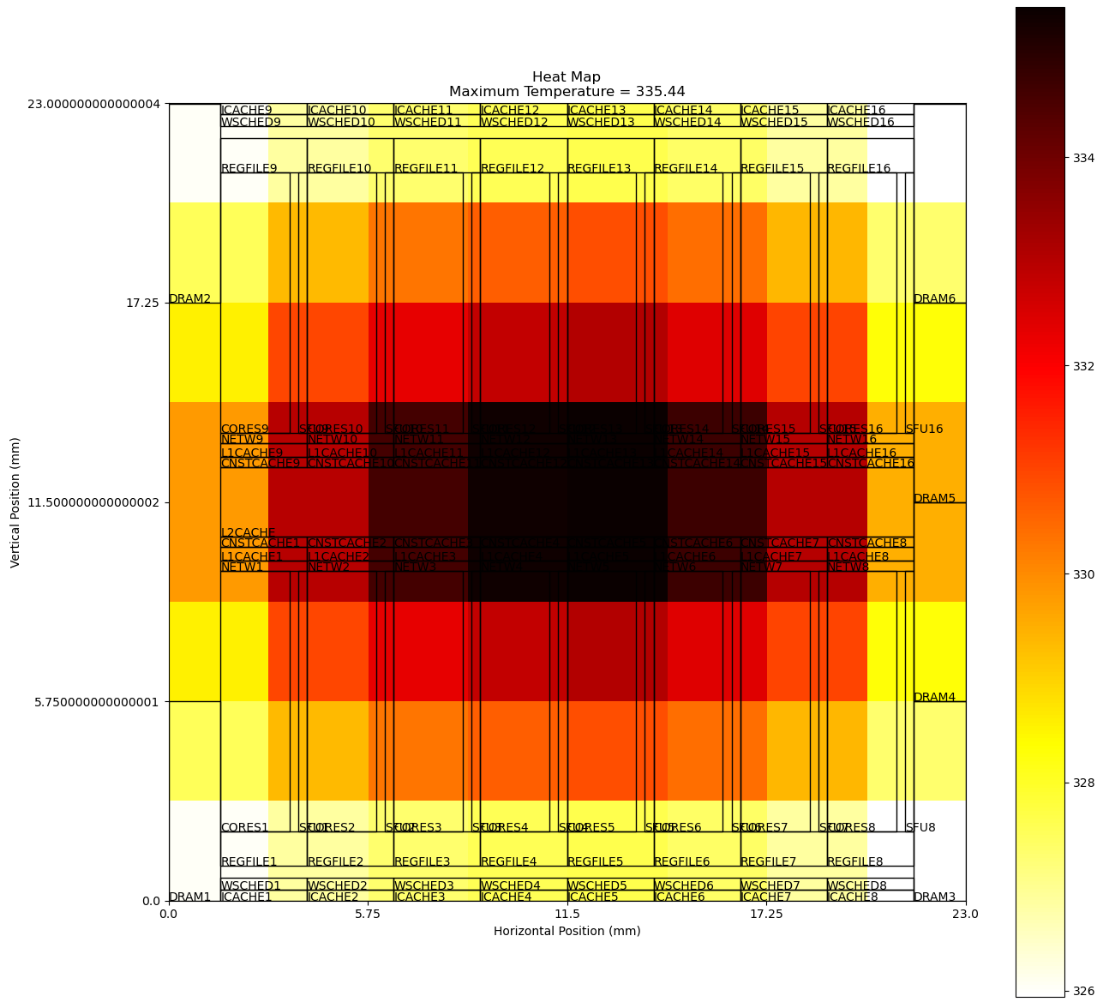
2. Matrix size 50*50-

Layer 0



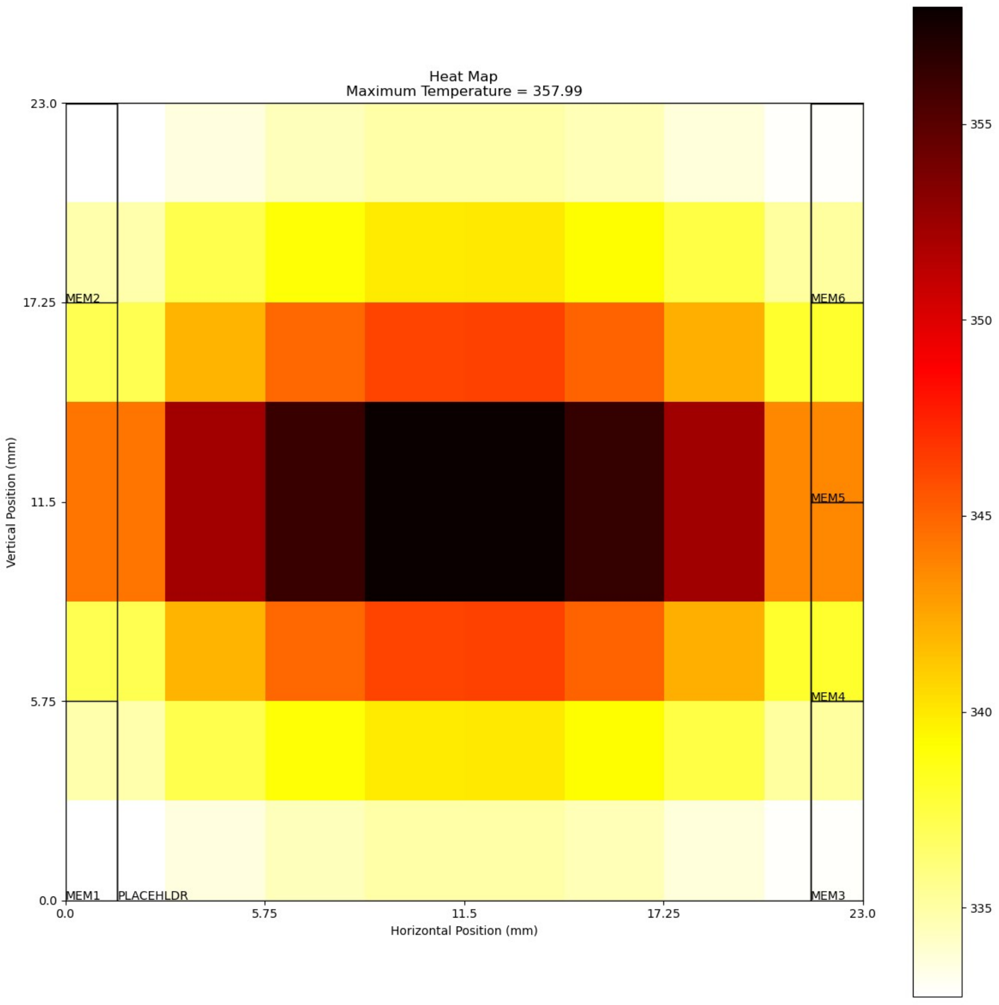
Heat Map
Maximum Temperature = 335.4
Vertical Position (mm)
Horizontal Position (mm)
0.0
5.75
11.5
17.25
23.0
MEM1
MEM2
MEM3
MEM4
MEM5
MEM6
PLACEHLDR
328
330
332
334

