Thermal analysis and imaging of GPUs using CUDA kernels: Matrix Multiplication & 2D Convolution

Ajay Narayanan Sridhar
The Pennsylvania State University
University Park, USA
afs6372@psu.edu

***Abstract*—With the advent of parallel computing using GPUs in machine learning and bio-informatics, there has been a significant increase in power and energy consumption. Thermal analysis of popular CUDA kernels would help in driving the research toward reducing the power and energy consumption of GPUs. In this work, we analyze the trends of heat dissipation to input size for matrix multiplication kernel and 2D convolution. We also utilize a thermal camera to see if we can identify workloads running on a system with just a thermal image of the CPU box. Our analysis suggests that the CUDA kernels become memory intensive after a fixed input, and the heat dissipation shifts towards memory (DRAM) components.**

***Index Terms*—Thermal analysis, CUDA kernels, Thermal Camera, GPGPU Simulation**

I. Introduction

In this work, we explore two common operations prevalent in deep learning, machine learning and bio-informatics: matrix multiplication, and 2D convolution. Since these two are base operations in CNNs, Transformers, etc., analyzing them will drive us one step closer to understanding the power consumed by a typical neural network.

In this work, we implement the operations in Nvidia CUDA [1] and simulate them to get their heat maps on Nvidia GTX 480 GPU. We hope that this analysis helps us understand CUDA better and exploit the architecture of CUDA to write high-performant kernels for applications in machine learning and bio-informatics. Kindly refer to this GitHub project (https://github.com/ajaynarayanan/cse_530) for python automation code (ptrace convertor, end to end job scheduler)

II. Design

We design a 3D floor plan with four layers (Silicon—TIM—Silicon—TIM) based on Nvidia Fermi Architecture [2]. We also design and implement CUDA kernels for matrix multiplication and 2D convolution.

A. Floor plan

We design our floor plan as close as possible to the fermi architecture design [2]. Based on the whitepaper of Nvidia GTX 480 and fig.1, we observe the following,

- Die size is 529 mm^2. We choose a square floor of size 23 mm for our floor plan.
- GPU has 16 streaming multiprocessors (SM). Each SM has 32 CUDA Cores, 16 load-store units, 64 kb of shared memory/L1 cache, 4 SFUs, and a register file.
- 6 DRAM components located at the edges of the die
- 1 GigaThread Scheduler and 1 Host Interface
- L2 cache of size 768kb located in the center.

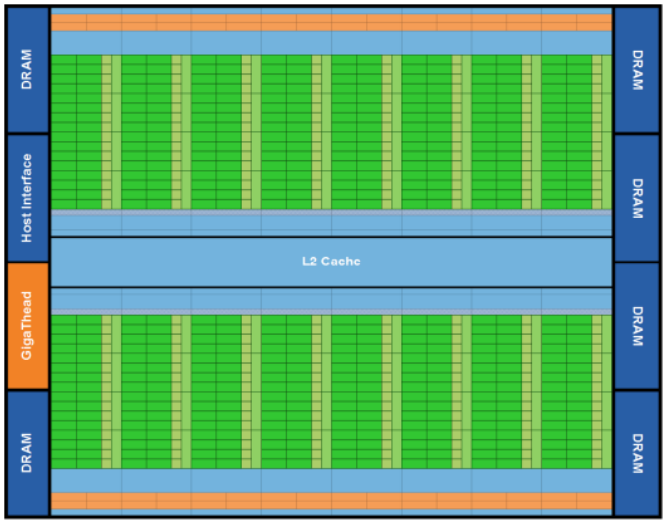


Fig. 1: Layout of fermi architecture with all components as per Nvidia's Fermi whitepaper.

.

Our 3D floor plan consists of 2 silicon layers (layer 0 and layer 2), which are heat-emitting and consume power and 2 TIM layers (layer 1 and layer 3) which are heat-insulating and do not consume any power. We stack DRAMs in all the layers (named DRAM and MEM in the floorplan), but we place the rest of all components in layer 0.

