ECE408 / CS483 / CSE408
Summer 2025

Applied Parallel Programming

Lecture 7: DRAM Bandwidth

What Will You Learn Today?

- organization of memory based on dynamic RAM (DRAM)
- the use of burst mode and multiple banks (both sources of parallelism) to increase DRAM performance (data rate)
- memory access coalescing, which connects GPU kernel performance to DRAM organization

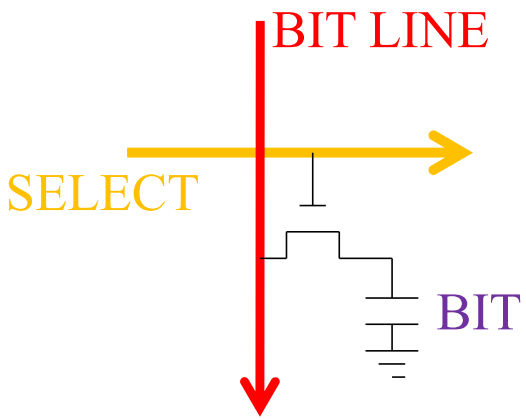
Global Memory (DRAM) Bandwidth

Ideal

Reality

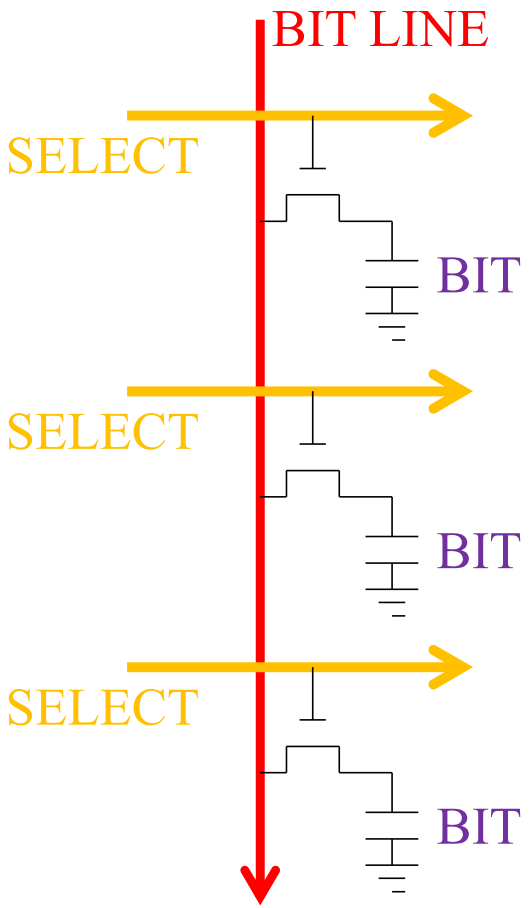
Most Large Memories Use DRAM

- **Random Access Memory (RAM)**:
  same time needed to read/write any address
- **Dynamic RAM (DRAM):**
  - **bit** stored on a capacitor
  - connected via transistor to **bit line** for read/write
  - **bits disappear** after a while (around 50 msec, due to tiny leakage currents through transistor), **and must be rewritten** (hence dynamic)



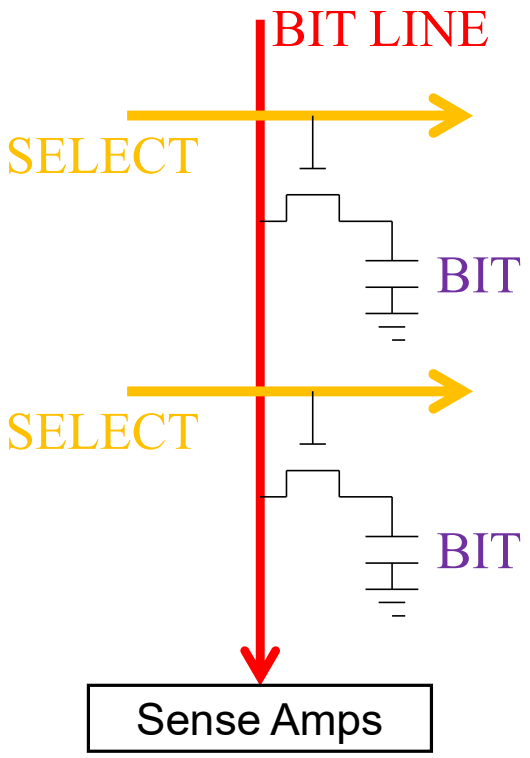
Many Cells (Bits) per Bit Line

- About **1,000 cells** connect to **each BIT LINE**.
- **Connection**/disconnection **depends** on **SELECT** line.
- Some **address bits** decoded to **connect exactly one cell to** the **BIT LINE**.