Layer 2

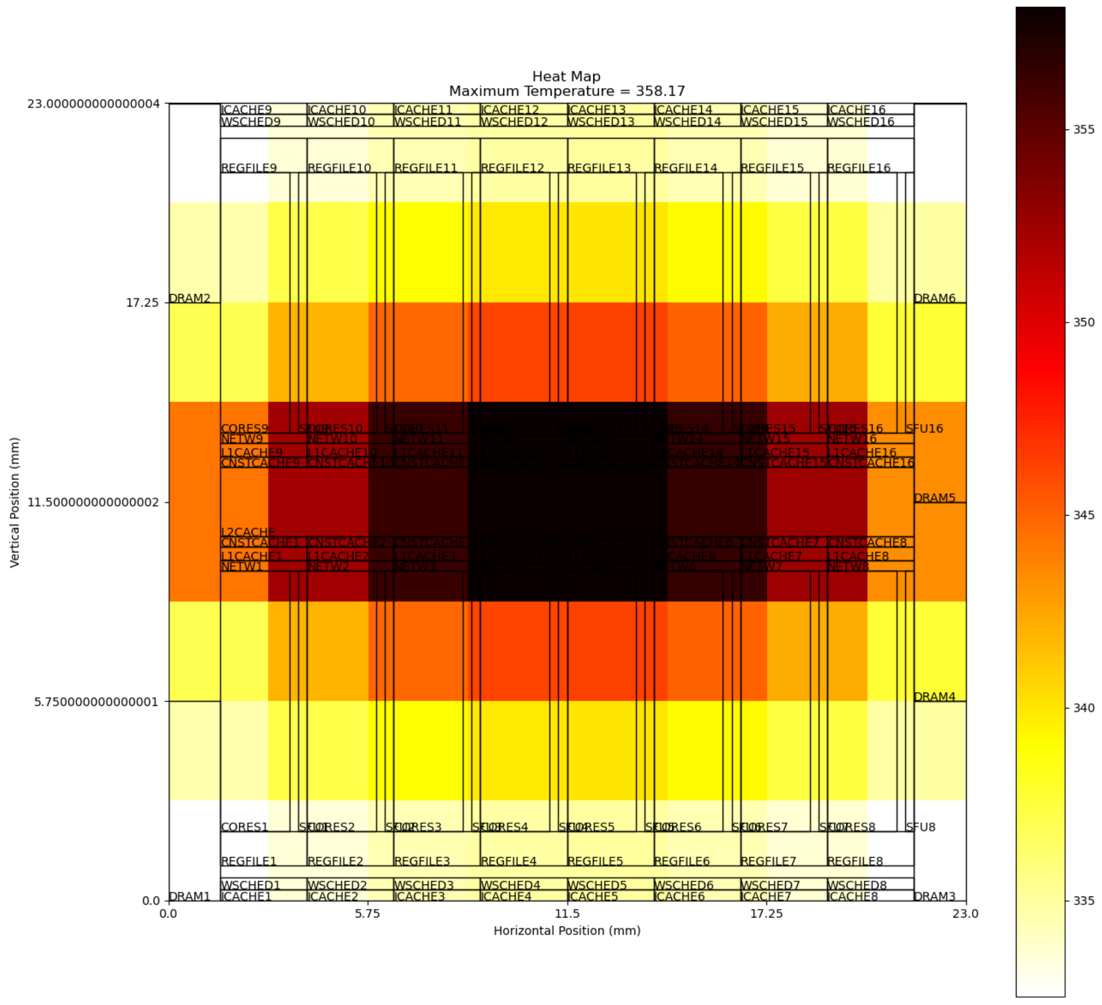


3. Matrix size 100*100-

Layer 0

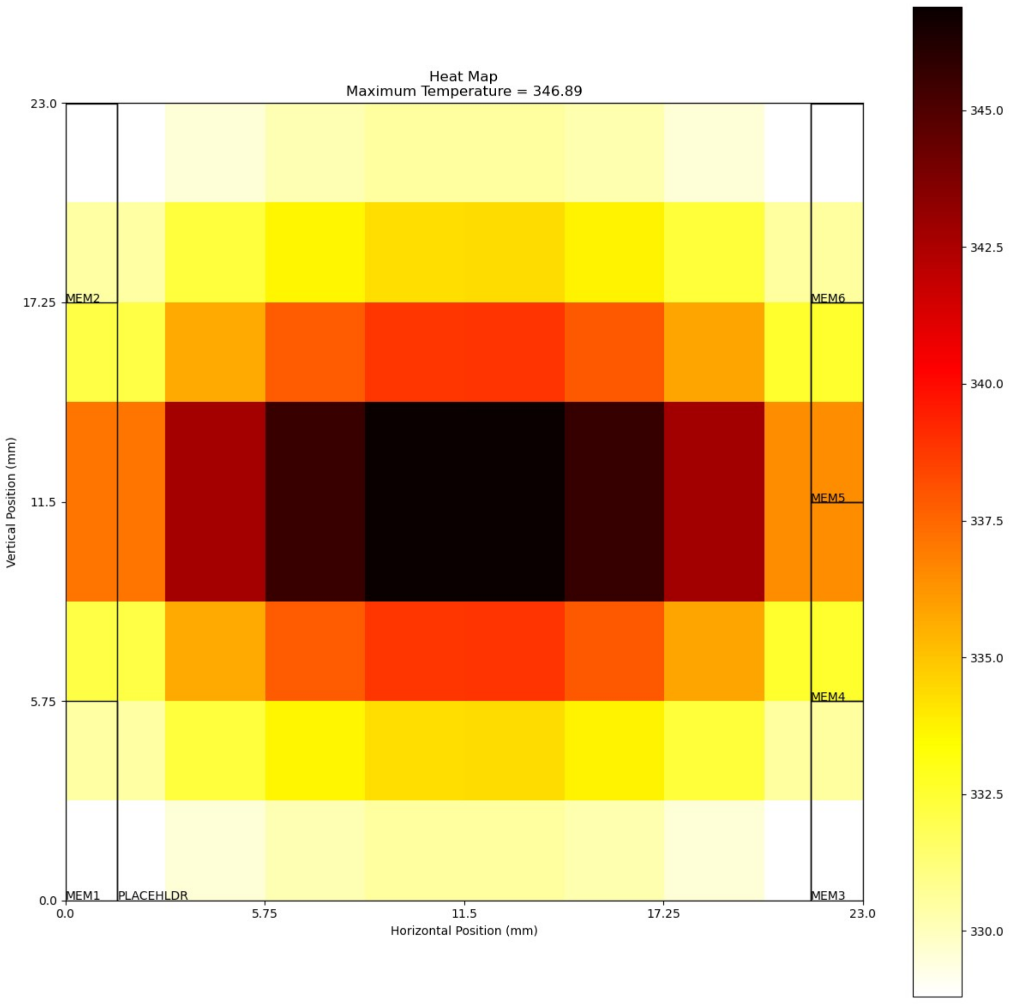

Heat Map
Maximum Temperature = 357.99
Vertical Position (mm)
Horizontal Position (mm)
23.0
17.25
11.5
5.75
0.0
0.0
5.75
11.5
17.25
23.0
MEM1
MEM2
MEM3
MEM4
MEM5
MEM6
PLACEHLDR
355
350
345
340
335

