Varsha Singh
EC527: Lab 7

Part 1: Tutorial

```
(base) [varsingh@scc-k02 varsingh]$ nvcc -arch compute_60 -code sm_60 cuda_test_lab7.cu -o cuda_test
(base) [varsingh@scc-k02 varsingh]$ ls
cuda_test  cuda_test_lab7.cu  cuPrintf.cu  cuPrintf.cuh  gpu_info.cu
(base) [varsingh@scc-k02 varsingh]$ ./cuda_test
Length of the array = 50000

Initializing the arrays ...     ... done


GPU time: 0.333856 (msec)

TEST PASSED: All results matched
(base) [varsingh@scc-k02 varsingh]$
```

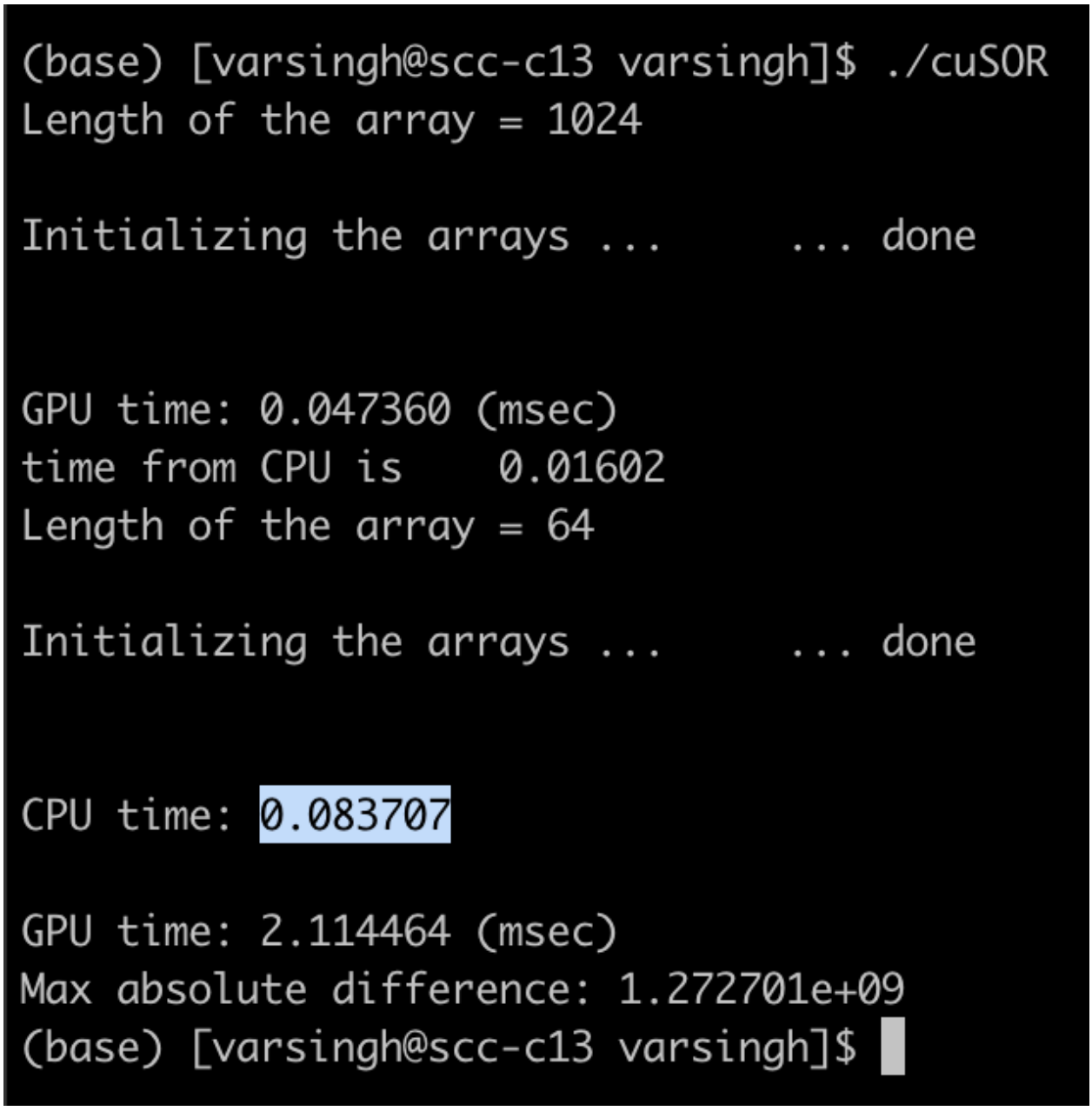
Part 2: Single Block

2a.

```
(base) [varsingh@scc-k01 varsingh]$ ./cuSOR
Length of the array = 1024

Initializing the arrays ...     ... done


GPU time: 0.041344 (msec)
time from CPU is    0.01371
```

I used the interval function that was used in previous labs to get the time in seconds. This is a lot faster than I would expect it to be, at around 331x faster on the GPU as compared to the CPU.

