



GPU Teaching Kit
Accelerated Computing



Module 8.3 – Parallel Computation Patterns (Stencil)

Tile Boundary Conditions

Objective

- To learn to write a 2D convolution kernel
 - 2D Image data types and API functions
 - Using constant caching
 - Input tiles vs. output tiles in 2D
 - Thread to data index mapping
 - Handling boundary conditions

2D Image Matrix with Automated Padding

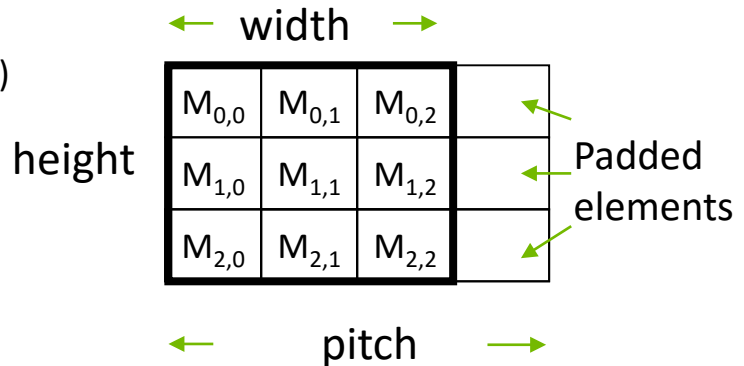
- It is sometimes desirable to pad each row of a 2D matrix to multiples of DRAM bursts
 - So each row starts at the DRAM burst boundary
 - Effectively adding columns
 - This is usually done automatically by matrix allocation function
 - Pitch can be different for different hardware
- Example: a 3X3 matrix padded into a 3X4 matrix

Height is 3

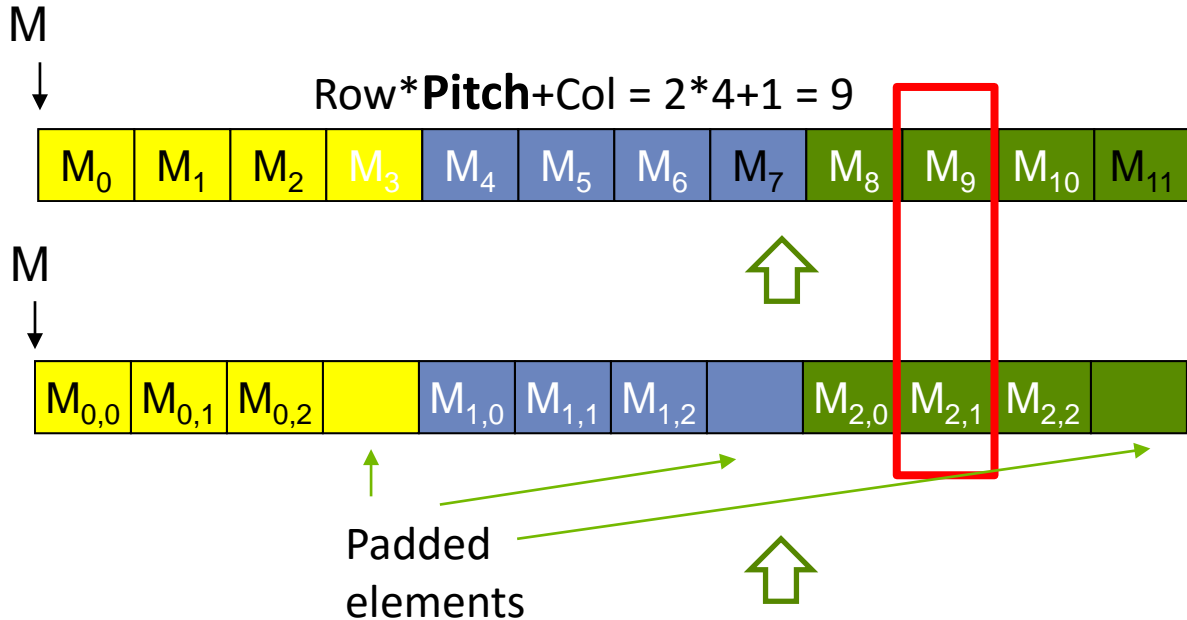
Width is 3

Channels is 1 (See MP Description)

Pitch is 4



Row-Major Layout with Pitch



M _{0,0}	M _{0,1}	M _{0,2}	
M _{1,0}	M _{1,1}	M _{1,2}	
M _{2,0}	M _{2,1}	M _{2,2}	

Image Matrix Type in this Course

```
// Image Matrix Structure declaration
//
typedef struct {
    int width;
    int height;
    int pitch;
    int channels;
    float* data;
} * wbImage_t;
```

This type will only be used in the host code of **the** MP.

wbImage_t API Function for Your Lab

```
wbImage_t  wbImage_new(int height, int  
width, int channels)  
wbImage_t  wbImport(char * File);
```

```
void wbImage_delete(wbImage_t img)
```

```
int  wbImage_getWidth(wbImage_t img)  
int  wbImage_getHeight(wbImage_t img)  
int  wbImage_getChannels(wbImage_t img)  
int  wbImage_getPitch(wbImage_t img)
```

```
float *wbImage_getData(wbImage_t img)
```

For simplicity, the pitch of all matrices are set to be width * channels (no padding) for our labs.