Layer 2

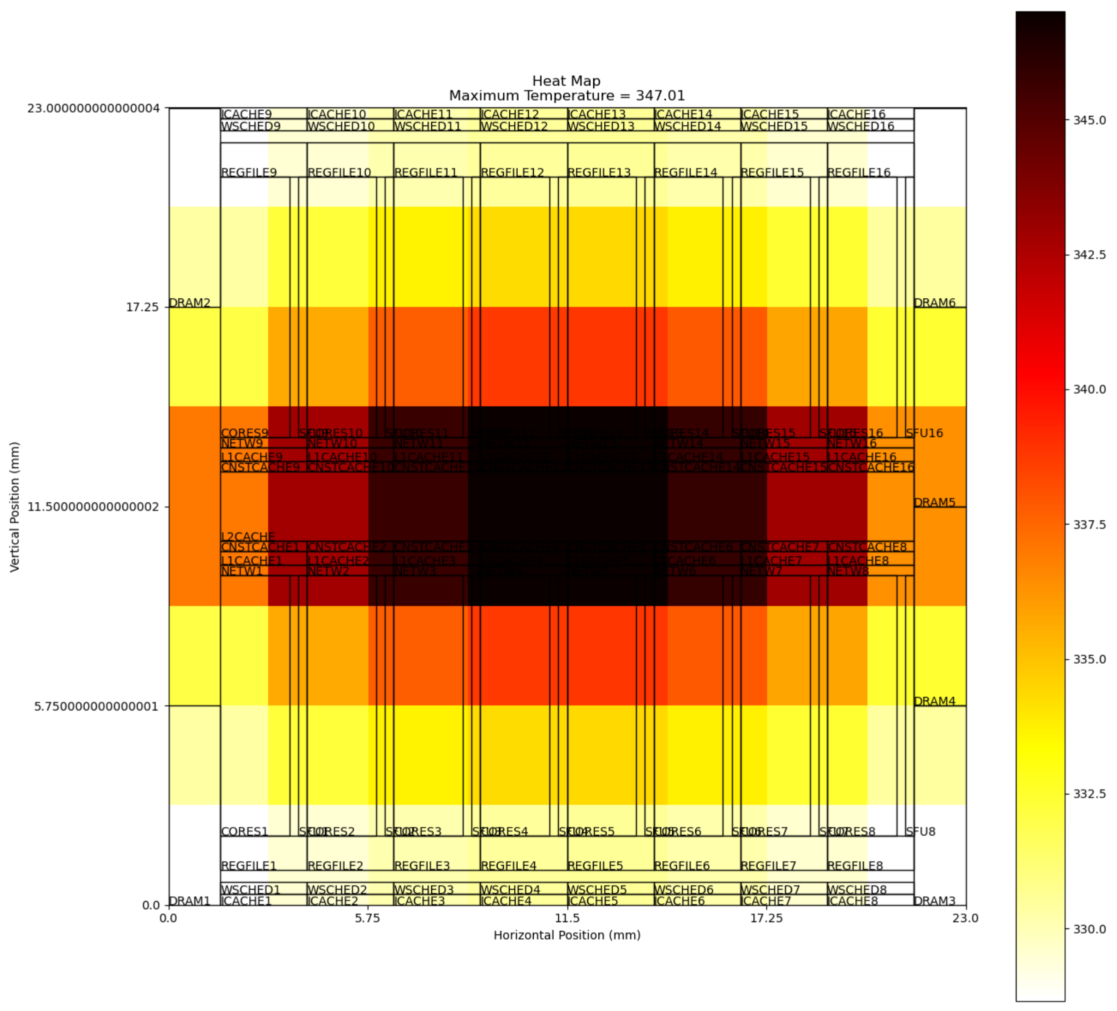


4. Matrix size 200*200-

Layer 0

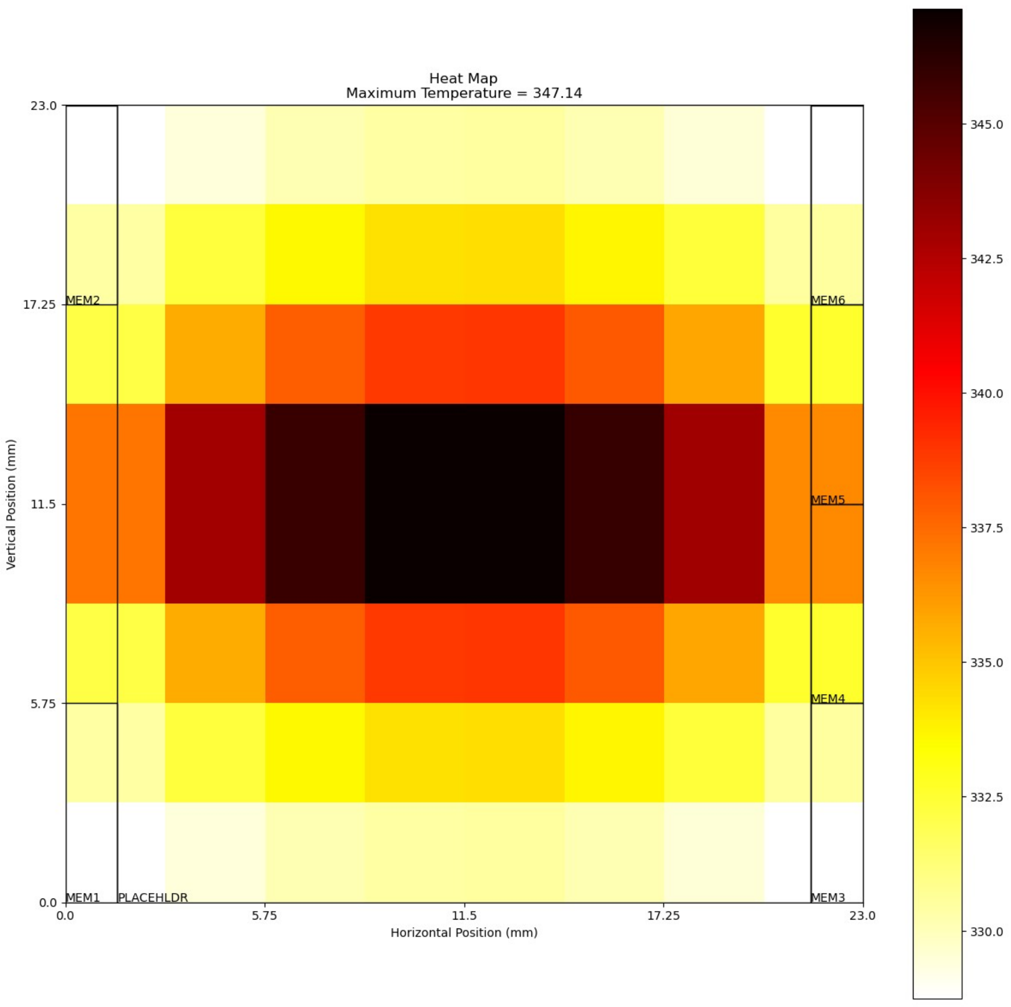

Heat Map
Maximum Temperature = 346.89
Vertical Position (mm)
Horizontal Position (mm)
0.0
5.75
11.5
17.25
23.0
MEM1
MEM2
MEM3
MEM4
MEM5
MEM6
PLACEHLDR
345.0
342.5
340.0
337.5
335.0
332.5
330.0

