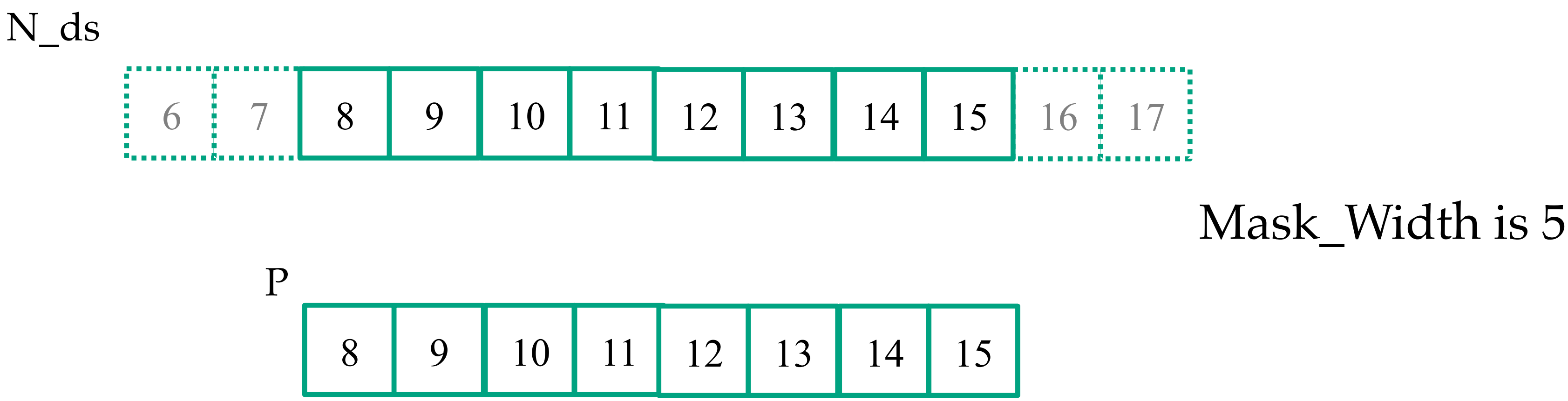
CS/EE 217 GPU Architecture and Parallel Programming

Lecture 9: Tiled Convolution Analysis

Objective

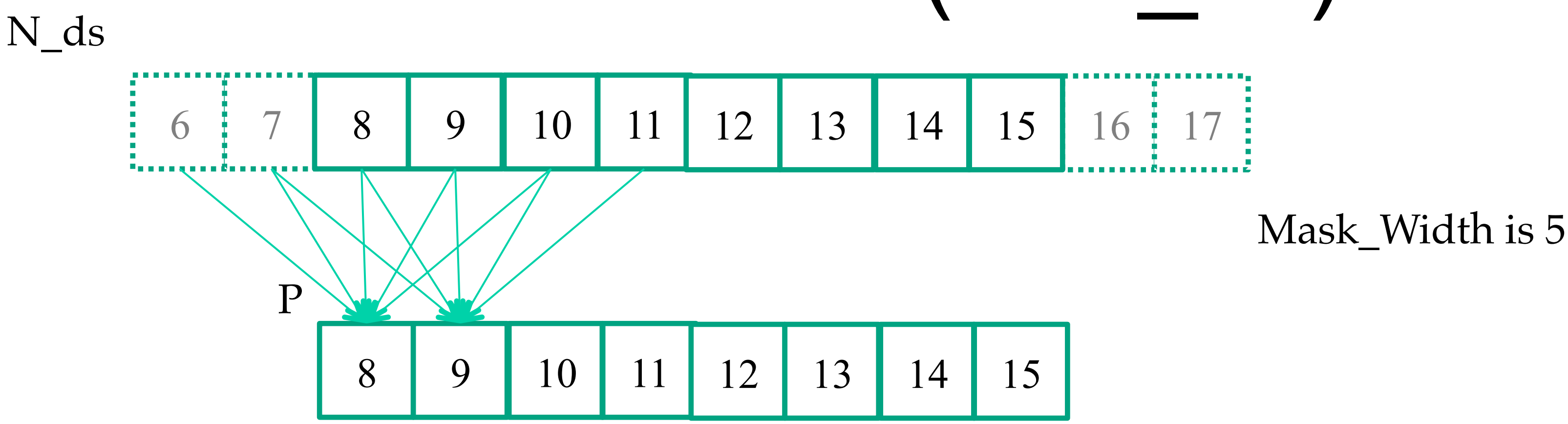
- To learn more about the analysis of tiled algorithms

If we used a larger (8 element) tile



- For Mask_Width = 5, we load 8+5-1 = 12 elements (12 memory loads)

Each output P element uses 5 N elements (in N_ds)