B. Kernels

1) Matrix Multiplication:

$$c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + \cdots + a_{in} + b_{nj} = \sum_{k=1}^{n} a_{ik}b_{kj} \quad (1)$$

Here, A is of mxn, B is of size nxp and C_{mxp} is the result of multiplying matrix A and B

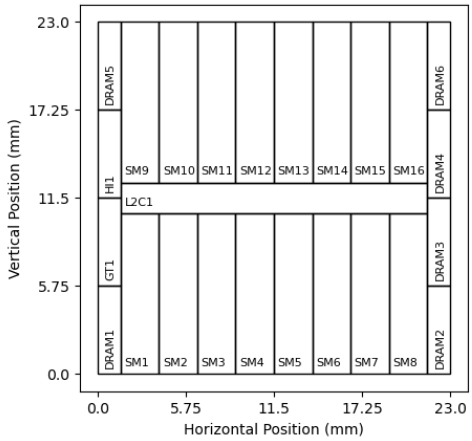


Fig. 2: Layer 0 of our floorplan

.

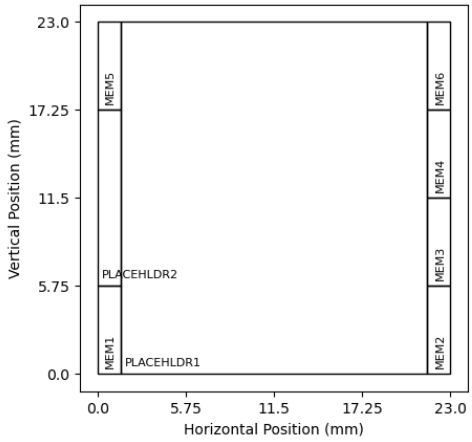


Fig. 3: Layer 2 of our floorplan

.

We consider only square matrices for our analysis. Thus, we optimize the kernel code for the same. We use the concept of tiled matrix multiplication [3]. We use shared memory to compute intermediate multiplication results and then finally write the result to global memory once per tile block. We fix the block size as 16x16 for this kernel.

2) 2D Convolution:

$$y[i,j] = \sum_{m=0}^{(M_a-1)} \sum_{n=0}^{(N_a-1)} A[m,n] \cdot B[i-m, j-n] \qquad (2)$$

Here $0 \leq i < Ma + Mb - 1$ and $0 \leq j < Na + Nb - 1$ and $y = A * B$ where A is of size (Ma, Mb) and B is of size (Na, Nb).

For our convolution kernel [4], we follow the tiling approach similar to matrix multiplication. We fix the second convolutional input to a size of 5x5. Here, we fix the block size of CUDA threads to be 32x32.

III. Methodology

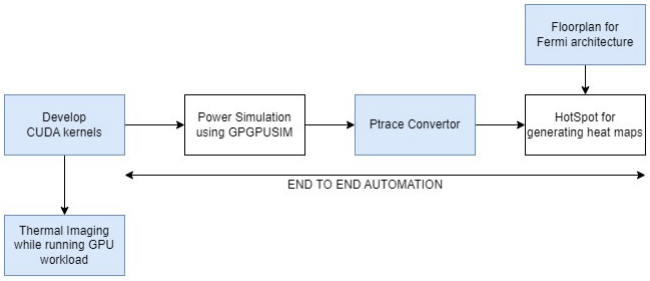


Fig. 4: Pipeline/Flow diagram of our work

.

Fig. 4 shows our work's methodology/pipeline. We used the following tools as part of this project,

- NVIDIA CUDA Compiler (NVCC) [5]; for compiling CUDA kernels
- GPGPU-Sim 4.0 [6]; to generate power trace for the given CUDA kernel
- Hotspot 7.0 [7]; thermal simulation of given 3D floor plan with power trace
- FLIR ONE PRO [8] ; thermal imaging camera

IV. Experiments and Discussions

A. Impact of input size of convolution

From Fig.7, we observe the following as we increase the input size of convolution kernel,

- As we increase the input size, there is an increase in heat dissipated by DRAM components. This is because, after a fixed input size, the kernel becomes memory intensive compared to compute-intensive. Since our CUDA kernel tiles are fixed, an increase in input size doesn't allow all the data to be retained in the shared memory of the L1 cache and also in L2 (size of 512*512 input: 1024 KB). Thus, there is always a fetch to DRAM to get the data, explaining the increased temperature of DRAMs.
- We notice that as we increase the input size, there is an increase in maximum temperature in the heat maps but the temperature decreases after 256x256. We hypothesize that at this point, the kernel becomes memory intensive, resulting in a steady state of utilization of SMs, leading to an almost steady maximum temperature in GPU.
- We notice that the heat dissipated by SMs is clustered around the center of the GPU. We hypothesize that this is due to the fact that we are not using all threads of SMs in GPU, and thus there is a high temperature in the middle SMs.
- The temperature is high around L2 cache due to frequent access to L2 cache for all the input sizes.