Layer 2

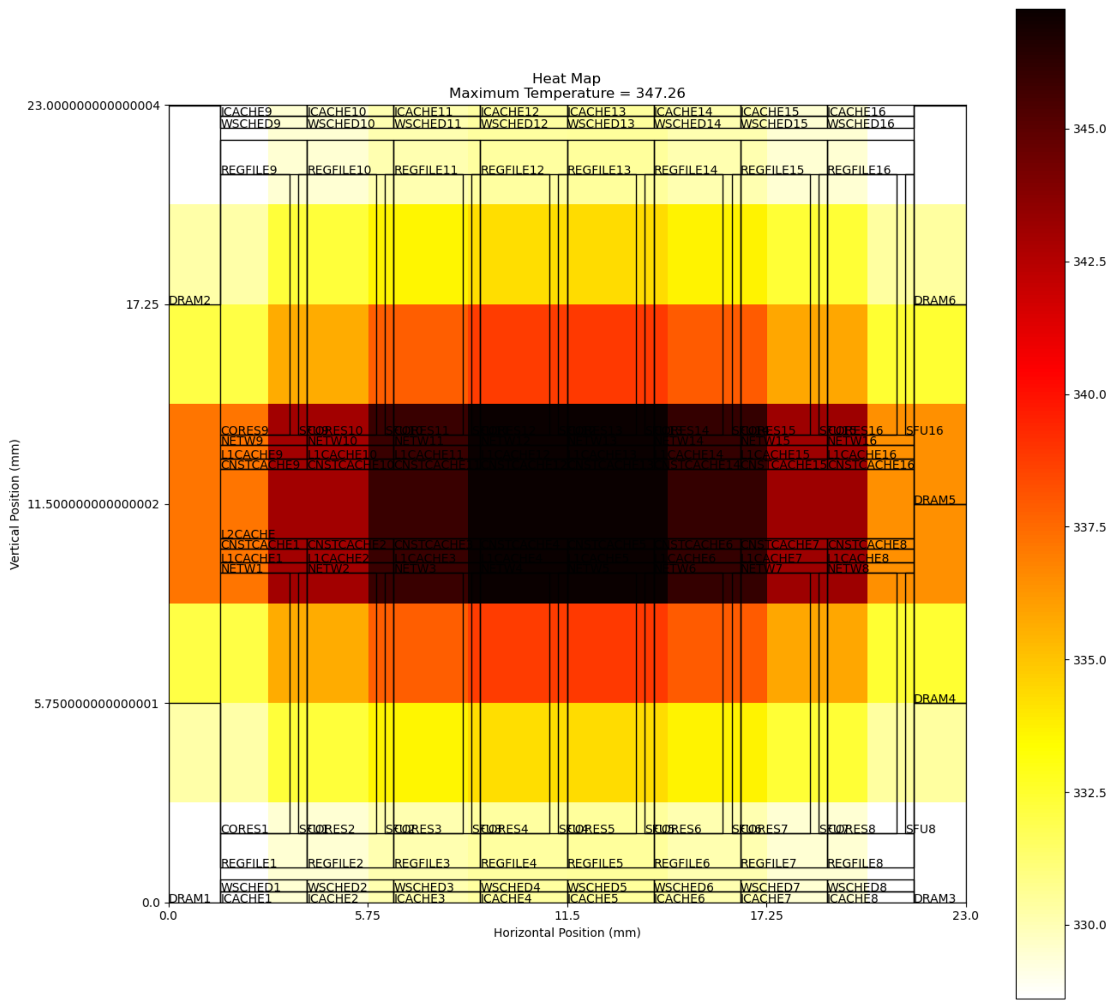


5. Matrix size 300*300-

Layer 0

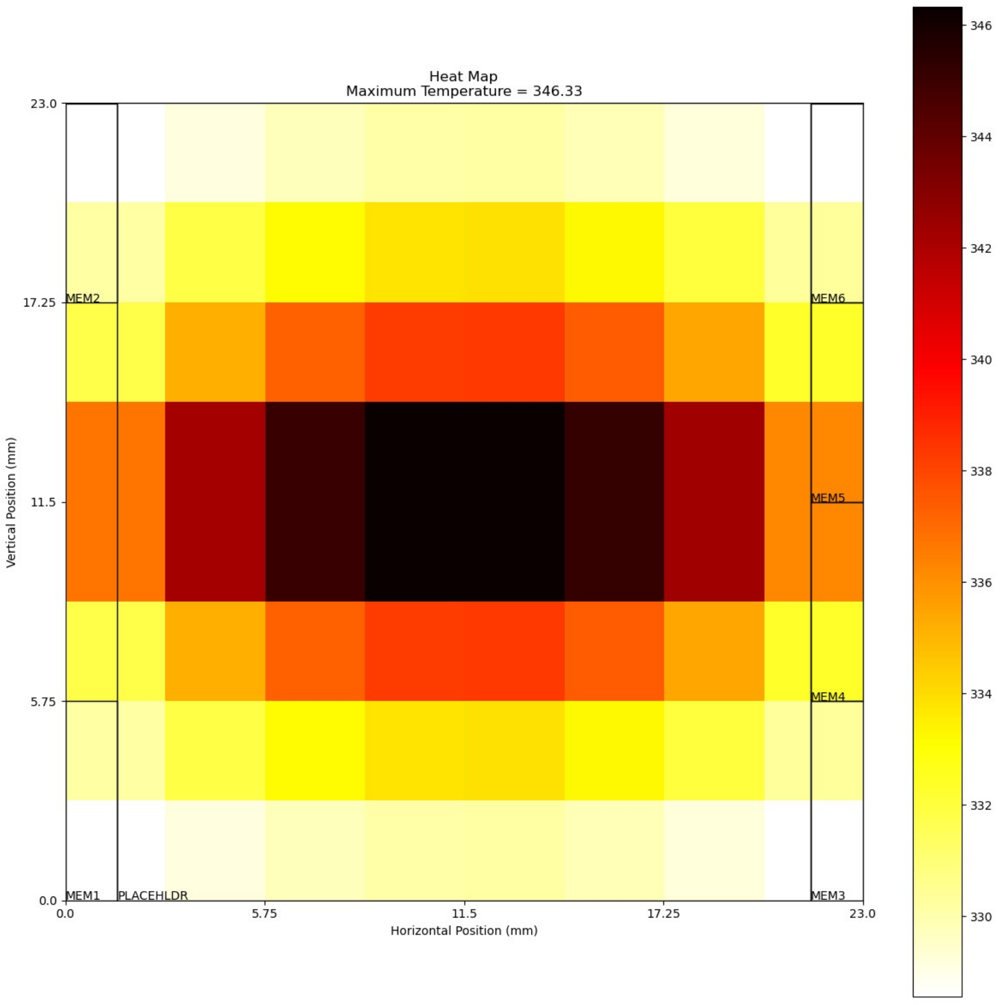

Heat Map
Maximum Temperature = 347.14
Vertical Position (mm)
Horizontal Position (mm)
23.0
17.25
11.5
5.75
0.0
0.0
5.75
11.5
17.25
23.0
MEM1
MEM2
MEM3
MEM4
MEM5
MEM6
PLACEHLDR
345.0
342.5
340.0
337.5
335.0
332.5
330.0