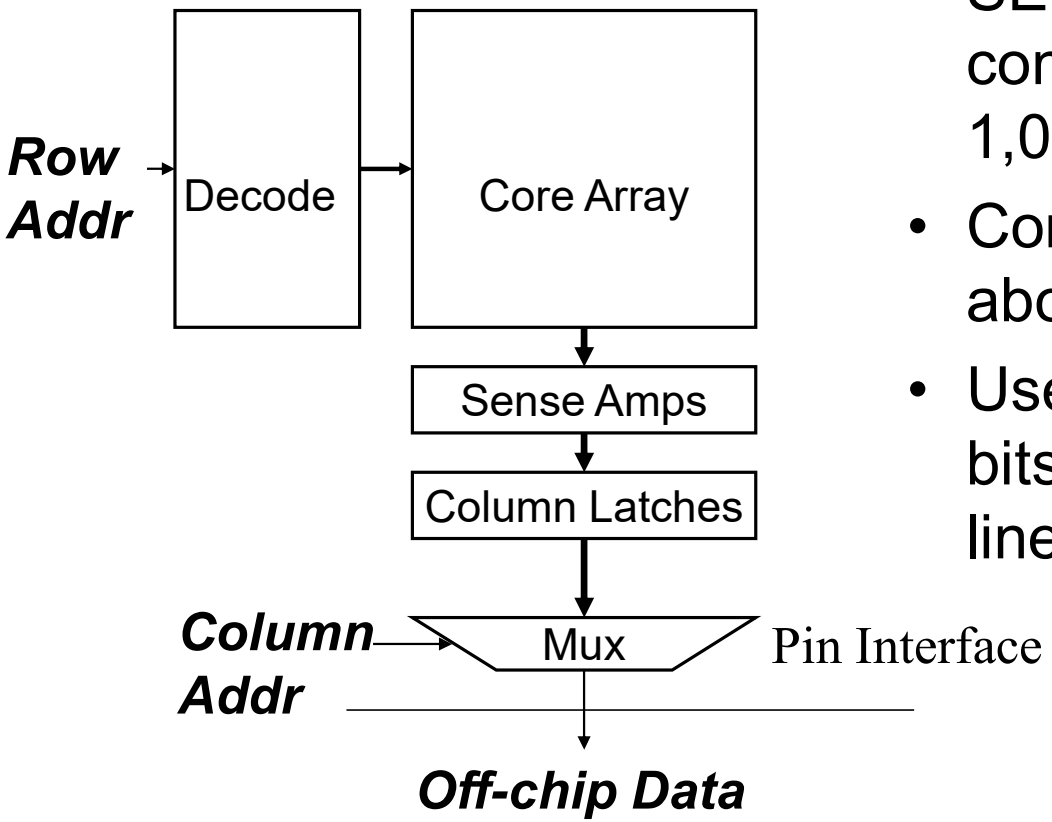


DRAM is Slow But Dense

- Capacitance…
  - tiny for the **BIT**, but
  - huge for the **BIT LINE**
- Use an amplifier for higher speed!
- Still **slow**…
- But only need **1 transistor per bit**.



DRAM Bank Organization



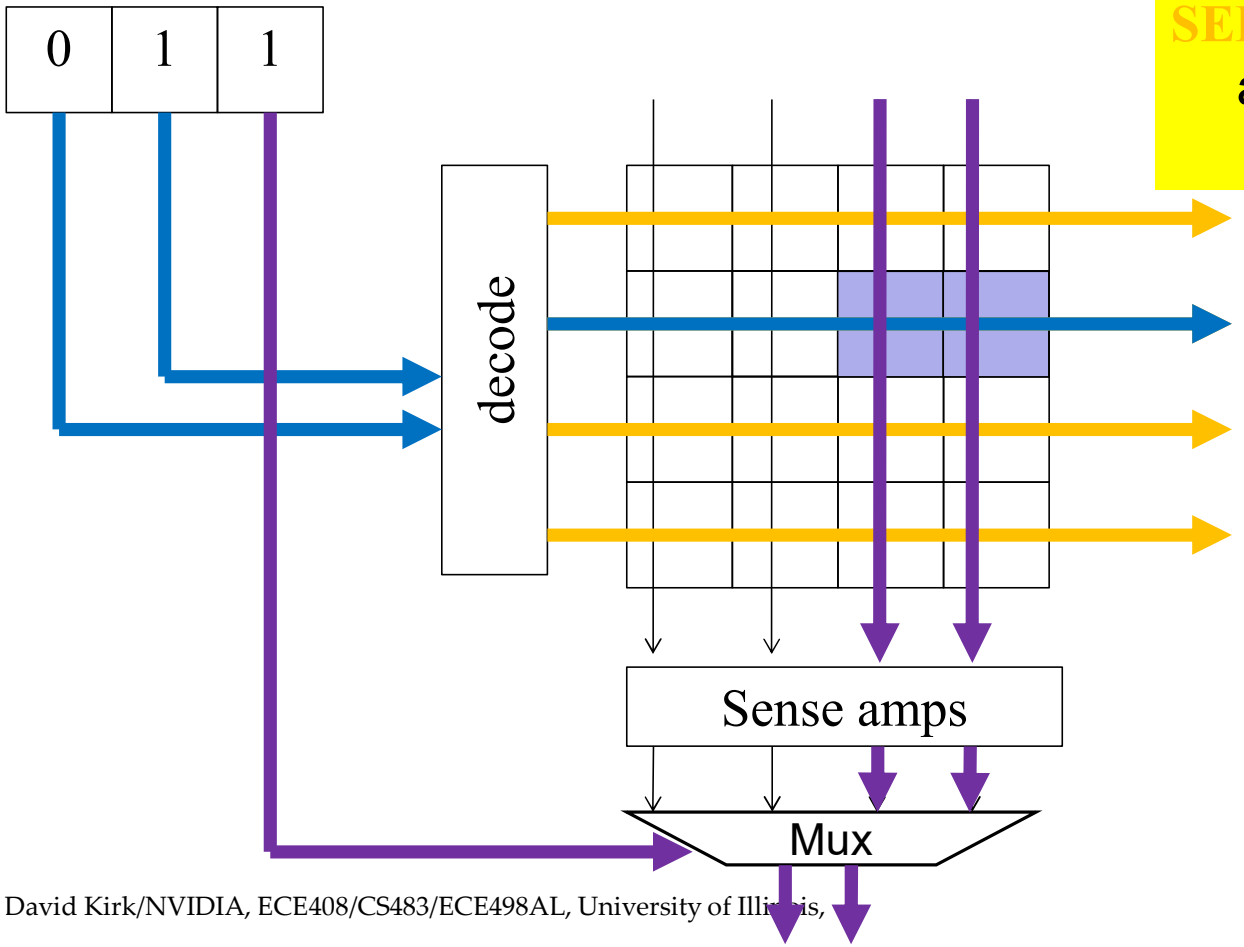
- SELECT lines connect to about 1,000 bit lines.
- Core array has about O(1M) bits
- Use more address bits to choose bit line(s).

DRAM Interfaces are Clocked

- DRAM **cells** are **not clocked** (clocking requires transistors).

- DRAM **interfaces** are **clocked**.
  - DDR: Core speed = ½ interface speed
  - DDR2/GDDR3: Core speed = ¼ interface speed
  - DDR3/GDDR4: Core speed = ⅛ interface speed
  - … likely to be worse in the future