- P[8] uses N[6], N[7], N[8], N[9], N[10]
- P[9] uses N[7], N[8], N[9], N[10], N[11]
- P[10] uses N[8], N[9], N[10], N[11], N[12]
- …
- P[14] uses N[12], N[13], N[14], N[15],N[16]
- P[15] uses N[13], N[14], N[15], N[16], N[17]

A simple way to calculate tiling benefit

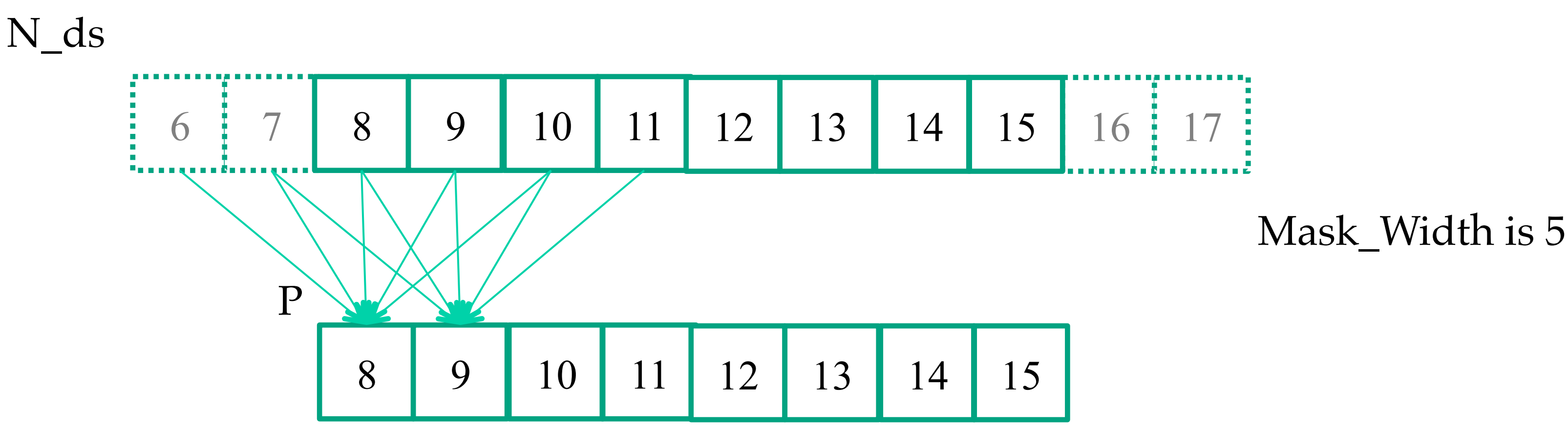
- (8+5-1)=12 elements loaded
- 8*5 global memory accesses replaced by shared memory accesses
- This gives a bandwidth reduction of 40/12=3.3

In General

- Tile_Width + Mask_Width -1 elements loaded
- Tile_Width * Mask_Width global memory accesses replaced by shared memory access
- This gives a reduction of bandwidth by

(Tile_Width *Mask_Width)/(Tile_Width+Mask_Width-1)

Another Way to Look at Reuse



- N[6] is used by P[8] (1X)
- N[7] is used by P[8], P[9] (2X)
- N[8] is used by P[8], P[9], P[10] (3X)
- N[9] is used by P[8], P[9], P[10], P[11] (4X)
- N[10] is used by P[8], P[9], P[10], P[11], P[12] (5X)
- … (5X)
- N[14] is uses by P[12], P[13], P[14], P[15] (4X)
- N[15] is used by P[13], P[14], P[15] (3X)

Another Way to Look at Reuse

- The total number of global memory accesses (to the (8+5-1)=12 N elements) replaced by shared memory accesses is

$$1 + 2 + 3 + 4 + 5 * (8-5+1) + 4 + 3 + 2 + 1$$

$$= 10 + 20 + 10$$

$$= 40$$

So the reduction is

$$40/12 = 3.3$$

Ghost elements change ratios

- For a boundary tile, we load Tile_Width + (Mask_Width-1)/2 elements
  - 10 in our example of Tile_Width =8 and Mask_Width=5
- Computing boundary elements do not access global memory for ghost cells
  - Total accesses is 3 + 4+ 6*5 = 37 accesses

The reduction is 37/10 = 3.7

In General for 1D

- The total number of global memory accesses to the (Tile_Width+Mask_Width-1) N elements replaced by shared memory accesses is

1 + 2 + … + Mask_Width-1+ Mask_Width * (Tile_Width -Mask_Width+1) + Mask_Width-1+… + 2 + 1

= (Mask_Width-1) *Mask_Width+ Mask_Width*(Tile_Width-Mask_Width+1)

= Mask_Width*(Tile_Width)