Layer 2

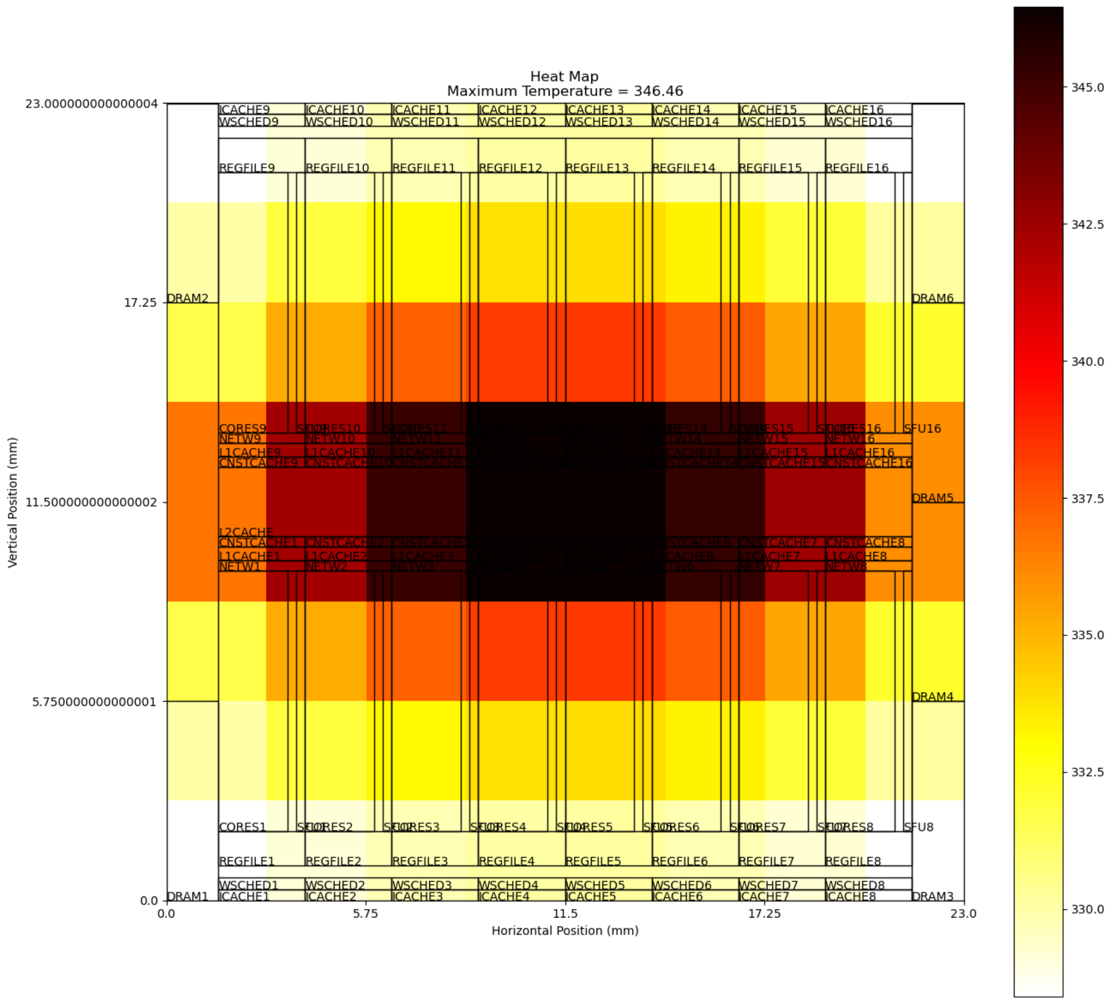


6. Matrix size 400*400-

Layer 0


Heat Map
Maximum Temperature = 346.33
Vertical Position (mm)
Horizontal Position (mm)
23.0
17.25
11.5
5.75
0.0
0.0
5.75
11.5
17.25
23.0
MEM2
MEM6
MEM5
MEM4
MEM1
PLACEHLDR
MEM3
346
344
342
340
338
336
334
332
330