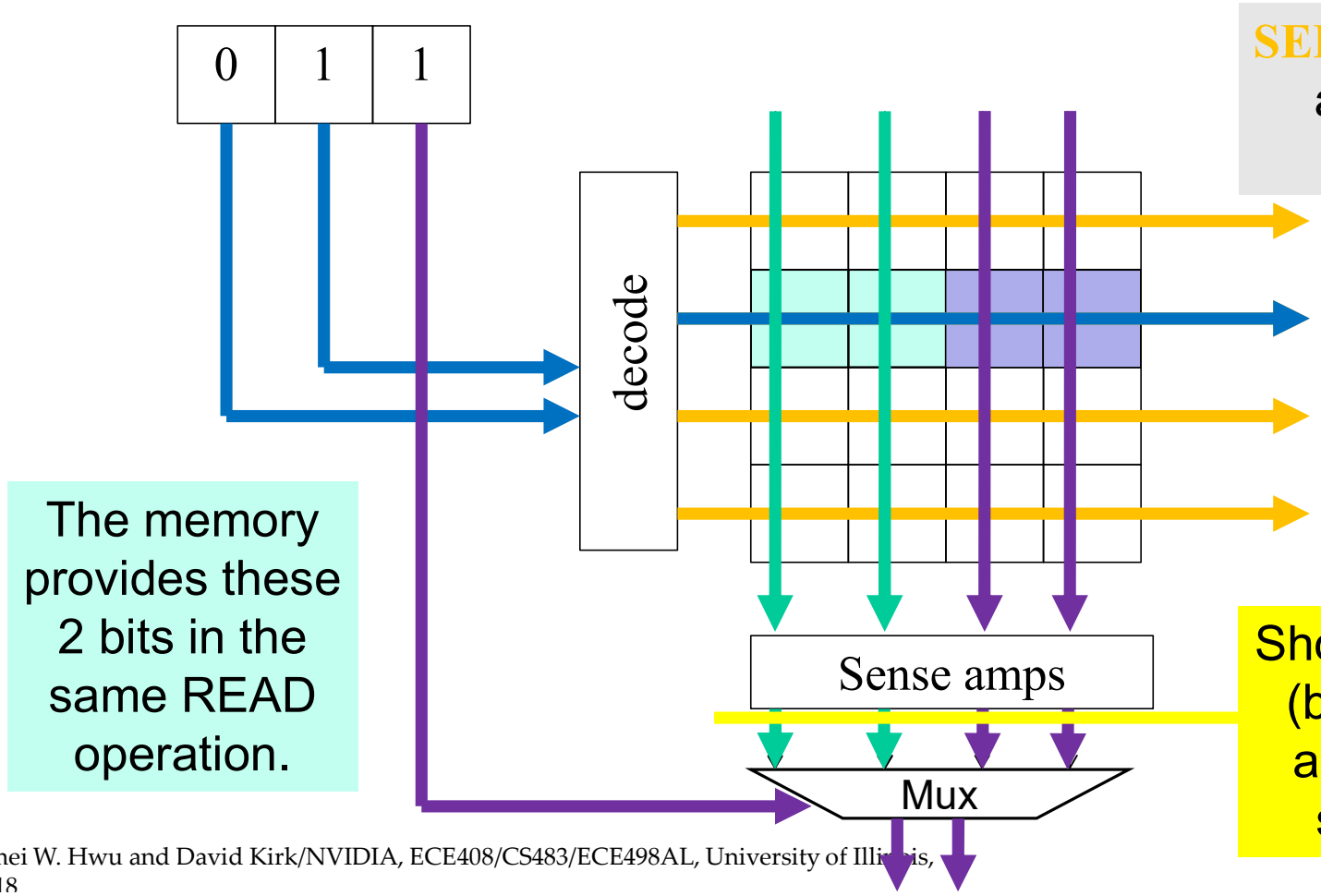
Consider a READ Operation from an 8×2-bit DRAM



SELECT lines with active line 01 highlighted.

Let's READ these two bits.

The Other 2 Bits on the Line are “Free”



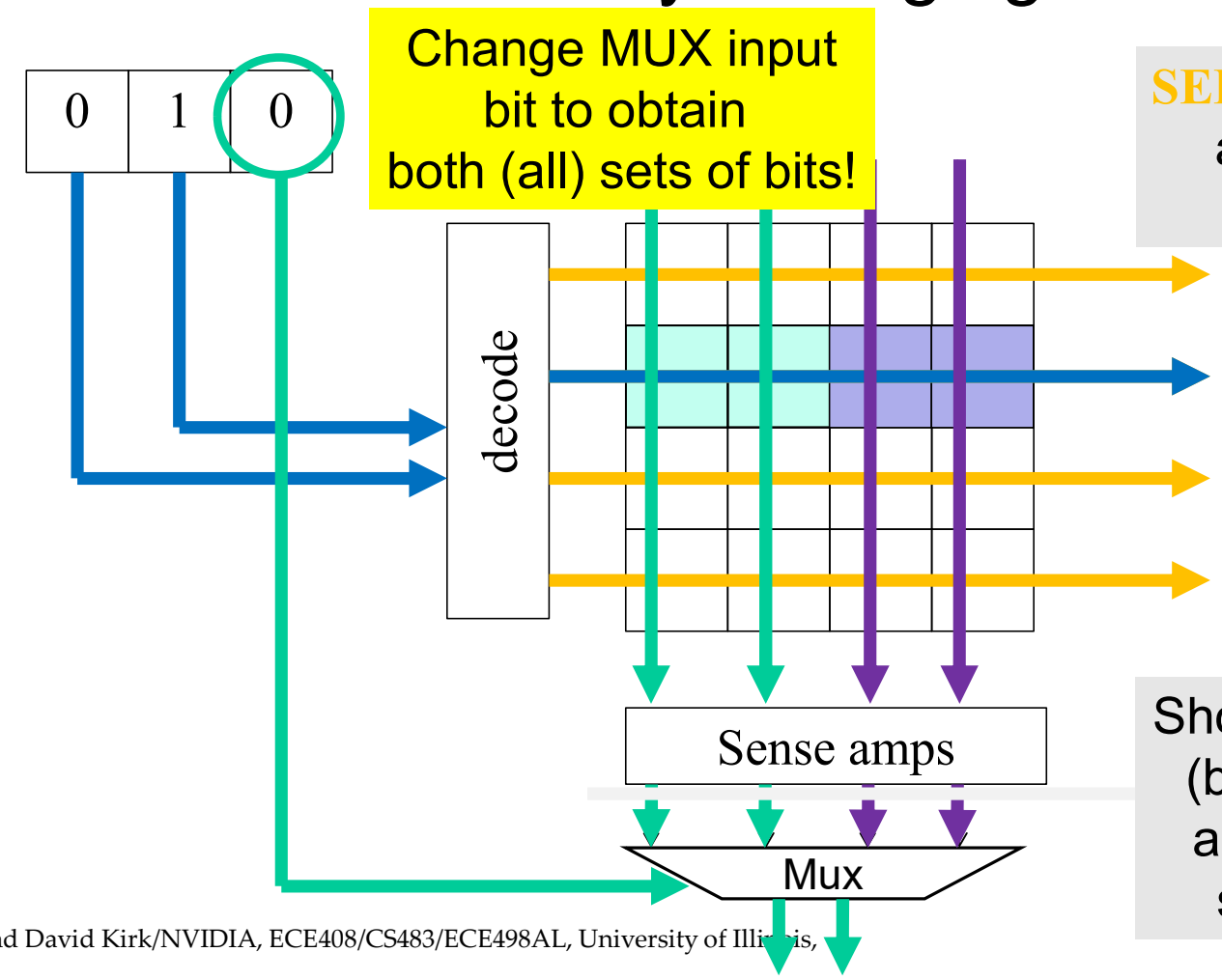
SELECT lines with active line 01 **highlighted**.

We wanted these two bits.

The memory provides these 2 bits in the same READ operation.