B. Impact of input size of matrix multiplication

From Fig.8, we observe the following as we increase the input size of matrix multiplication kernel,

- As we increase the input size, there is an increase in heat dissipated by DRAM components. However, this is only

observed on the DRAMs that are on the right side. We speculate that this is because when heated components are close together, their overall temperature increases, similar to constructive interference of temperature. As the DRAMs on the left are separated by components that we assume don't emit heat, there is no constructive interference on temperature. However, this is more prevalent on the right side, where all four DRAMs are close together.

- We notice that as we increase the input size, there is an increase in maximum temperature in the heat maps. This observation is in line with the general trend that the more we compute, the more heat is dissipated.
- We notice that the heat dissipated by SMs is clustered around the center of the GPU. We hypothesize that this is due to the fact that we are not using all threads of SMs in GPU, and thus there is a high temperature in the middle SMs.
- The temperature is high around the L2 cache due to frequent access to the L2 cache for all the input sizes.

C. Impact of Input size on average Power

From Fig. 5, we observe that as we increase the input size, there is an increase in the average power consumed by the GPU. We also see that the power consumed by matrix multiplication is higher compared to the convolution kernel.

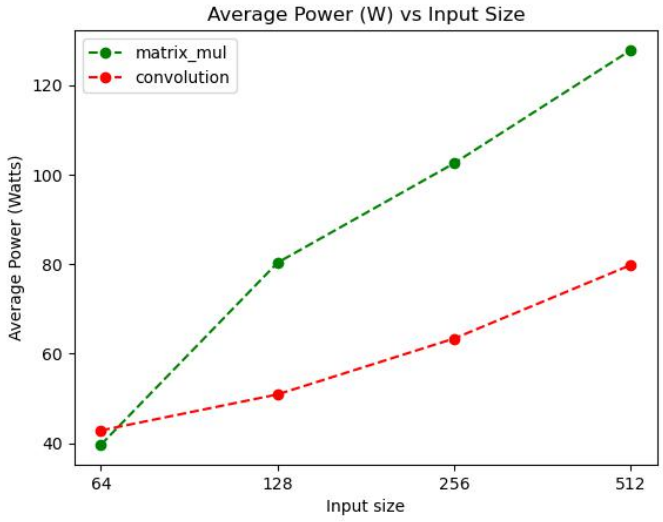


Fig. 5: Average Power (W) vs input size for convolution and matrix multiplication kernel

.

D. Thermal Imaging of CPU & GPU

We take thermal images of the CPU and GPU while a minor workload is running to see if the image has features that can help in identifying the workload. However, from Fig. 6, we observe that the heat signatures doesn't help much as they are smudged all around the GPU and CPU. Additionally, thermal imaging might be inaccurate because GPU is not isolated in real-life scenarios.

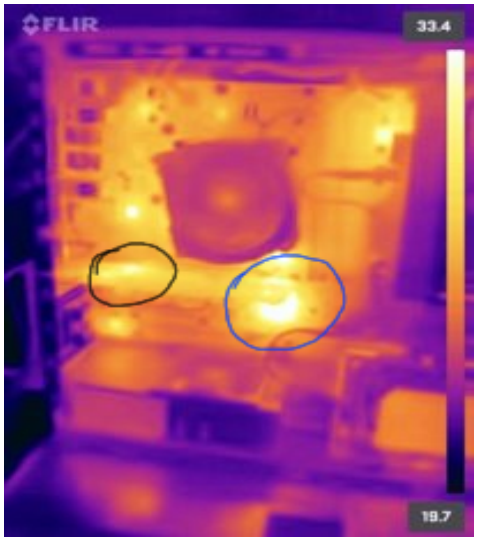


Fig. 6: Thermal image of GPU and CPU while running a minor workload. The blue circle shows the CPU, while the black circle shows the GPU

.