Bandwidth Reduction for 1D

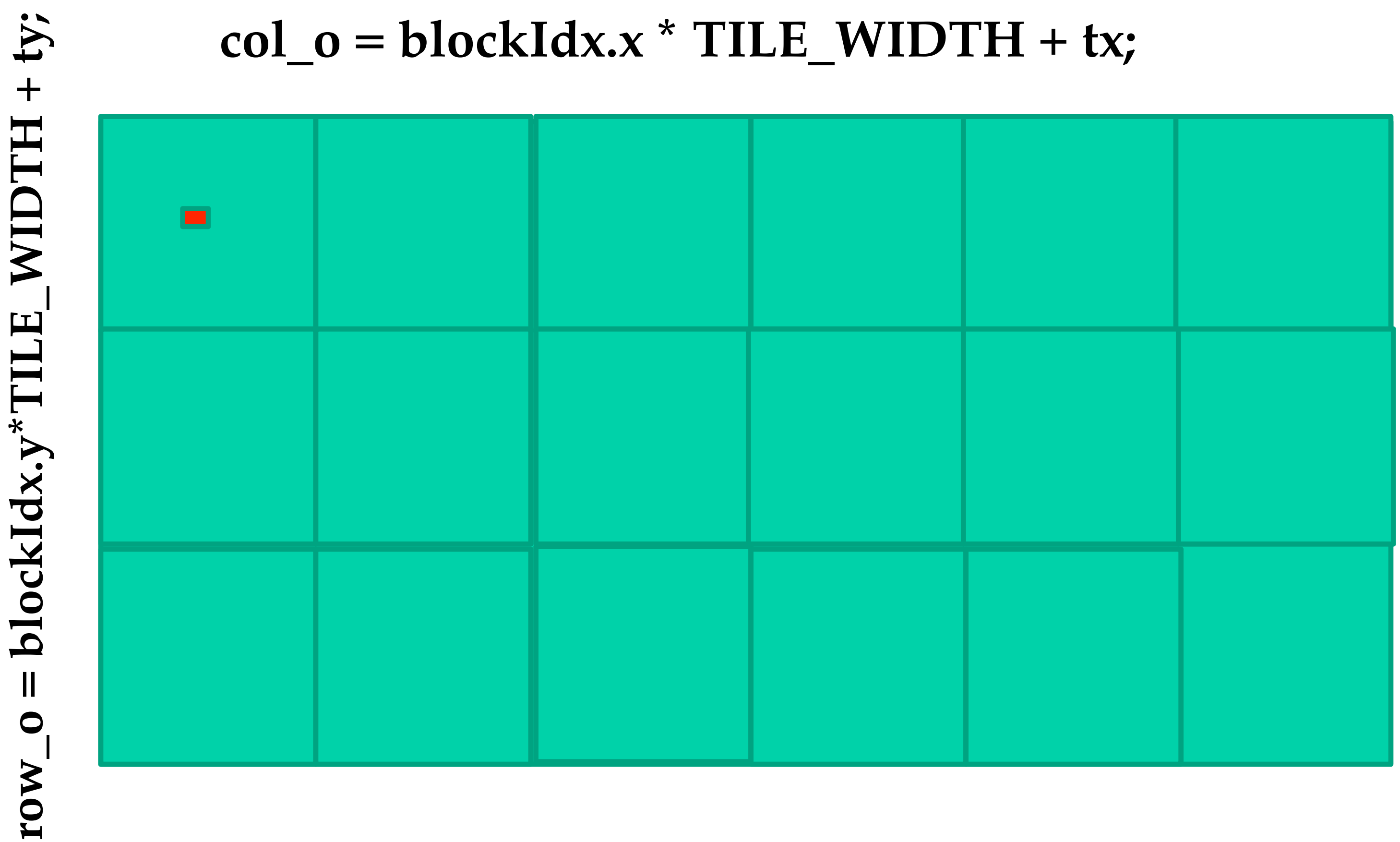
- The reduction is

Mask_Width * (Tile_Width)/(Tile_Width+Mask_Size-1)