Layer 2

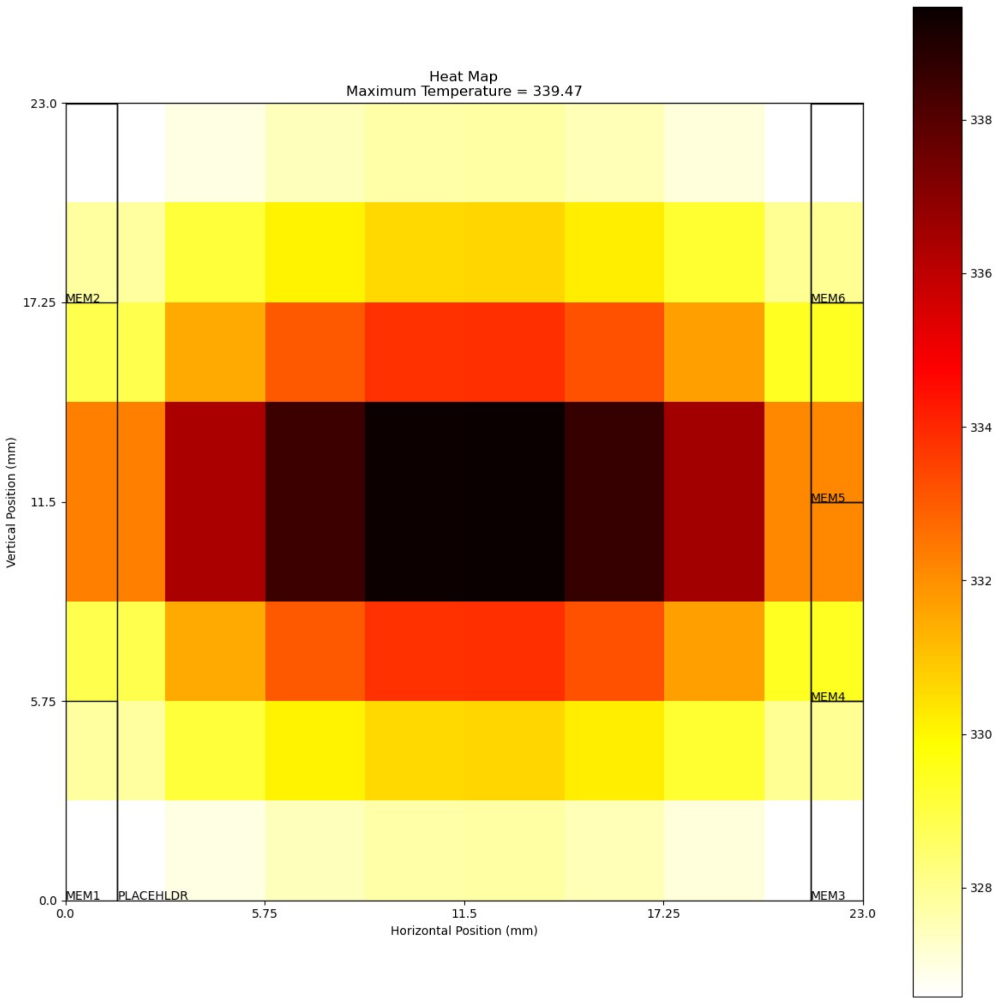


2. Reduced L2 Size

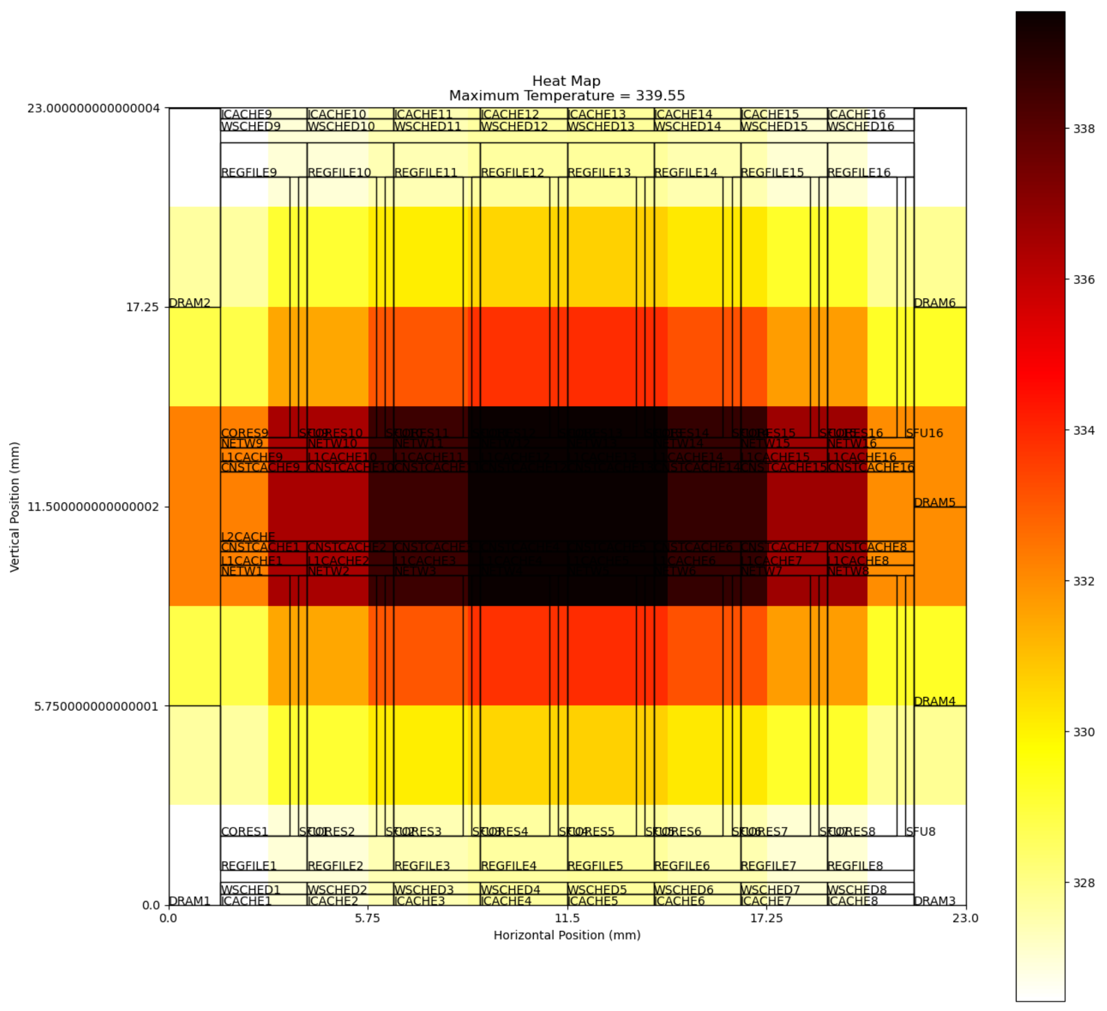
To observe the overflow of L2 accesses into DRAM, L2 size was reduced to half of its original value (i.e. 384 KB) by making configuration changes in both the gpgpusim config and the gpu wattch xml file. The tensor dimensions 250, 325 and 400 were simulated.

1. Matrix size 250*250-

Layer 0

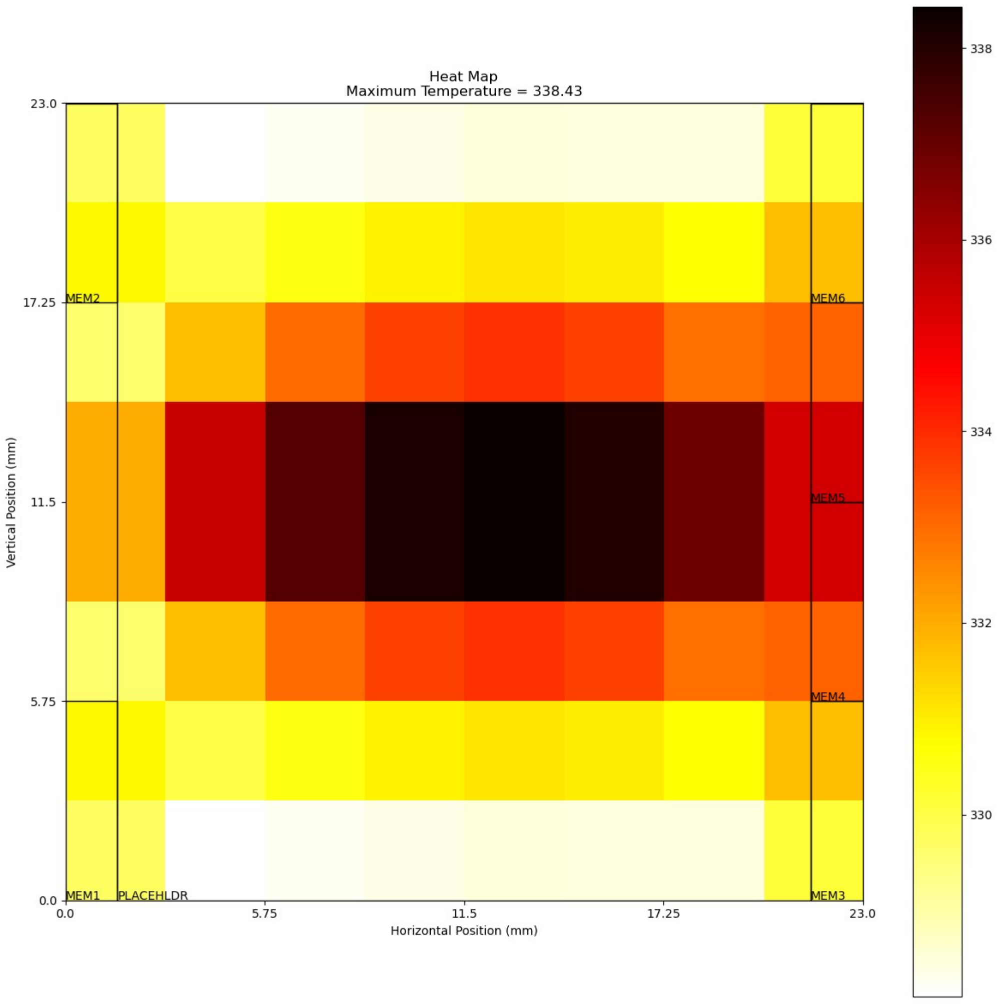

Heat Map
Maximum Temperature = 339.47
Vertical Position (mm)
Horizontal Position (mm)
0.0
5.75
11.5
17.25
23.0
MEM1
MEM2
MEM3
MEM4
MEM5
MEM6
PLACEHLDR
328
330
332
334
336
338