V. Learning Outcomes

- I learnt how to writing CUDA kernels, generating floorplans from whitepapers, automating workflows
- I also learnt how to use thermal camera, accounting for environmental factors while taking thermal images
- This project also helpd me contribute to open-source tools (improvements in HotSpot)

References

[1] NVIDIA, P. Vingelmann, and F. H. Fitzek, "Cuda, release: 10.2.89," 2020. [Online]. Available: https://developer.nvidia.com/cuda-toolkit

[2] "Nvidia fermi architecture whitepaper." [Online]. Available: https://www.nvidia.com/content/PDF/fermi_white_papers/NVIDIA_Fermi_Compute_Architecture_Whitepaper.pdf

[3] "Matrix multiplication in cuda tutorial." [Online]. Available: https://www.es.ele.tue.nl/~mwijtvliet/5KK73/?page=mmcuda

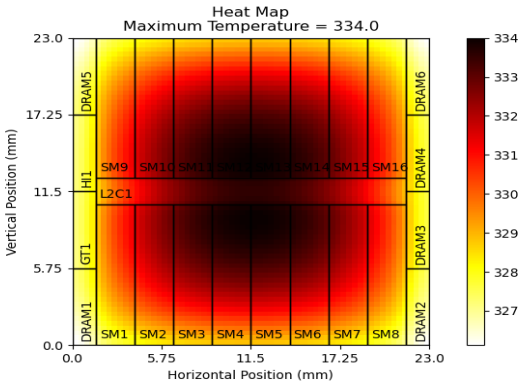
[4] "2d convolution in cuda tutorial." [Online]. Available: https://github.com/krunal1313/2d-Convolution-CUDA

[5] "Nvidia cuda compiler driver nvcc." [Online]. Available: https://docs.nvidia.com/cuda/cuda-compiler-driver-nvcc/index.html

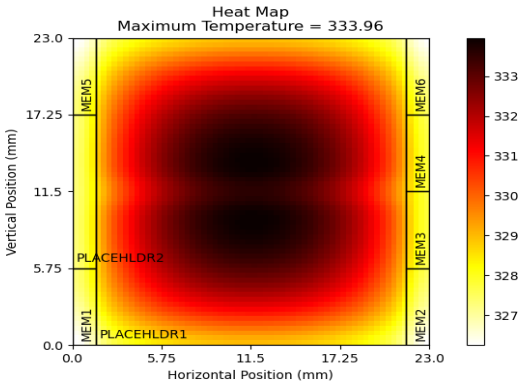
[6] J. Leng, T. Hetherington, A. ElTantawy, S. Gilani, N. S. Kim, T. M. Aamodt, and V. J. Reddi, "Gpuwattch: Enabling energy optimizations in gpgpus," *ACM SIGARCH Computer Architecture News*, vol. 41, no. 3, pp. 487–498, 2013.

[7] "Hotspot 7.0." [Online]. Available: https://github.com/uvahotspot/hotspot

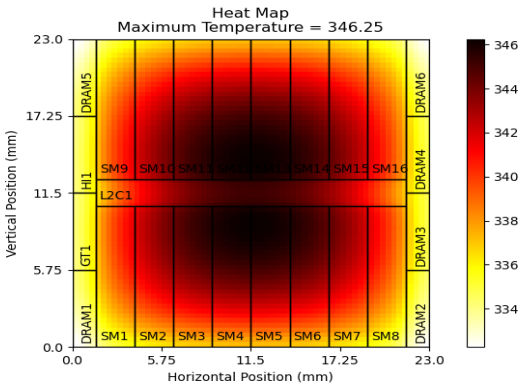
[8] "Flir one pro thermal camer," 2020. [Online]. Available: https://www.flir.com/products/flir-one-pro/?model=435-0007-03&vertical=condition+monitoring&segment=solutions



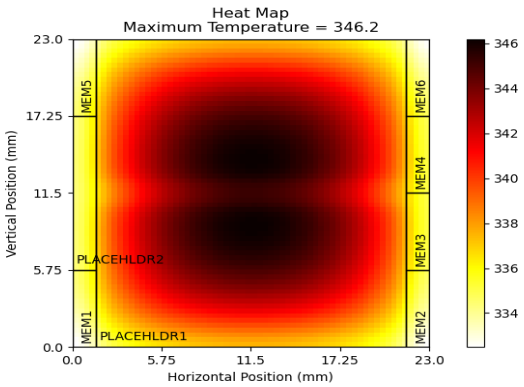
(a) Heat map of layer 0 at input size of 64x64



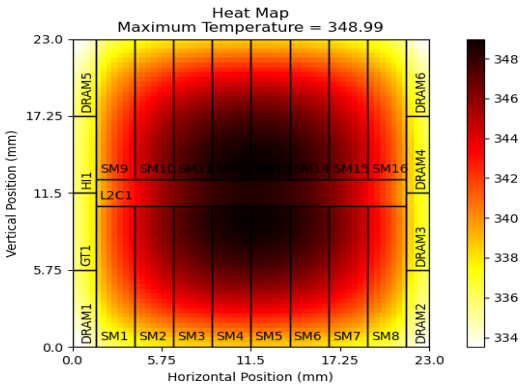
(b) Heat map of layer 2 at input size of 64x64