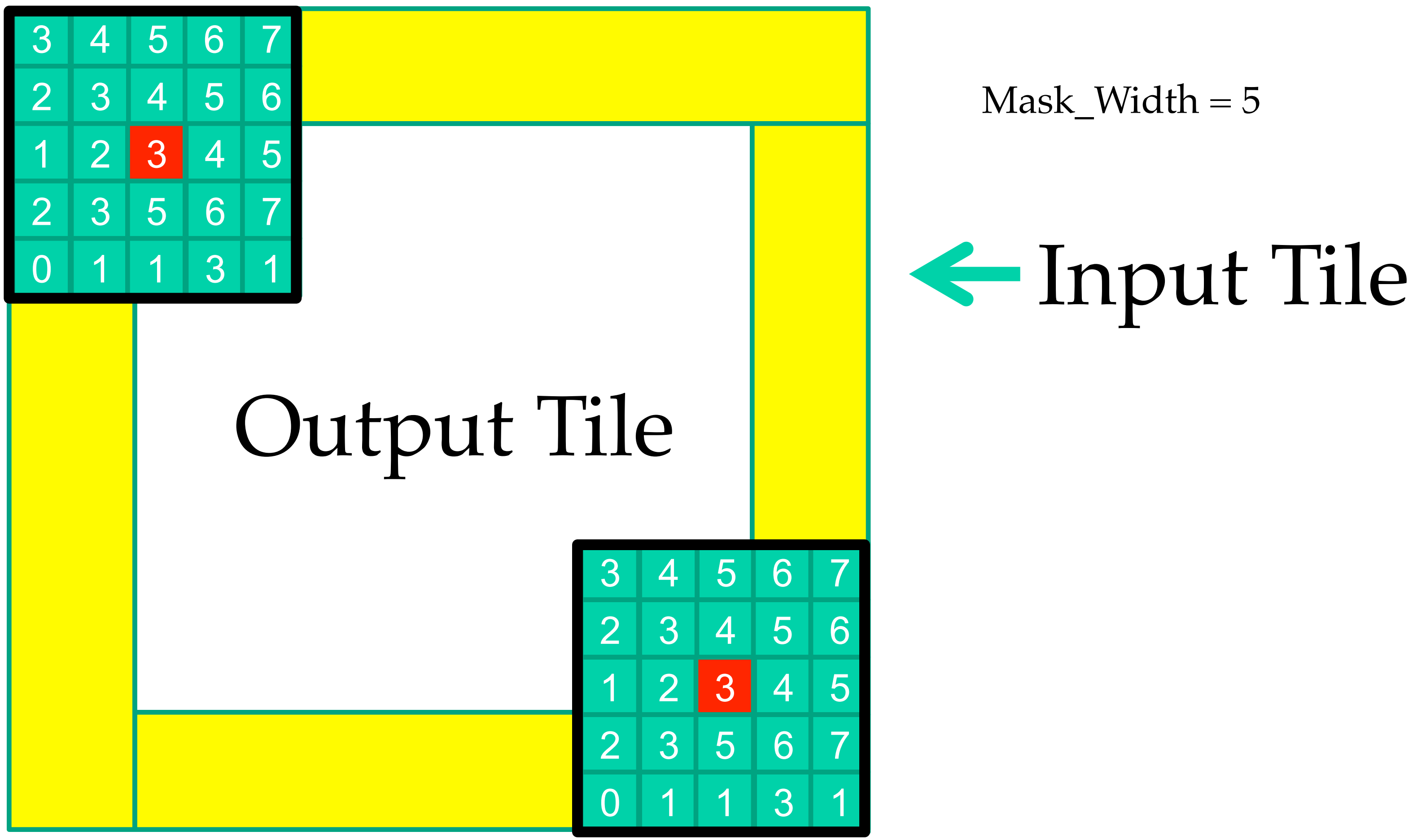
Tile_Width	16	32	64	128	256
Reduction Mask_Width = 5	4.0	4.4	4.7	4.9	4.9
Reduction Mask_Width = 9	6.0	7.2	8.0	8.5	8.7

2D Output Tiling and Indexin (P)

- Use a thread block to calculate a tile of P
  - Each output tile is of TILE_SIZE for both x and y



Input tiles need to cover halo elements.



A Simple Analysis for a small 8X8 output tile example

- 12X12=144 N elements need to be loaded into shared memory
- The calculation of each P element needs to access 25 N elements
- 8X8X25 = 1600 global memory accesses are converted into shared memory accesses
- A reduction of 1600/144 = 11X

In General

- $(Tile\_Width+Mask\_Width-1)^2$ elements from N need to be loaded into shared memory for each tile
- The calculation of each P element needs to access $Mask\_Width^2$ elements
  - $Tile\_Width^2 * Mask\_Width^2$ global memory accesses are converted into shared memory accesses
- The reduction is

$$Tile\_Width^2 * Mask\_Width^2 / (Tile\_Width+Mask\_Width-1)^2$$

Bandwidth Reduction for 2D

- The reduction is

$$Mask\_Width^{2} * (Tile\_Width)^{2} / (Tile\_Width + Mask\_Size - 1)^{2}$$

Tile_Width	8	16	32	64
Reduction Mask_Width = 5	11.1	16	19.7	22.1
Reduction Mask_Width = 9	20.3	36	51.8	64

Ghost elements change ratios

- Left as homewok.

ANY MORE QUESTIONS? READ CHAPTER 8