Layer 2

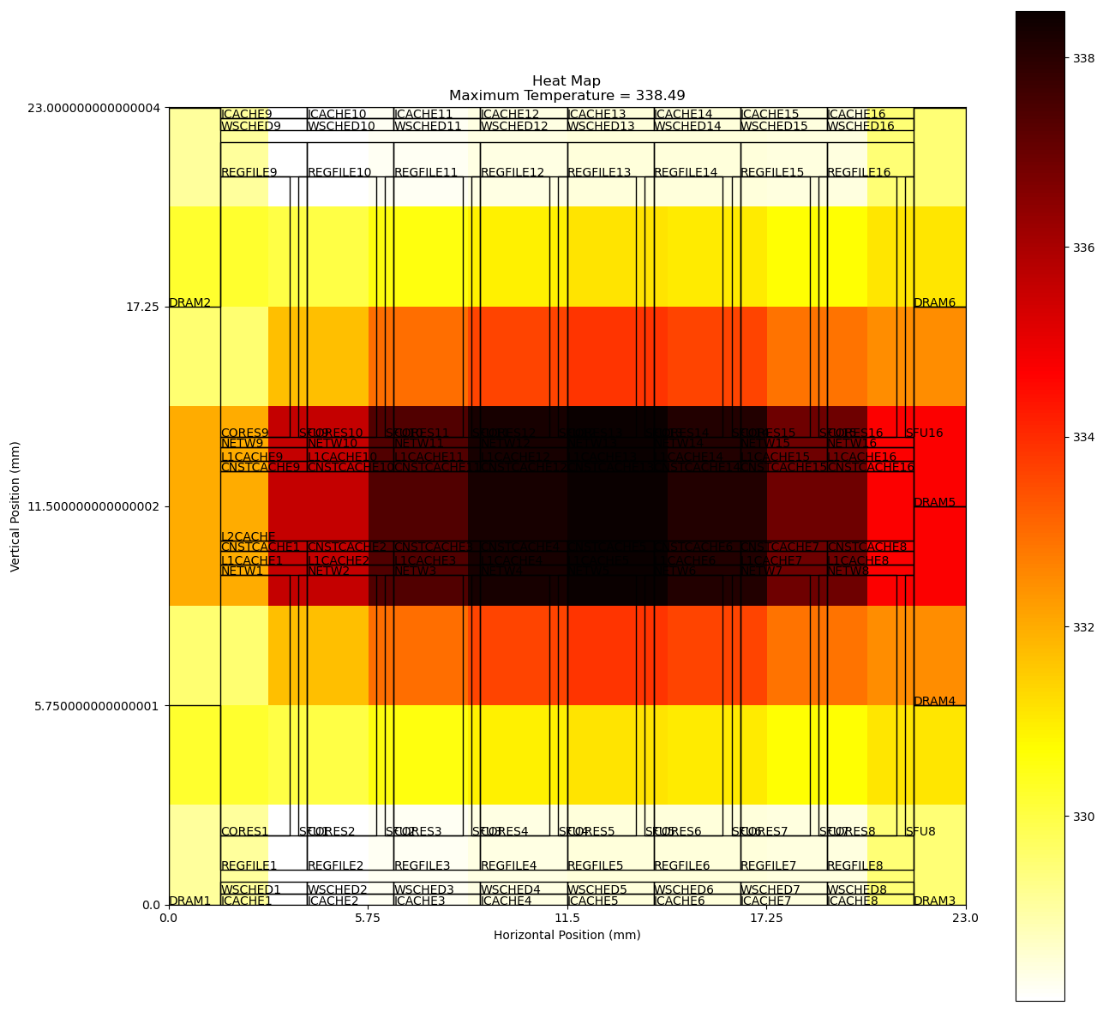


2. Matrix size 325*325-

Layer 0

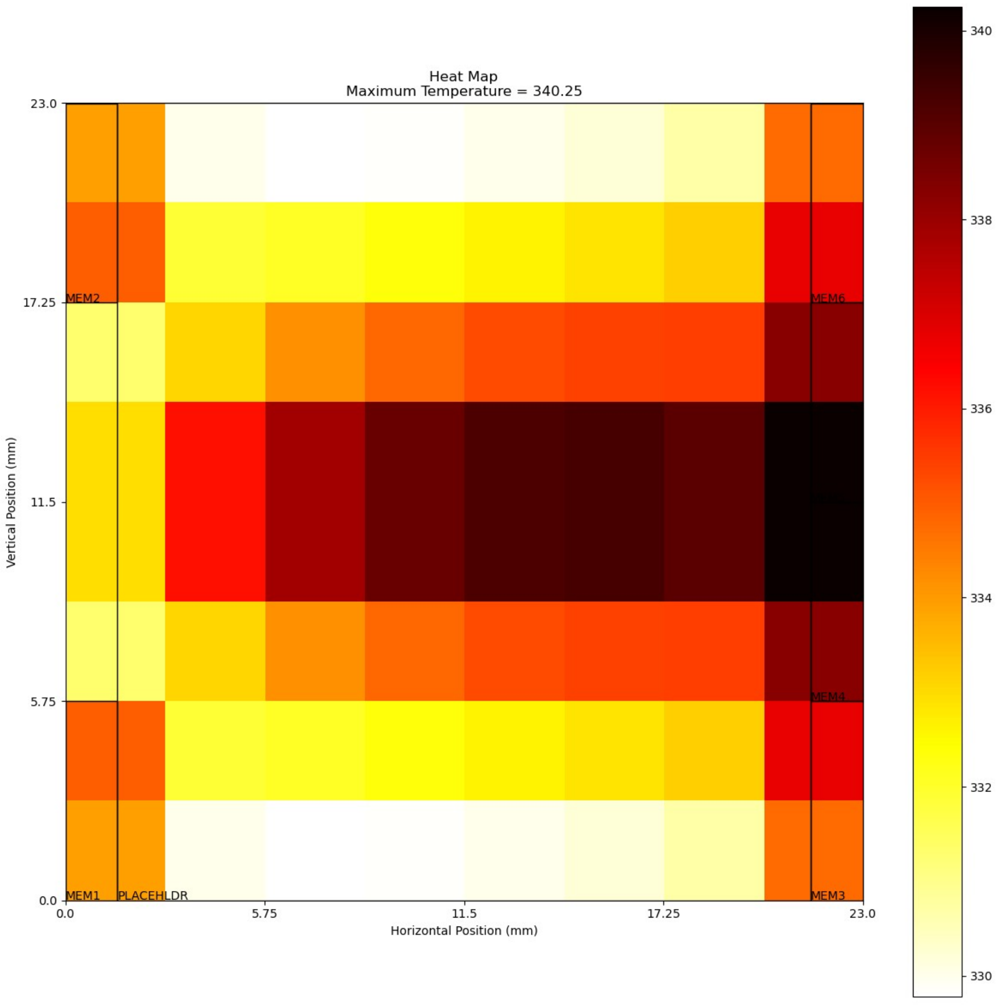

Heat Map
Maximum Temperature = 338.43
Vertical Position (mm)
Horizontal Position (mm)
0.0
5.75
11.5
17.25
23.0
MEM1
MEM2
MEM3
MEM4
MEM5
MEM6
PLACEHLDR
330
332
334
336
338