2b.



For the 64x64 array, the speedup is about 39.59x. This I still find extremely fast, but much more reasonable as compared to the first results. I wonder that because I did not include a stall for time delay on the SCC, if this code will experience lag time in the CPU time.

I believe the correctness is not matched. I say this because the max absolute difference is very large. This might be because of the way I have implemented SOR and how the GPU threads will execute the code in a different manner as compared to on the host (CPU).

Part 3: Multiple Blocks

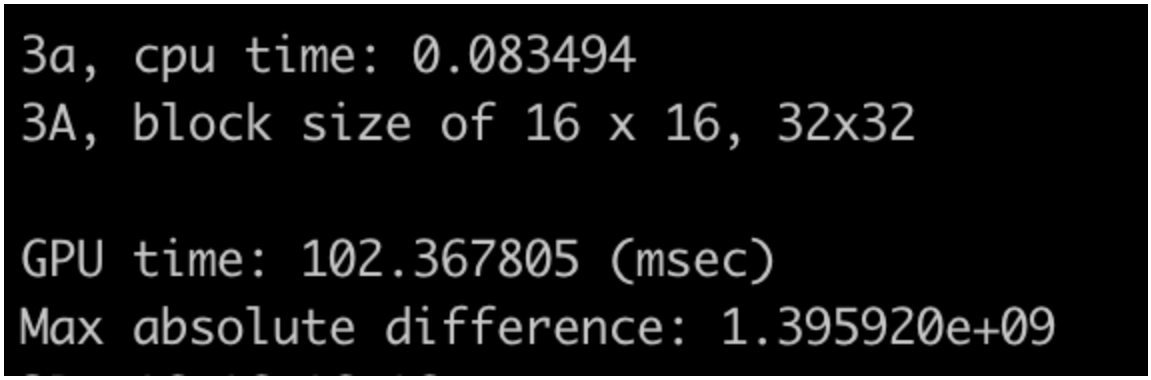
3a.

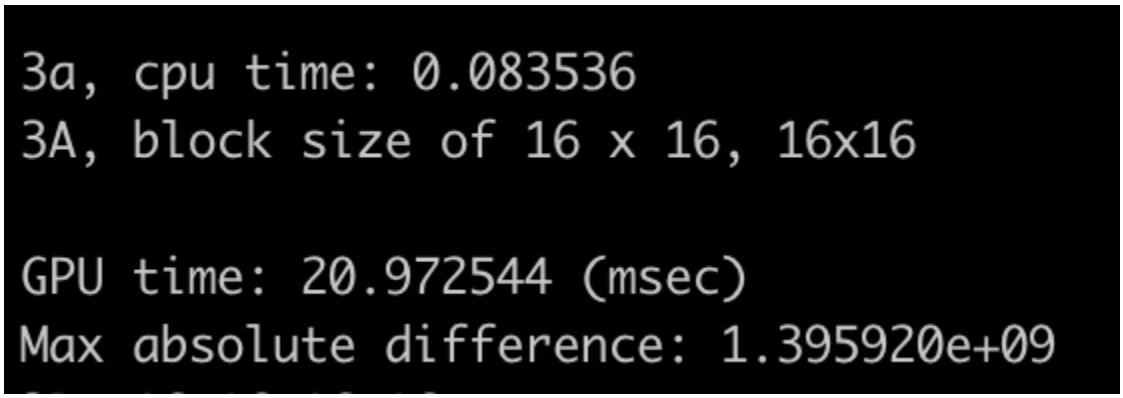
```
3a, cpu time: 0.084524
3A, block size of 32 x 32, 4x4

GPU time: 16.746817 (msec)
Max absolute difference: 1.272701e+09
```

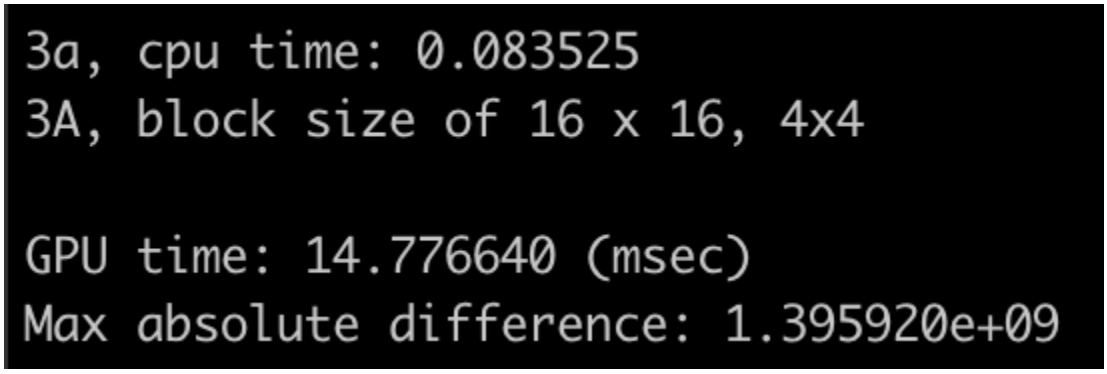
```
3a, cpu time: 0.083541
3A, block size of 32 x 32, 16x16

GPU time: 33.901249 (msec)
Max absolute difference: 1.272701e+09
```

```
3a, cpu time: 0.083675
3A, block size of 32 x 32, 32x32

GPU time: 93.489471 (msec)
Max absolute difference: 1.272701e+09
```