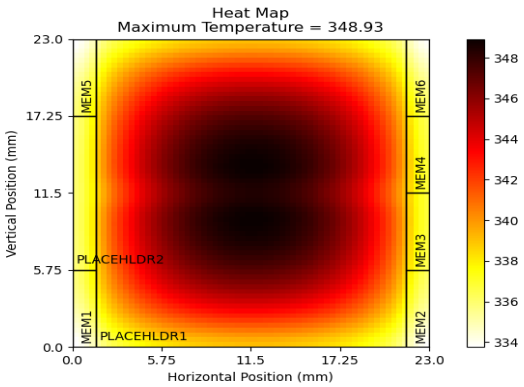
(c) Heat map of layer 0 at input size of 128x128



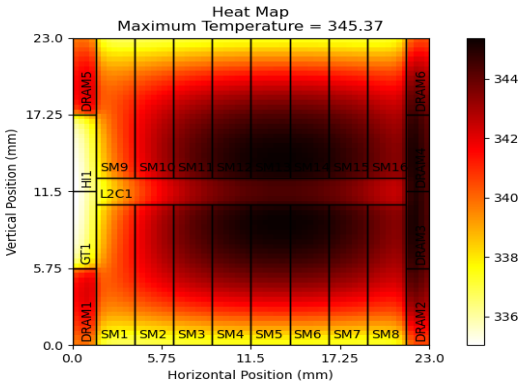
(d) Heat map of layer 2 at input size of 128x128



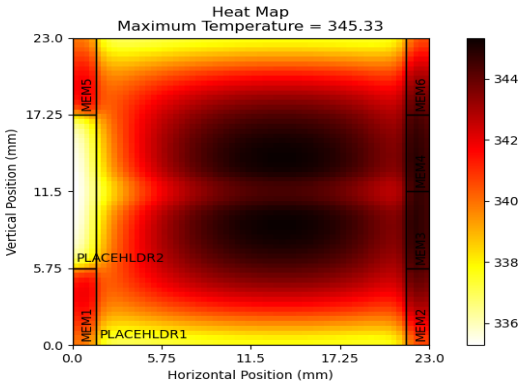
(e) Heat map of layer 0 at input size of 256x256



(f) Heat map of layer 2 at input size of 256x256

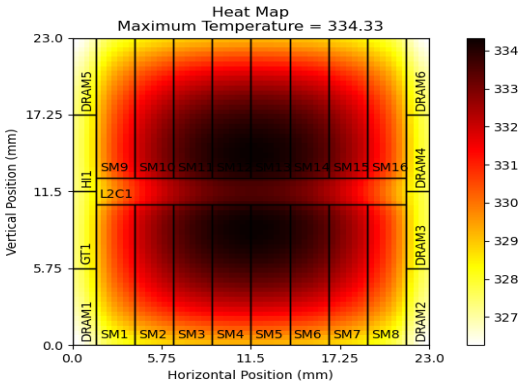


(g) Heat map of layer 0 at input size of 512x512

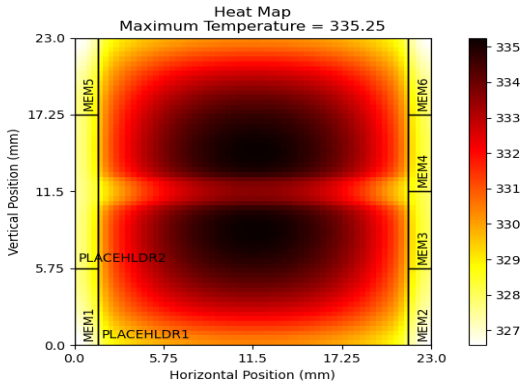


(h) Heat map of layer 2 at input size of 512x512

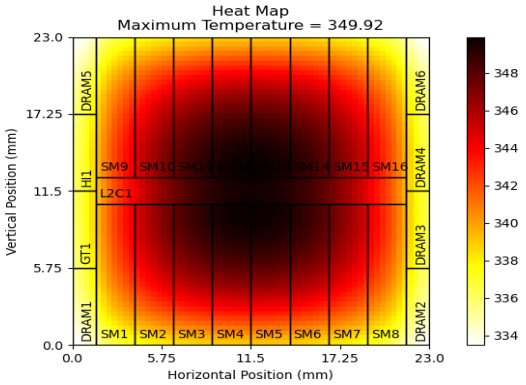
Fig. 7: Heatmaps for convolution kernel at different input sizes (64, 128, 256, 512)