Shown previously (but not here): all outputs are stored here.

Obtain a 4-bit "Burst" by Changing MUX Input Bit

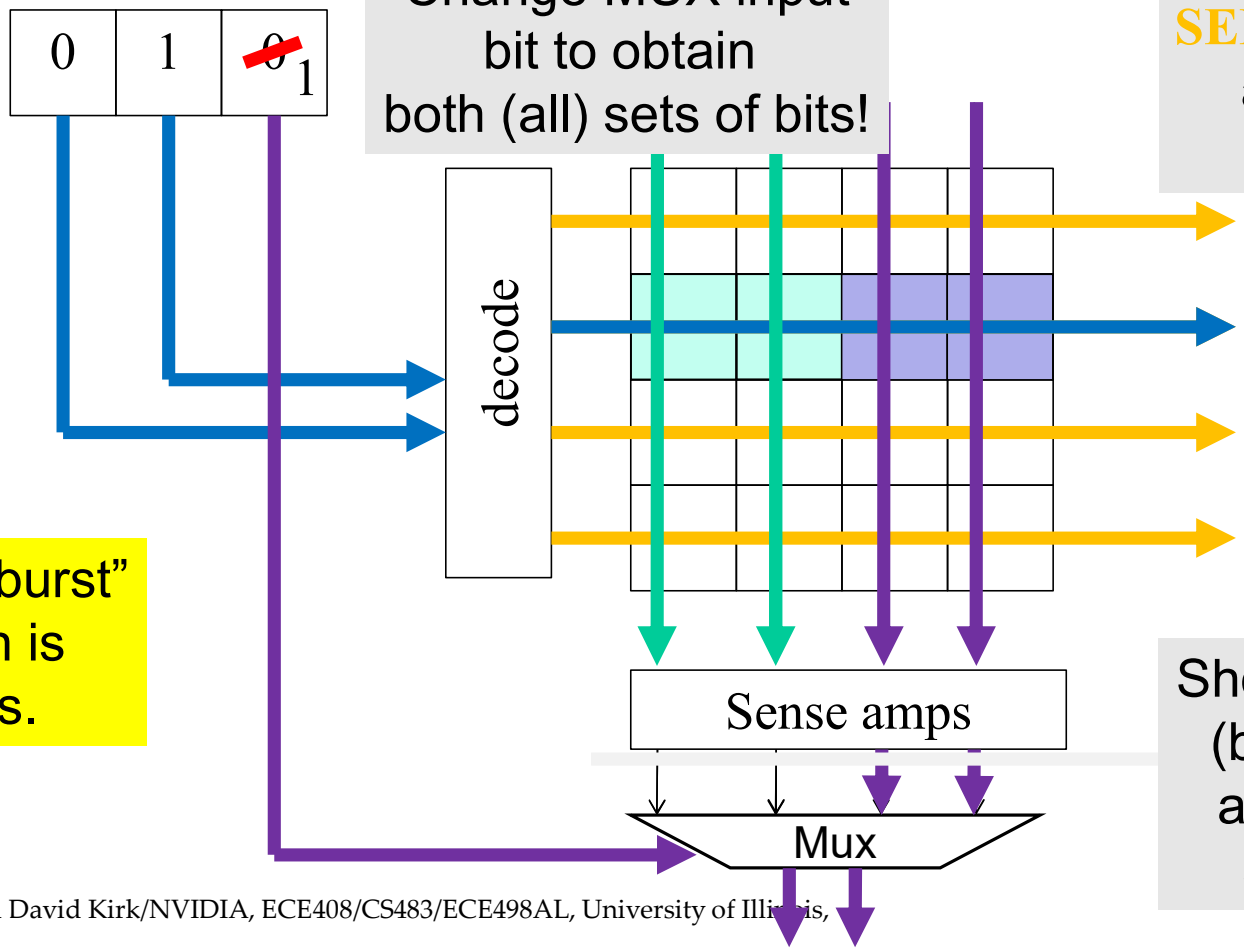


SELECT lines with active line 01 **highlighted**.

First step: READ using MUX select 0.

Shown previously (but not here): all outputs are stored here.

Second Step: READ Other 2 Bits



Change MUX input bit to obtain both (all) sets of bits!

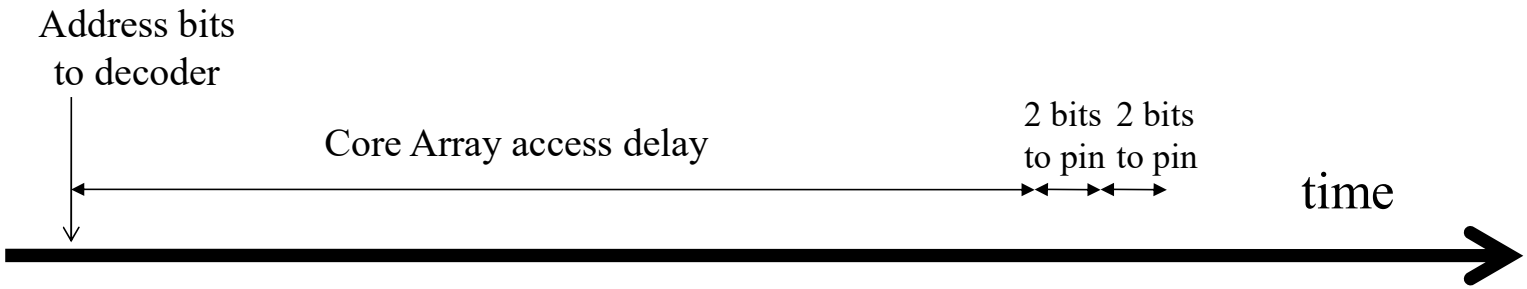
SELECT lines with active line 01 **highlighted**.

Second step: read another 2 bits.

DRAM "burst" length is 4 bits.

Shown previously (but not here): all outputs are stored here.