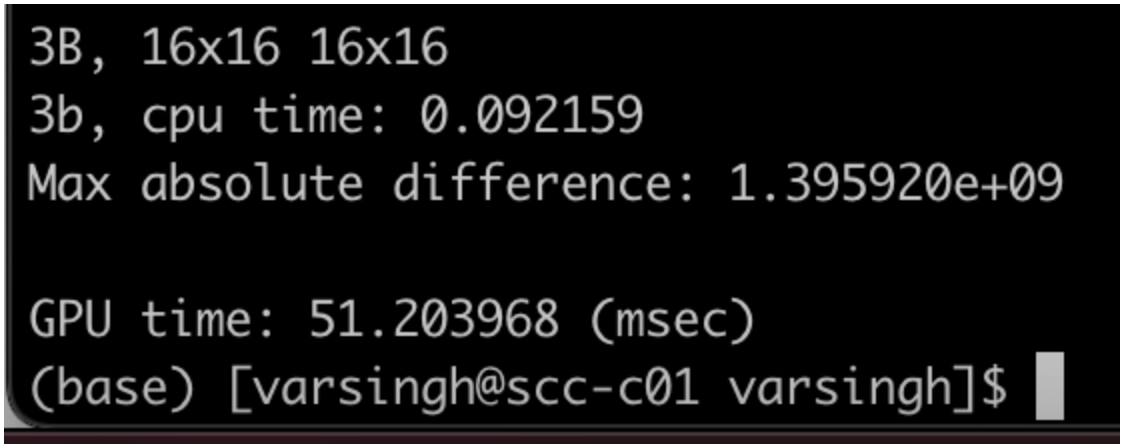
\

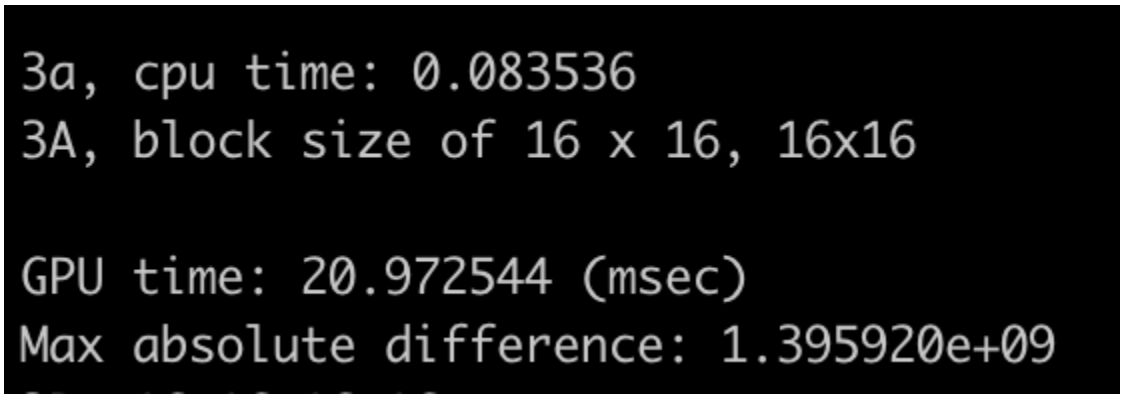






When increasing kernel size from 4x4, 16x16, to 32x32, we see that the difference is the same for the same block size. The difference for a block size of 16x16 is greater than 32x32.
For the block size of 32x32, we see that the difference decreases from 4x4 to 16x16 and then increases largely from 16x16 to 32x32.
As for why, in parallel computation using multiple blocks, each block executes independently. But, there might be dependencies between blocks for SOR, considering that it is a time step sensitive operation to match results as expected with the physics as we see it (referencing Prof. Herbordt's analogy to bubbles). Control from the host ensures that all blocks complete their current iteration before any block starts the next iteration, which can be done with cudaDeviceSynchronize().
3b.





For this, I am showing the results with 3b and then the results from 3a. We see that the absolute difference between the two are the same, but the time it takes for the GPU is about twice as long if it can control the iterations. This could mean that there would be internal synchronization techniques employed by the GPU to ensure that it does not cause issues. This introduces extra overhead and might utilize the GPU's parallelism as we would imagine it can be utilized. There may also be memory access issues when the GPU handles iterations.