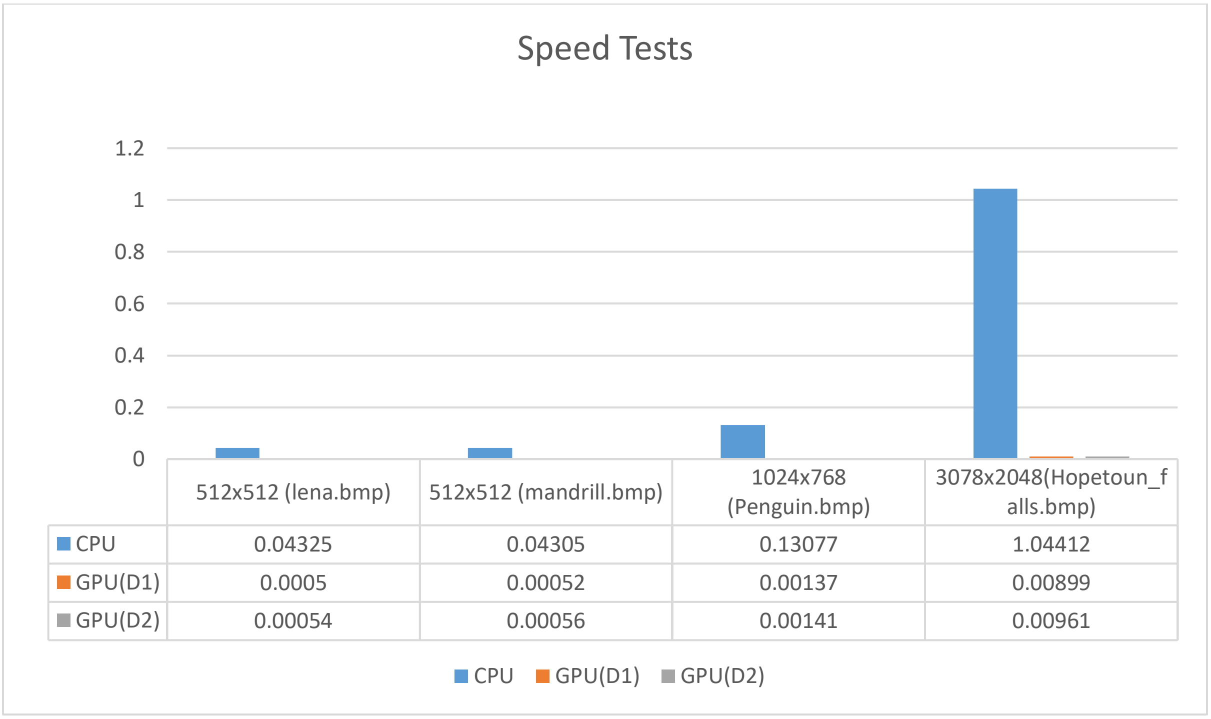
CS398 Report

Data Presentation for block size of 16



	512x512 (lena.bmp)	512x512 (mandrill.bmp)	1024x768 (Penguin.bmp)	3078x2048(Hopetoun_falls.bmp)
CPU	0.04325	0.04305	0.13077	1.04412
GPU(D1)	0.0005	0.00052	0.00137	0.00899
GPU(D2)	0.00054	0.00056	0.00141	0.00961

<u>Fig 1. Speed tests for each size of image.</u>

From the graph above, we can see that design 1 is faster than design 2.

Answered Questions

For the section below we are only considering an image size of 512x512.

1. How many multiplication/addition operations are being performed in your convolution kernel? explain.

   Number of masks = 8,
   Number of additions per mask = 9,
   Number of multiplications per mask = 9,
   Number of pixels = W*H*C = 512*512*3 = 786,432

   Number of total operations = 8*2*9*W*H*C = 113,246,208

2. How many global memory reads are being performed by your kernel? explain.

   Design 1
   Number of pixels = W*H*C = 512*512*3 = 786,432
   Tile width = 16, Block width = 18
   Number of blocks per width and height = ((512-1) / 16) +1= 32
   Number of total blocks = 32*32 * 3 = 3072
   Number of reads per block = 18 *18 = 324
   Number of total reads = total blocks * number of reads per block = 995,328