DRAM Bursting Increases Useful Activity

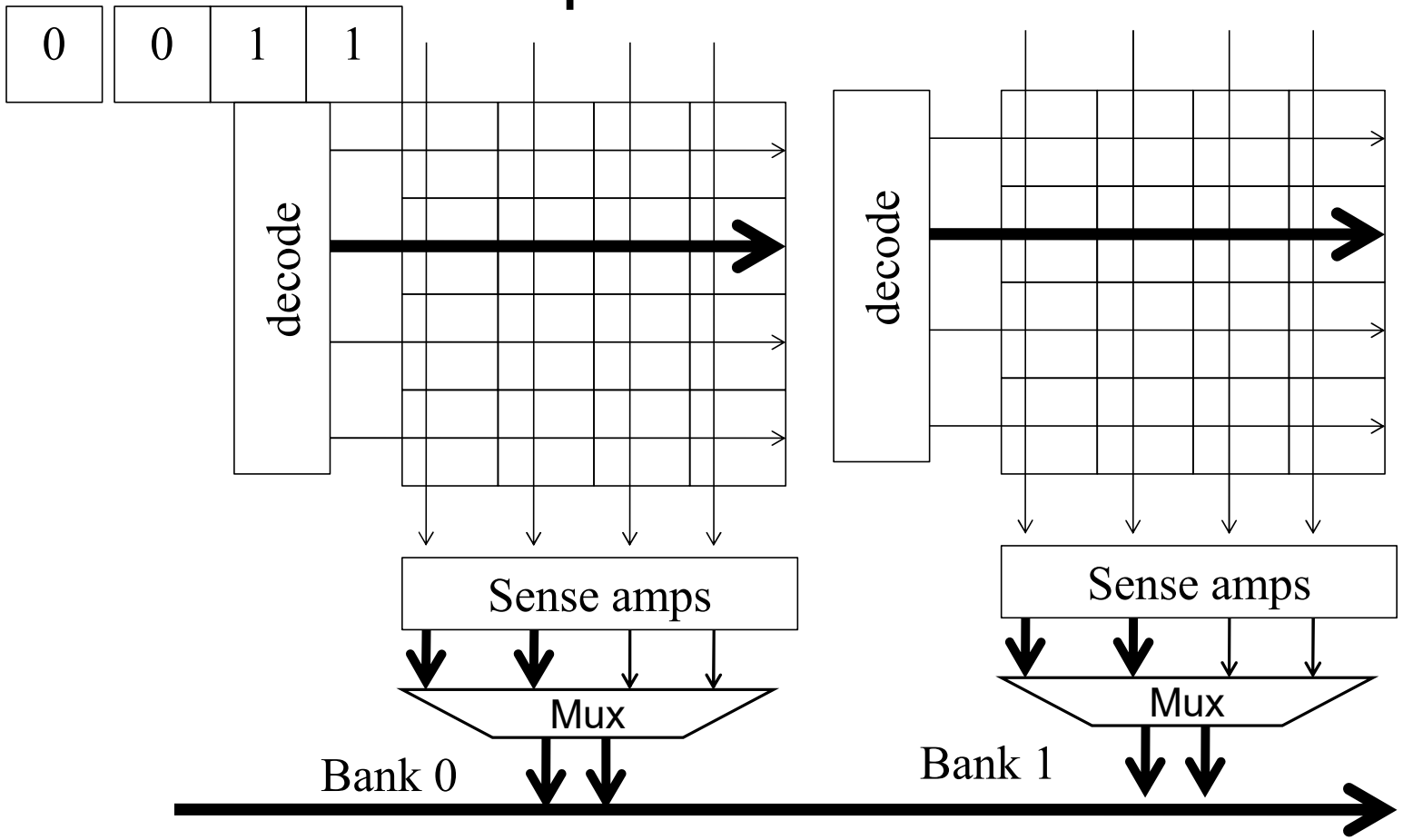


Non-burst timing

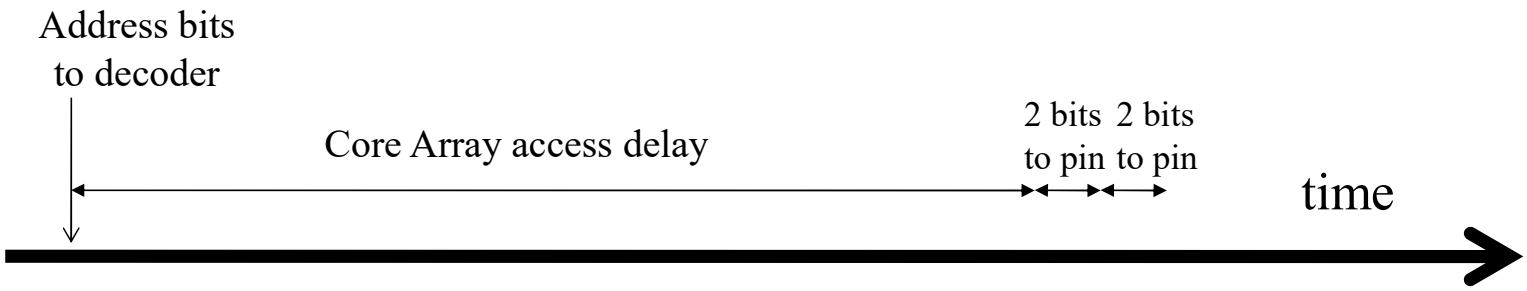
Burst timing

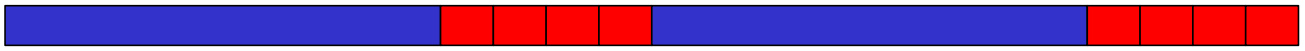
Modern DRAM systems are designed to be always accessed in burst mode. Burst bytes are transferred but discarded when accesses are not to sequential locations.

