WiDS Week 1

Task 1: Identify and Analyse a GPU-Accelerable Workload

Workload: 2D Thermal Simulation of an IC

1.1 Operation Breakdown

1.1.1 Description

There is both static and dynamic power dissipation in an IC. Transistors on an IC behave differently in different temperatures. In order for us to efficiently map transistors onto the IC, we need to first simulate the heat map on the IC. This helps us understand where we can and cannot place components on the IC.
We can model the heat diffusion across the chip by representing the floorplan as a 2D grid. The temperature of a grid cell depends on only two things: whether it has a power source or not and the temperature of the surrounding cells. Thus, we can compute the temperature of each cell independently.

1.1.2 Formula & Pseudocode

$$\frac{\partial T}{\partial t} = \alpha \left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \right)$$

$$T_{i,j}^{(t+1)} = T_{i,j}^{(t)} + \alpha \Delta t \left(\frac{T_{i+1,j}^{(t)} - 2T_{i,j}^{(t)} + T_{i-1,j}^{(t)}}{\Delta x^2} + \frac{T_{i,j+1}^{(t)} - 2T_{i,j}^{(t)} + T_{i,j-1}^{(t)}}{\Delta y^2} \right) + P_{i,j}$$

Where,

- alpha is the thermal diffusivity
- T is the temperature of a given cell
- P is the power dissipated at that grid cell

Below is the pseudocode for the above formula (CPU implementation)

```
for t in range(T_max):
    for i in range(1, H-1):
        for j in range(1, W-1):
            laplacian = T[i+1][j] + T[i-1][j] + T[i][j+1]
                + T[i][j-1] - 4 * T[i][j]
            T_new[i][j] = T[i][j] + alpha * dt * laplacian + P[i][j]
    swap(T, T_new)
```

1.1.3 Specifications

Input size would be the size of the 2D grid. We can have a grid size of 1024 x 1024 for ~ 1M cells. The loop would probably run for 1k iterations but that would depend on when we get the actual answer. The actual number can only be calculated after testing.

1.2 Compute vs Memory Analysis

1.2.1 Operation Bounds

The calculation of the temperature in each cell is not computation heavy. The bound comes from the thread having to load the temperature values of the adjacent cells. Thus this is memory bound.

1.2.2 Arithmetic Intensity

Arithmetic intensity is the ratio of arithmetic operations to memory operations in a kernel. We are not computing much per grid cell. Thus, the arithmetic intensity is quite low.

1.2.3 Reuse Opportunities

There is a high opportunity for using shared memory. The temperature stored in a single grid cell is used by four other threads. Having a shared memory can be extremely beneficial.

1.2.4 Limitations

Threads must be synchronised before a time step starts. This is not much of an issue since all threads are doing the same amount of computation. Only the threads handling cells at the boundary require condition handling.

1.3 Expected Behaviour on a GPU

1.3.1 Mapping the Threads

1 CUDA thread is mapping onto a single grid cell. A thread block can map to a small tile on the chip.

1.3.2 Scaling

For a 1024 x 1024 grid, there are ~ 1M cells. This requires 1 million threads per time step which can be easily handled by modern GPUs. It is only limited by memory.

1.3.3 Challenges

There is a low chance of warp divergence since every thread runs the same instructions. Only the boundary checks can create divergence. The only problem would arise in incorrectly using the shared memory since it is limited by the GPU's capacity.

1.4 CPU Baseline

The CPU Baseline for 1k timesteps is **`1323.24s`**

Task2: CUDA Execution Model Mapping Diagram