   Design 2
   Number of pixels = W*H*C = 512*512*3 = 786,432
   Tile width = 14, Block width = 16
   Number of blocks per width and height = ((512-1) / 14) +1= 37
   Number of total blocks = 37*37 * 3 = 1369 * 3 = 4107
   Number of reads per block = 16*16 = 256
   Number of total reads = total blocks * number of reads per block = 1,051,392

3. How many global memory writes are being performed by your kernel? explain.

   Number of global writes = Number of pixels = W*H*C = 512*512*3 = 786,432.
   For every thread, only one value is written to the final output. Therefore, the number of global writes is equal to the number of pixels

4. What is the minimum, maximum, and average number of real operations that a thread will perform? Real operations are those that directly contribute to the final output value.

   8 times of 9 additions and 9 multiplications, 1 division/multiplication for final result.

5. How much time is spent as an overhead cost for using the GPU for computation? Consider all code executed within your host function with the exception of the kernel itself, as overhead. How does the overhead scale with the size of the input?

   The gpu overhead is memcpy-ing to device local memory and memcpy-ing back. The overhead is significantly lesser as the size of the input increases.

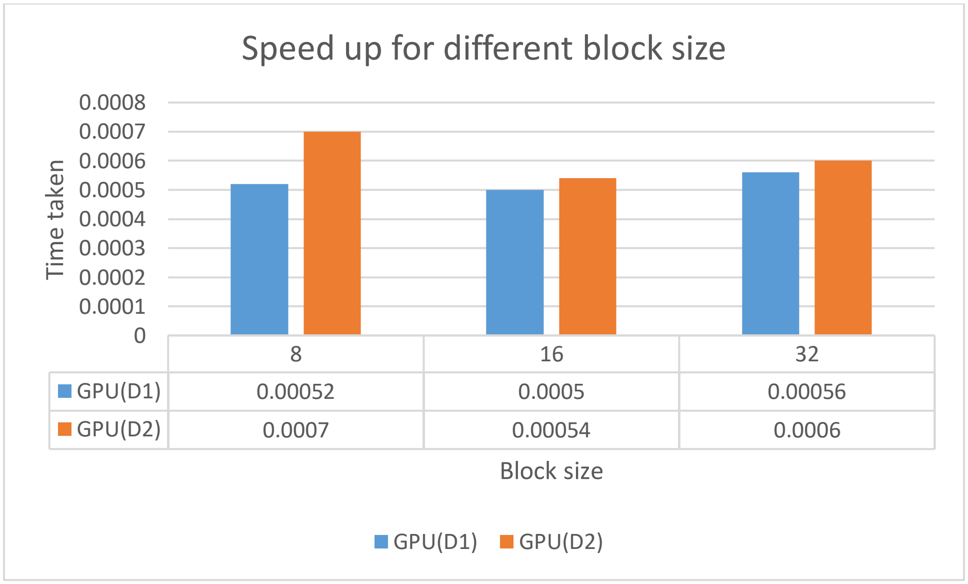
6. What do you think happens as you increase the mask size (say to 1024) while you set the block dimensions to 16x16? What do you end up spending most of your time doing? Does that put other constraints on the way you'd write your algorithm (think of the shared/constant memory size)?

   Each thread will have to load and sync from the input array for more block dimensions, essentially having a tile-based approach on the gpu. Most of the time will be spent on loading from the input array. The amount of shared memory has to be increased to hold the amount of masks that each tile will need, which may exceed the shared memory limitations of the device.

7. Do you have to have a separate output memory buffer? Put it in another way, why can't you perform the convolution in place?

   Yes. You cannot guarantee atomic read and write for the memory buffer, another thread may be reading the buffer while the current is writing to it, causing a race condition.

8. What is the best parameter for block size?



	8	16	32
🟦 GPU(D1)	0.00052	0.0005	0.00056
🟧 GPU(D2)	0.0007	0.00054	0.0006

Fig 2. Speed up for different block size for each design.

The best parameter for block size as seen in above table is 16 for both design 1 and design 2.