The use of all API functions has been done in the provided host code.

Setting Block Size

```
#define O_TILE_WIDTH 12
#define BLOCK_WIDTH (O_TILE_WIDTH + 4)

dim3 dimBlock(BLOCK_WIDTH,BLOCK_WIDTH);
dim3 dimGrid((wbImage_getWidth(N)-1)/O_TILE_WIDTH+1,
             (wbImage_getHeight(N)-1)/O_TILE_WIDTH+1, 1)
```

In general, BLOCK_WIDTH should be
 $O_TILE_WIDTH + (MASK_WIDTH-1)$

Using constant memory and caching for Mask

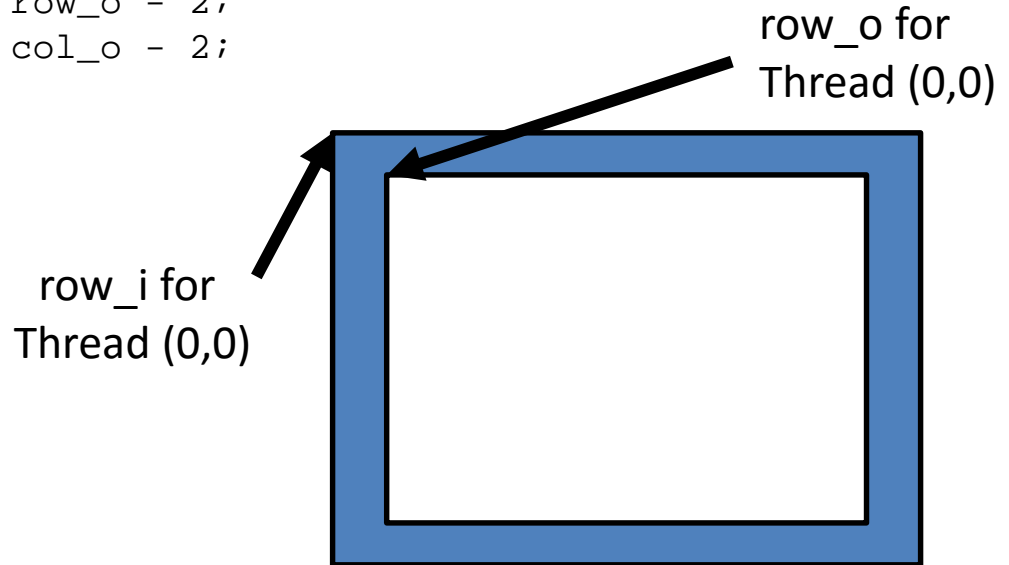
- Mask is used by all threads but not modified in the convolution kernel
 - All threads in a warp access the same locations at each point in time
- CUDA devices provide constant memory whose contents are aggressively cached
 - Cached values are broadcast to all threads in a warp
 - Effectively magnifies memory bandwidth without consuming shared memory
- Use of `const __restrict__` qualifiers for the mask parameter informs the compiler that it is eligible for constant caching, for example:

```
__global__ void convolution_2D_kernel(float *P,  
    float *N, height, width, channels,  
    const float __restrict__ *M) {
```


Shifting from output coordinates to input coordinate


```
int tx = threadIdx.x;  
int ty = threadIdx.y;  
int row_o = blockIdx.y*O_TILE_WIDTH + ty;  
int col_o = blockIdx.x*O_TILE_WIDTH + tx;
```

```
int row_i = row_o - 2;  
int col_i = col_o - 2;
```



Taking Care of Boundaries (1 channel example)

```
if((row_i >= 0) && (row_i < height) &&
    (col_i >= 0) && (col_i < width)) {
    Ns[ty][tx] = data[row_i * width + col_i];
} else{
    Ns[ty][tx] = 0.0f;
}
```



Use of width here is OK since pitch is set to width for this MP.

Some threads do not participate in calculating output. (1 channel example)

```
float output = 0.0f;
if(ty < O_TILE_WIDTH && tx < O_TILE_WIDTH){
    for(i = 0; i < MASK_WIDTH; i++) {
        for(j = 0; j < MASK_WIDTH; j++) {
            output += M[i][j] * Ns[i+ty][j+tx];
        }
    }
}
```

Some threads do not write output (1 channel example)

```
if(row_o < height && col_o < width)
    data[row_o*width + col_o] = output;
```

You need to write the kernel for a 3-channel (RGB) image.
See more details in the Lab MP Description.



GPU Teaching Kit

Accelerated Computing



The GPU Teaching Kit is licensed by NVIDIA and the University of Illinois under the [Creative Commons Attribution-NonCommercial 4.0 International License](https://creativecommons.org/licenses/by-nc/4.0/).