Layer 2

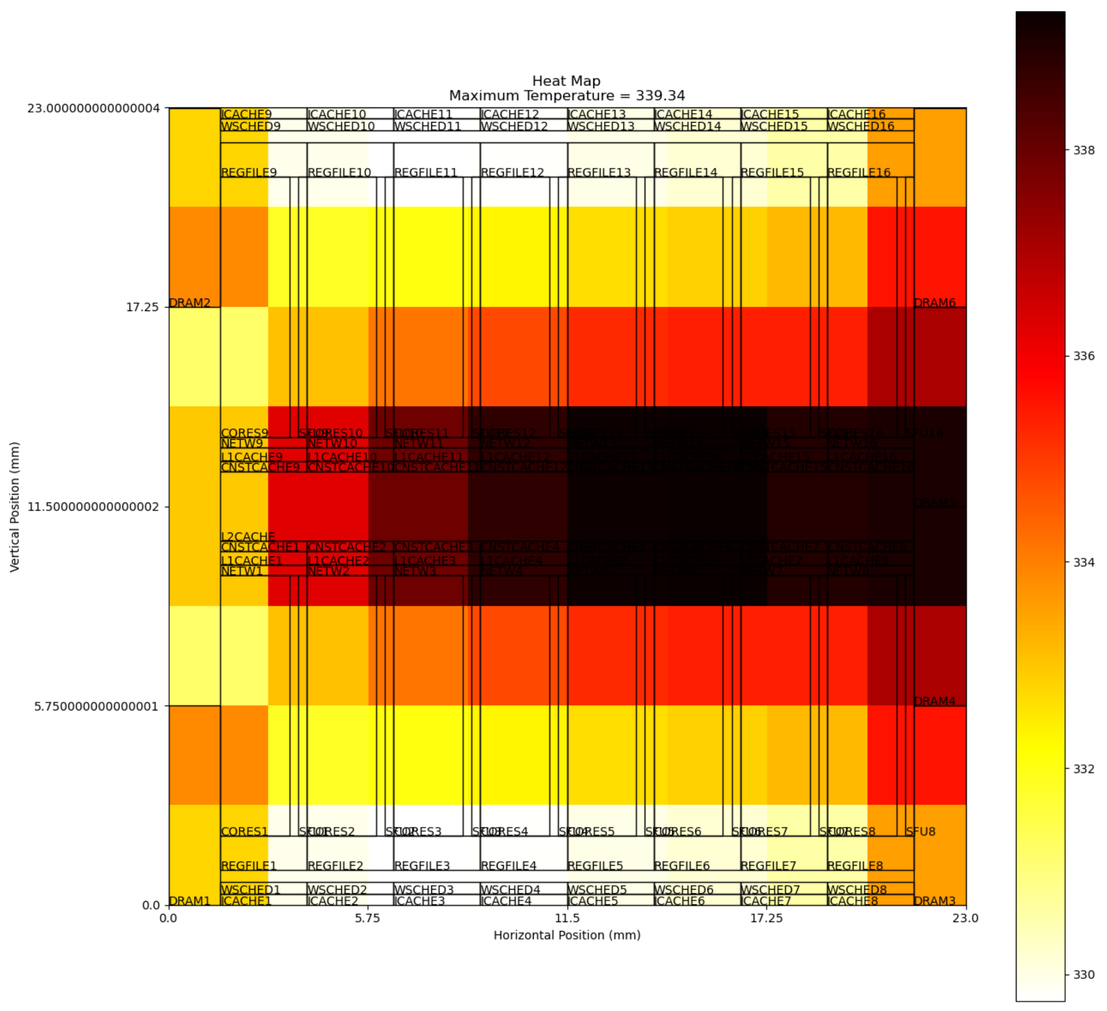


3. Matrix size 400*400-

Layer 0


Heat Map
Maximum Temperature = 340.25
Vertical Position (mm)
Horizontal Position (mm)
0.0
5.75
11.5
17.25
23.0
MEM1
PLACEHLDR
MEM2
MEM3
MEM4
MEM6
330
332
334
336
338
340

Layer 2



3. Summary

Original L2 Size			
Tensor Dimension	**Layer**	**Maximum Temperature (Kelvin)**	**Average Power (Watts)**
10*10	0	330.25	40.1578
	2	330.26	
50*50	0	335.4	51.1219
	2	335.44	
100*100	0	357.99	98.4227
	2	358.17	
200*200	0	346.89	68.7229

	2	347.01	
300*300	0	347.14	67.3437
	2	347.26	
400*400	0	346.33	65.2705
	2	346.46	
Reduced L2 Size (Half)			
Tensor Dimension	**Layer**	**Maximum Temperature (Kelvin)**	**Average Power (Watts)**
250*250	0	339.47	51.4952
	2	339.55	
325*325	0	338.43	62.7543
	2	338.49	
400*400	0	340.25	75.2021
	2	339.34	

Original L2 Size	**Power (Watts)**			
Tensor Dimension	**L1 Cache**	**L2 Cache**	**DRAM**	**Idle Core**
10*10	0.416685	3.8568	0.265433	22.2605
50*50	3.65335	11.514	0.0579432	19.9235
100*100	14.1203	45.4487	0.103998	7.91584
200*200	7.16462	30.9289	0.0275468	3.40807
300*300	6.77014	31.4282	0.0333015	1.1501
400*400	6.27261	30.4133	0.528438	0.224949
Reduced L2 Size (Half)	**Power (Watts)**			
Tensor Dimension	**L1 Cache**	**L2 Cache**	**DRAM**	**Idle Core**
250*250	6.4809	16.0276	0.594152	1.01905
325*325	6.60097	13.0853	13.5765	0.809248
400*400	6.31613	10.3629	28.748	0.273012

6. Explanation-

Original L2 size-

With an increase in the tensor dimension up to 100, the average power increases and as a result, proportionately the maximum temperature also increases. Though, beyond this tensor dimension, the average power reduces and then remains fairly constant for 200, 300 and 400 sizes (this trend is somewhat inexplicable currently). As a result, the maximum temperature also follows the same pattern. A surprising observation shows that for the tensor dimension of 10, the high temperature distribution is more spread out around the central L2 region, covering more area withing the adjacent SMs, whereas for rest of the tensor dimensions, this spread is less. Another observation in the change in maximum temperature is that a significant change in the average power only causes a slight change in the maximum temperature. By looking at the individual power trend for some of the main components, both L1 and L2 cache follow the same pattern as explained above, whereas the idle core power continuously reduces with an increase in the tensor dimension. This is expected since as the tensor dimension increases, the workload increases and it gets distributed across more cores, where each occupied core does some compute and hence consumes power. The DRAM power remains fairly constant with a change in the tensor dimension, but there seems to be no particular trend. As explained previously, all the heat maps show low temperatures for DRAM since it isn't getting accessed at all as the total input size fits into the L2 cache itself. On the other hand, there are considerably high temperatures in the central region occupying a portion of L2 and some SMs (within these SMs, mainly L1 and the cores close to L1).

Reduced L2 size-

In order to observe the overflow of accesses from L1 into DRAM, the L2 cache size was reduced to half. Subsequently, for the tensor dimension of 400, the heat map indicates high temperatures for some of the DRAMs which is in line with the expectation. For tensor dimensions of 250 and 325, L2 accesses still do not need to fall back to the DRAM since the entire input fits into L2 itself. With an increase in the tensor dimension, the average power increases and so does the maximum temperature. Observing the trends in power for primary individual components shows that DRAM power increases significantly, whereas L2 power reduces slightly. Power values for L2 and idle core remain almost constant, with minor decreases in the idle core power since more cores actively participate in the compute.