This survey collects your answers for the PIKA on Kirk & Hwu chapter 7 in CPSC 418 2017-2. In order to receive PIKA credit, you must enter your name and student number below.

Enter your last name (as it appears in Connect).

Enter your student number.

Consider the 2D convolution problem shown in K&H figure 7.4. The example provided gives an input value N[2,2] = 5 and an output value P[2,2] = 321. The input value N[3,3] = 7. What is the output value P[3,3]? (Remember: 2D indexing is "rows, columns".)

Consider the 2D convolution problem shown in K&H figure 7.4. The example provided gives an input value N[2,2] = 5 and an output value P[2,2] = 321. The input value N[5,0] = 6. What is the output value P[5,0]? (Remember: 2D indexing is "rows, columns".)

We use constant memory to store the convolution mask array because: (check all that apply)

- ☐ The mask array will not fit into shared memory.
- ☐ Loading the mask array data from constant memory is faster than loading it from global into shared memory.
- ☐ We cannot store two types of data (mask and image) in shared memory at the same time.
- ☐ Every thread will use the same mask array but will work with a different subset of the image array.
- ☐ We want to leave more shared memory for the image array in order to increase CGMA.
- ☐ Shared memory cannot handle the boundary conditions.
- ☐ The value of the image array entries remains fixed throughout the computation.
- ☐ We want to tile the mask array in order to increase CGMA.
- ☐ The value of the mask array entries remains fixed throughout the computation.

How can you load constant memory? (check all that apply)

- ☐ The CPU can do it using `cudaMemcpy()`.
- ☐ The GPU can do it using `cudaMemcpy()`.
- ☐ The CPU can do it using `cudaMemcpyToSymbol()`.
- ☐ The GPU can do it using `cudaMemcpyToSymbol()`.
- ☐ The threads on the GPU can do it using regular assignment statements.

→

Private Policy Terms of Use

Powered by Qualtrics

In the remaining questions, consider a 2D tiled convolution with a 21 x 21 mask array and image tiles of size 64 x 64 pixels. Both the mask and image arrays contain (single precision) floats.

How many halo elements are needed in the tile?

How many non-halo elements are needed in the tile?

How much constant memory (in bytes) is needed for the mask array?

How much shared memory (in bytes) is needed for the tile array?

If we use blocks with the maximum number of threads, how many output pixel elements will each thread need to compute? (Assume that you are using a GPU with CC 6.1.)

If we use blocks with the maximum number of threads, how many blocks can run on each SM? (Assume that you are using a GPU with CC 6.1.)