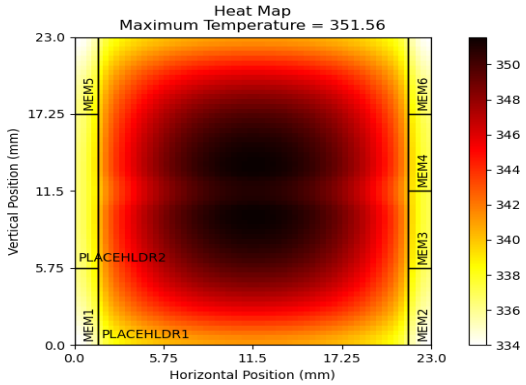
(a) Heat map of layer 0 at input size of 64x64



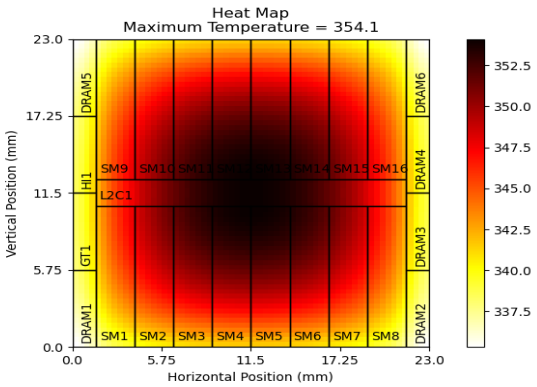
(b) Heat map of layer 2 at input size of 64x64



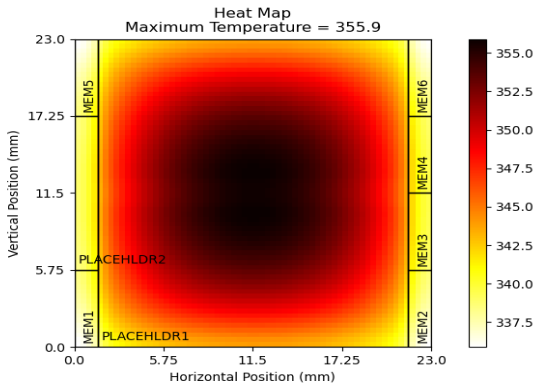
(c) Heat map of layer 0 at input size of 128x128



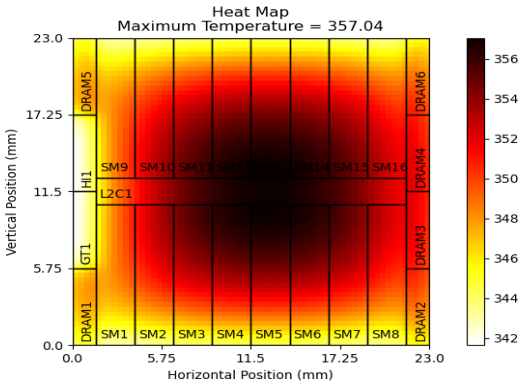
(d) Heat map of layer 2 at input size of 128x128



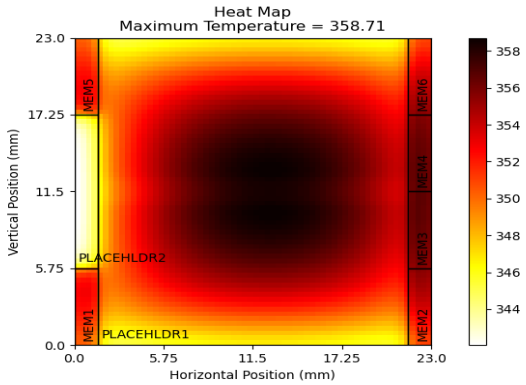
(e) Heat map of layer 0 at input size of 256x256



(f) Heat map of layer 2 at input size of 256x256



(g) Heat map of layer 0 at input size of 512x512



(h) Heat map of layer 2 at input size of 512x512

Fig. 8: Heatmaps for matrix multiplication kernel at different input sizes (64, 128, 256, 512)