Multiple DRAM Banks

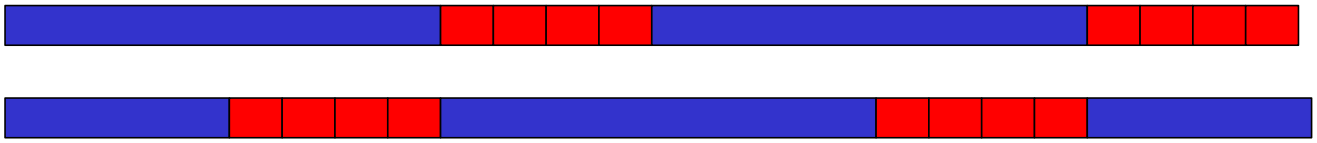


DRAM Bursting for the 8x2 Bank



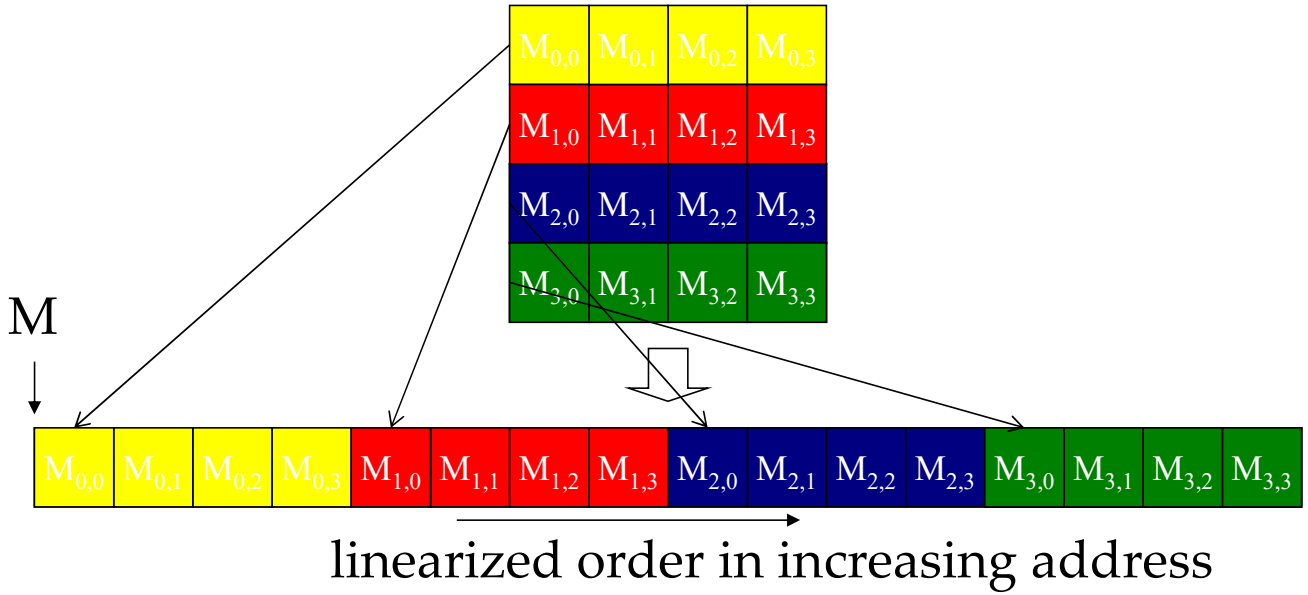


Single-Bank burst timing, dead time on interface



Multi-Bank burst timing, reduced dead time

Review: 2D Array Linearization in C



Review: A Simple Matrix Multiplication Kernel

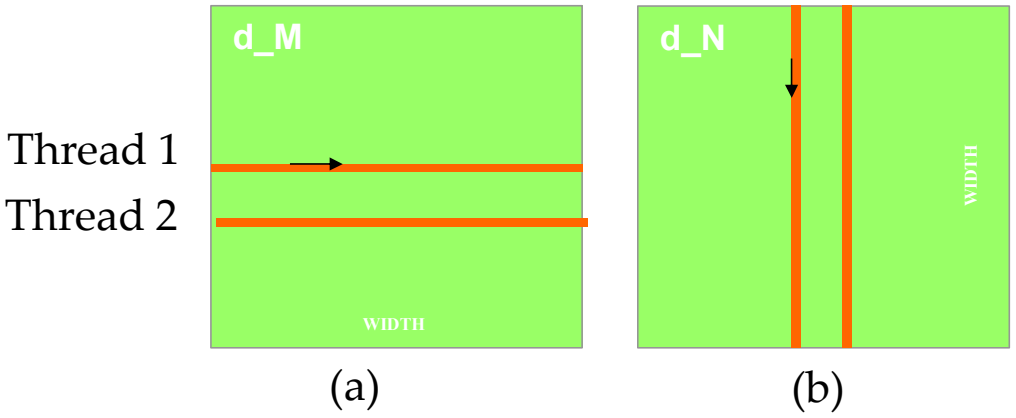
```
__global__ void MatrixMulKernel(float* M, float* N, float* P, int Width)
{
 // Calculate the row index of the P element and M
 int Row = blockIdx.y * blockDim.y + threadIdx.y;
 // Calculate the column index of P and N
 int Col = blockIdx.x * blockDim.x + threadIdx.x;

 if ((Row < Width) && (Col < Width)) {
    float Pvalue = 0;

    // each thread computes one element of the block sub-matrix
    for (int k = 0; k < Width; ++k)
      Pvalue += M[Row*Width+k] * N[k*Width+Col];

    P[Row*Width+Col] = Pvalue;
  }
}
```

