**EECS-368 LAB 4 - REPORT**

MATTHEW LENK and NITHIN KRISHNAN

A:

CPU Time: 46.451000 ms

GPU Time: 7.779000 ms

Speedup: 5.971333X

B:

For arrays that were not a power of two, we filled up all the blocks completely that we could and then the rest were written to the last block. By checking the global index of each thread we could determine whether the thread corresponded to data we needed or if it was a thread that was a part of the dead space of the final block

We minimized shared memory bank conflicts by using the provided the CONFLICT\_FREE\_OFFSET that was provided in the Mark Harris Parallel Prefix Sum Scan with Cuda Technical Report. We based our gpu\_prescan on that implementation as well, and modified it so that it would work for very large arrays that needed more than a single block.

Also for our shared memory, we wanted it to be larger than the maximum number of elements per block so we added 30 to it just to give ourselves some wiggle room.

C:

Maybe using some of the CUDA atomic functions would provide faster implementation?

It would be cool if instead of having all of these kernel invocations, we were able to simply have more kernel calls from within the kernels which would complete the same task. I think the transfer from host to device and back again so many times is a bottleneck that could be improved. Unrolling the loops could provide more performance. Number of threads and blocks always a bottleneck for GPUs . For CPU memory accesses is a bottleneck.

For CPU Time complexity is O(n). FLOPS = 16777216/(46.45/1000) = 3.611 x E8

For GPU Time complexity is O(2n) . GFLOPS = 16777216 \* 2/(7.78/1000) = 4.312 x E9