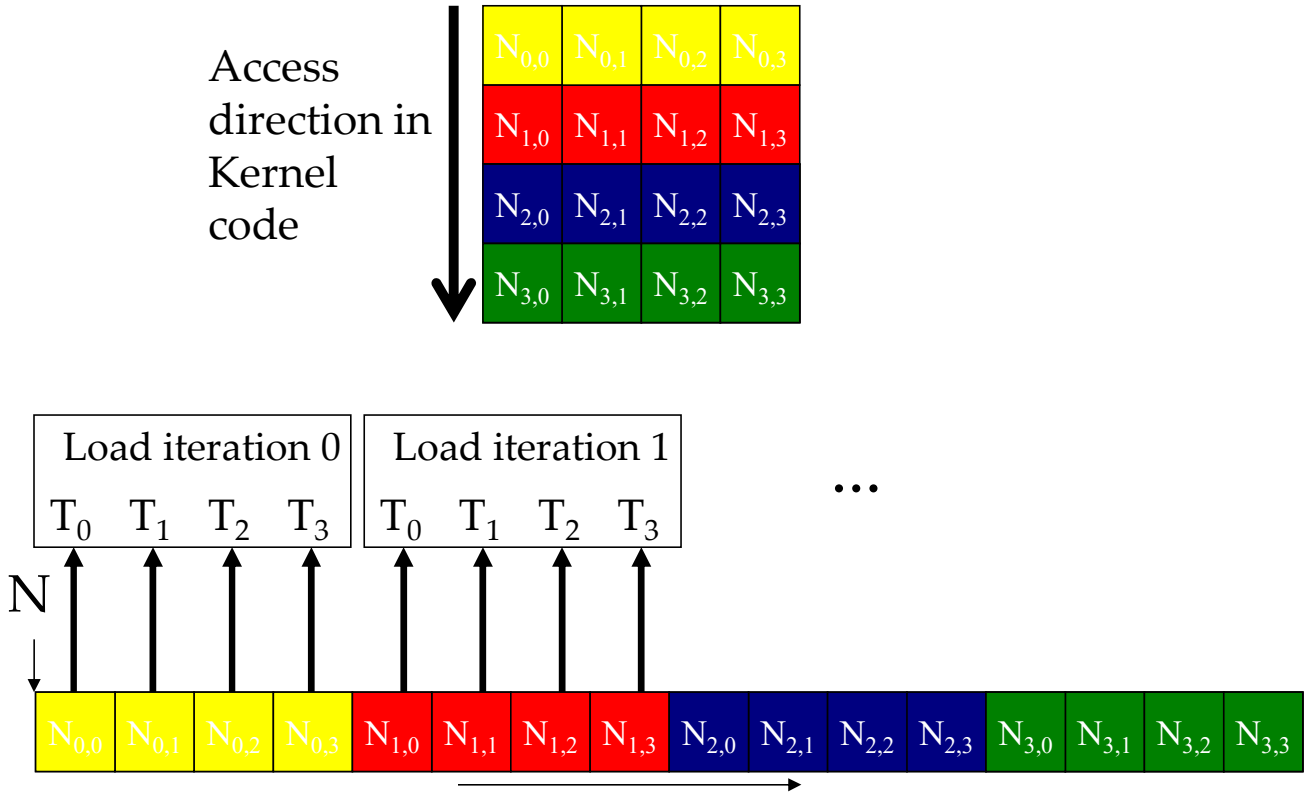
Two Access Patterns



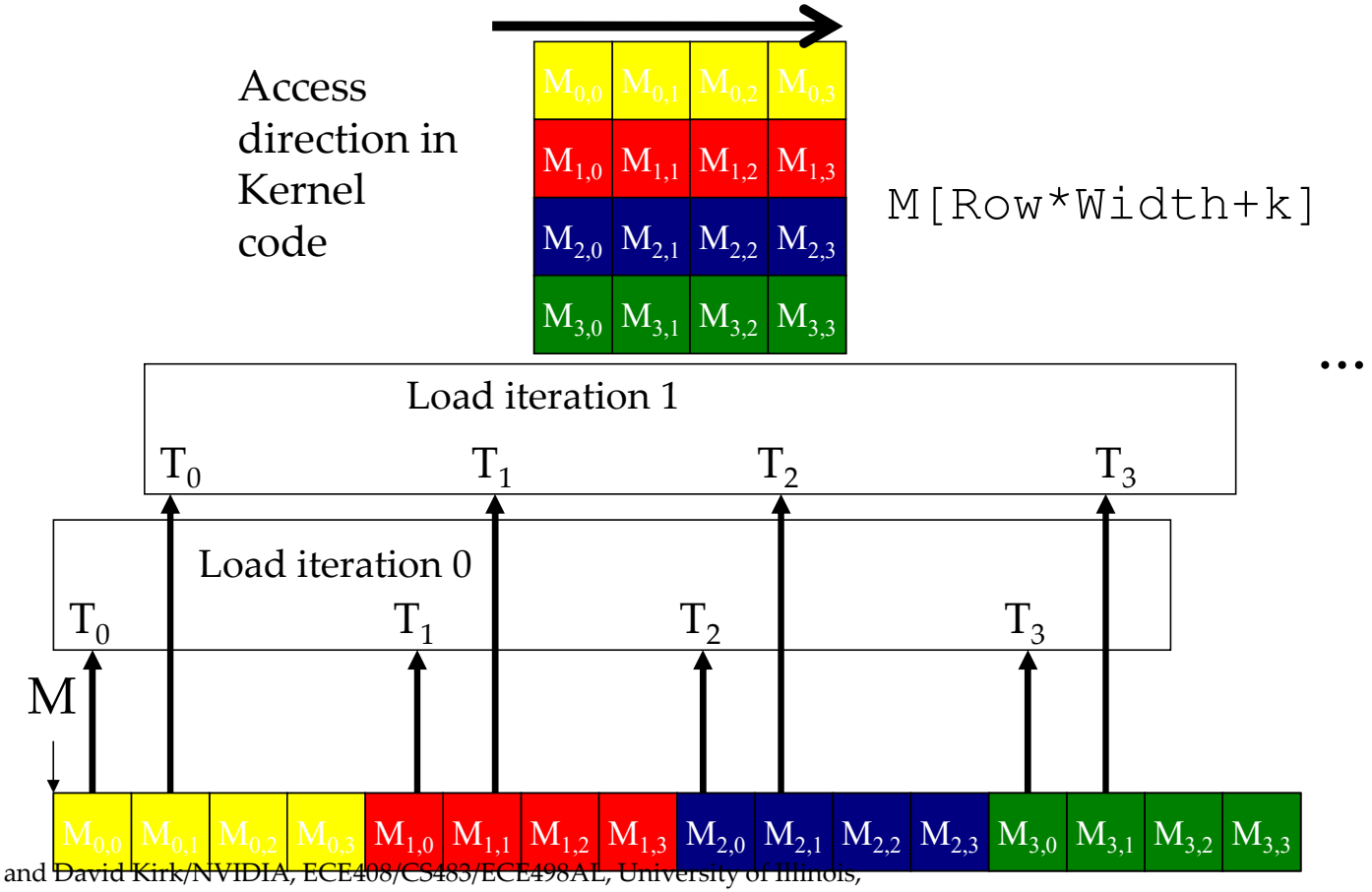
`M[Row*Width+k]` `N[k*Width+Col]`

k is loop counter in the inner product loop of the kernel code

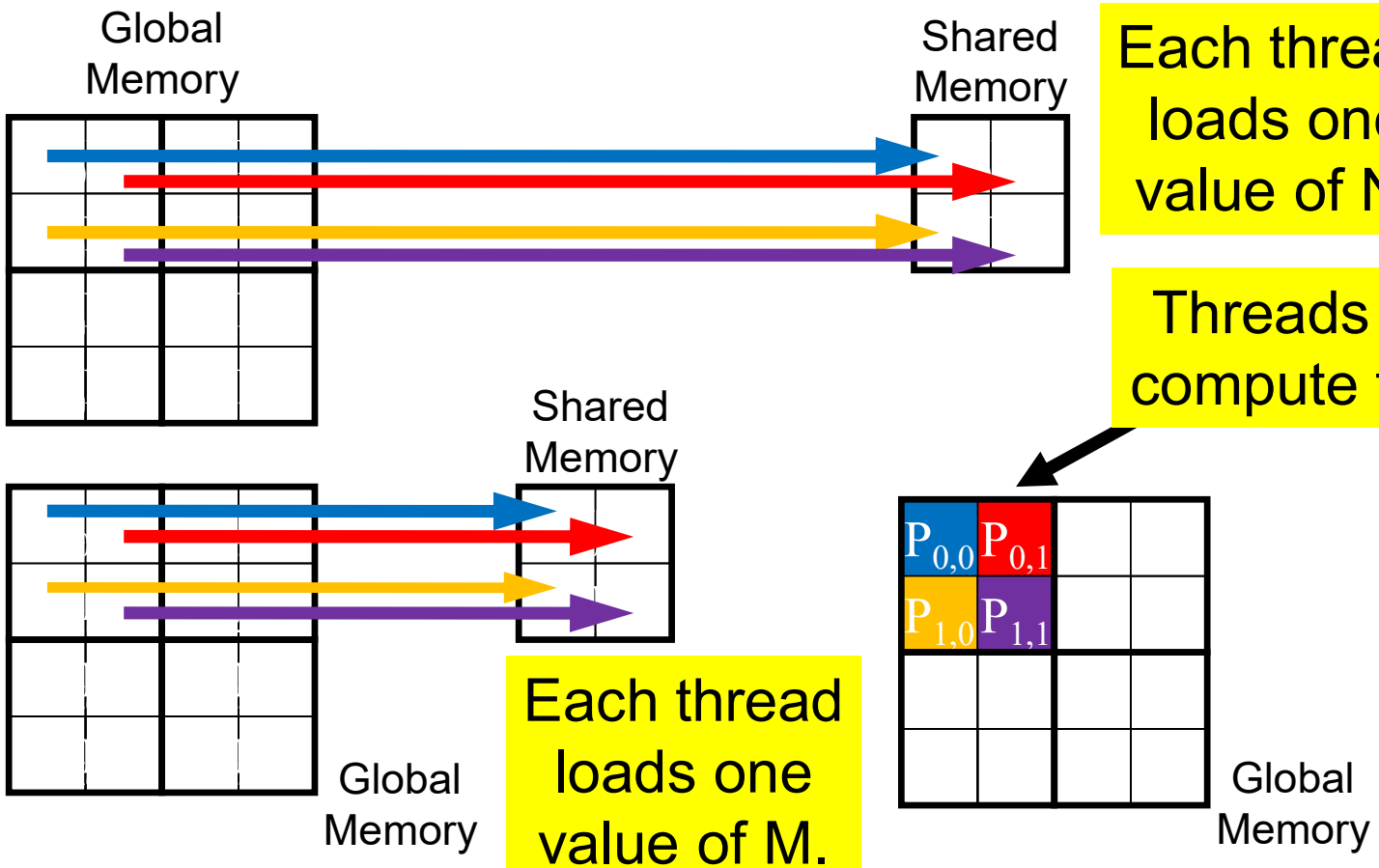
Accesses to N are Coalesced



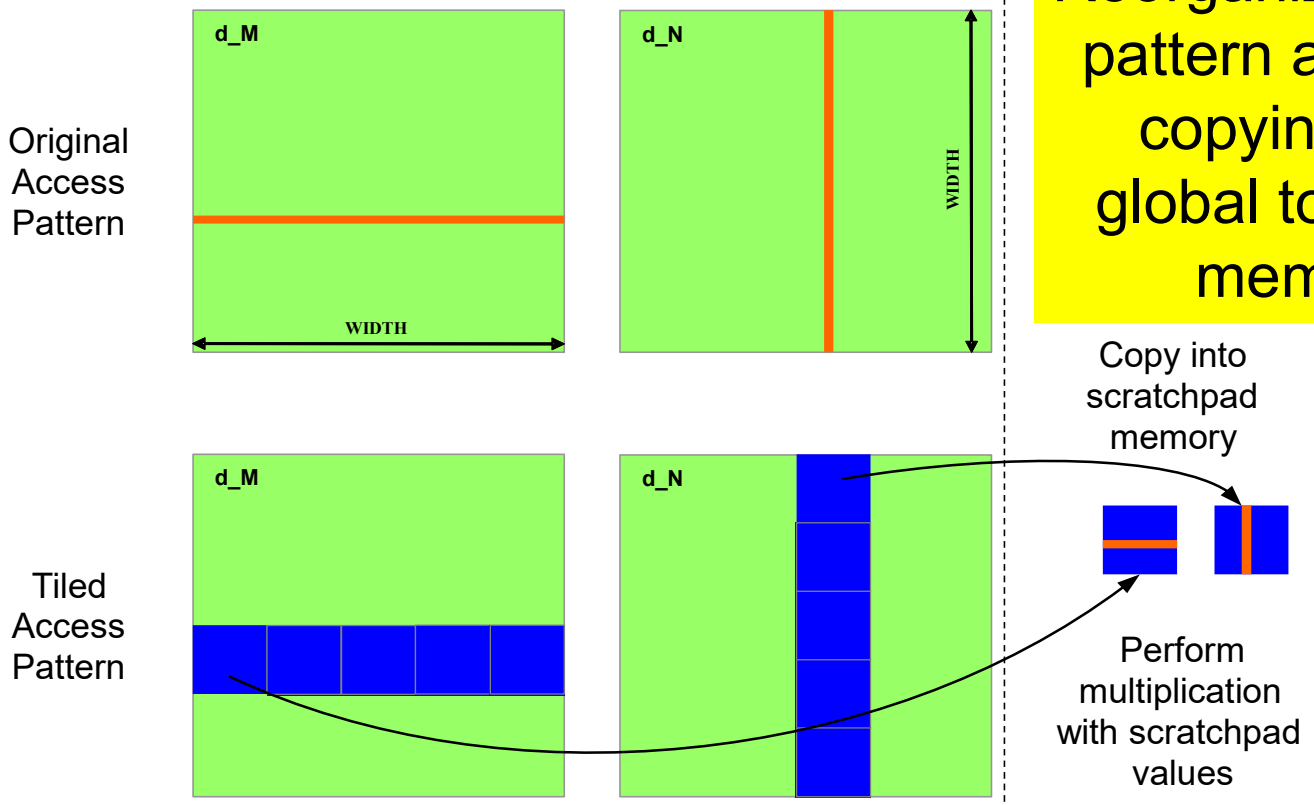
Accesses to M are NOT Coalesced!



Recall: First Tile Load by Thread Block (0,0)



Use of Shared Memory Enables "Corner Turning"



Reorganize access pattern as part of copying from global to shared memory!

QUESTIONS?

READ CHAPTER 5!

Problem Solving

- Q: Which of the following two lines of a kernel code has better performance?

```
    int tx = blockIdx.x * blockDim.x + threadIdx.x;
    int ty = blockIdx.y * blockDim.y + threadIdx.y;
A)  B[ty * Width + tx] = 2 * A[ty * Width + tx];
B)  B[tx * Width + ty] = 2 * A[tx * Width + ty];
```

A: Remember that X is the smallest dimension for linearizing thread indices, so "adjacent" in X (same Y, Z) often means the same warp.

Threads in a warp should access adjacent memory locations, so line A is better than line B.

Problem Solving

Q: Consider a DRAM system with a burst size of 512 bytes and a peak bandwidth of 240 GB/s. Assume a thread block size of 1024 and warp size of 32 and that A is a floating-point array in the global memory. What is the maximal memory data access throughput we can hope to achieve in the following access to A?

```
int i = blockIdx.x * blockDim.x + threadIdx.x;
float temp = A[4*i] + A[4*i+1];
```

A: From a burst of 512 bytes, we will only use every other 8 bytes due to reading from [4*i] and [4*i+1] memory locations.

So, we can expect 120 GB/s.

In early GPUs, with no caching, the two loads might be separated enough to cut that bandwidth to 60 GB/s (4 of every 16B used per load).