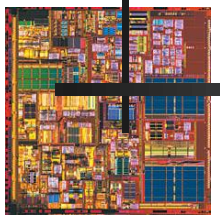
ECE 411
Fall 2015



Lecture 4

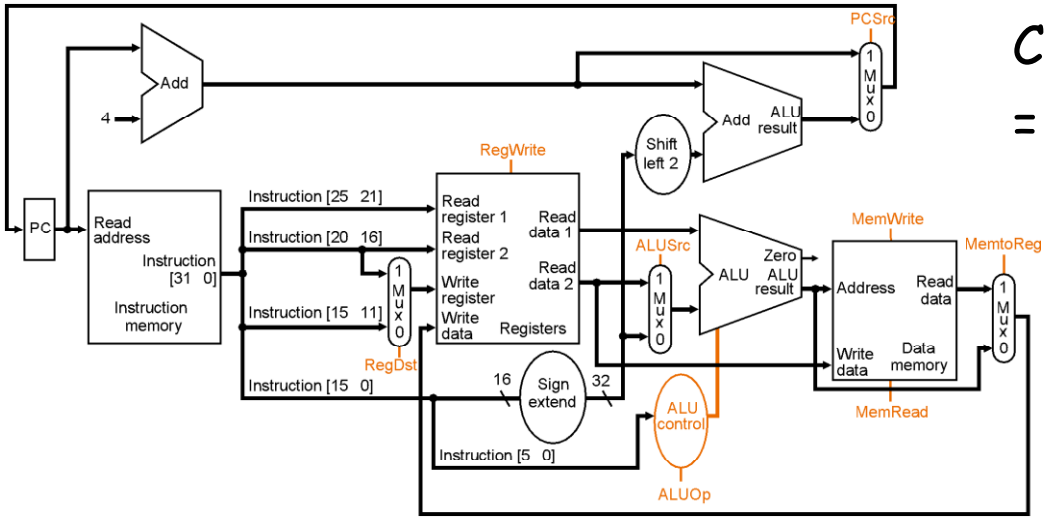
Performance Optimization Basics; Memory Hierarchy

ILLINOIS

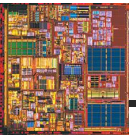
A Hypothetical Data Path Timing Analysis

Critical path: a path through combinational circuit that takes as long or longer than any other.

	I Fetch	Decode, R-Read	ALU	PC update	D Memory	R-Write	Total (ns)
Add	1	1	.9	-	-	.8	3.7
Load	1	1	.9	-	1	.8	4.7
Store	1	1	.9	-	1	-	3.9
beq	1	1	.9	.1	-	-	3.0



Clock cycle time
= 4.7ns + setup + hold



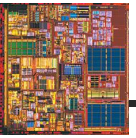
Cycle Time vs. CPI

Goal: balance amount of work done each cycle.

	I Fetch	Decode, R-Read	ALU	PC update	D Memory	R-Write	Total
Add	1	1	.9	-	-	.8	3.7
Load	1	1	.9	-	1	.8	4.7
Store	1	1	.9	-	1	-	3.9
beq	1	1	.9	.1	-	-	3.0

- Load needs 5 cycles
- Add and Store need 4
- beq needs 3

ILLINOIS



Will multicycle design be faster?

	I Fetch	Decode, R-read	ALU	PC update	D Memory	R-write	Total
Add	1	1	.9	-	-	.8	3.7
Load	1	1	.9	-	1	.8	4.7
Store	1	1	.9	-	1	-	3.9
beq	1	1	.9	.1	-	-	3.0

Let's assume setup + hold time for the state register = 0.1 ns

Single cycle design:

Clock cycle time = 4.7 + 0.1 = 4.8 ns

time/inst = 1 cycle/inst * 4.8 ns/cycle = 4.8 ns/inst

Multicycle design:

Clock cycle time = 1.0 + 0.1 = 1.1

time/inst = CPI * 1.1 ns/cycle (Depends on the types or mixture of instructions!)

A sample mixture for an application

	Cycles needed	Instruction frequency
Add	4	60%
Load	5	20%
Store	4	10%
beq	3	10%

What is CPI assuming this instruction mix???

4*0.6+5*0.2+4*0.1+3*0.1
= 2.4+1.0+0.4+0.3
= 4.1

Let's assume setup + hold time = 0.1 ns

Single cycle design:

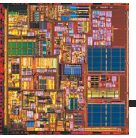
Clock cycle time = 4.7 + 0.1 = 4.8 ns

time/inst = 1 cycle/inst * 4.8 ns/cycle = 4.8 ns/inst

Multicycle design:

Clock cycle time = 1.0 + 0.1 = 1.1

time/inst = CPI * 1.1 ns/cycle = 4.1*1.1 = 4.5 ns/inst



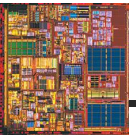
A More Extreme Example

- Calculation assumptions:
  - Most instructions take 10 nanoseconds (ns)
  - But multiply instruction takes 40ns
  - Multiplies are 10% of all instructions

How much faster is a multi-cycle data path over a single-cycle data path (for simplicity, assume setup=hold=0)?

Single-Cycle time/inst = 40ns

Multi-cycle with 10ns clock, time/inst = (1*0.9 + 4*0.1)*10
= 13ns



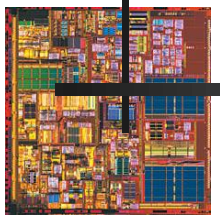
Another example

- Assumptions
  - 30% loads, 5ns
  - 10% stores, 5ns
  - 50% adds, 4ns
  - 10% multiplies, 20ns

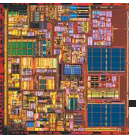
How much faster is a Multi-cycle Datapath over a single-cycle datapath (setup=hold=0)?

Single-cycle time/inst = 20ns

Multi-cycle with 5ns clock time/inst = (1*0.9 + 4*0.1) * 5
= 6.5 ns

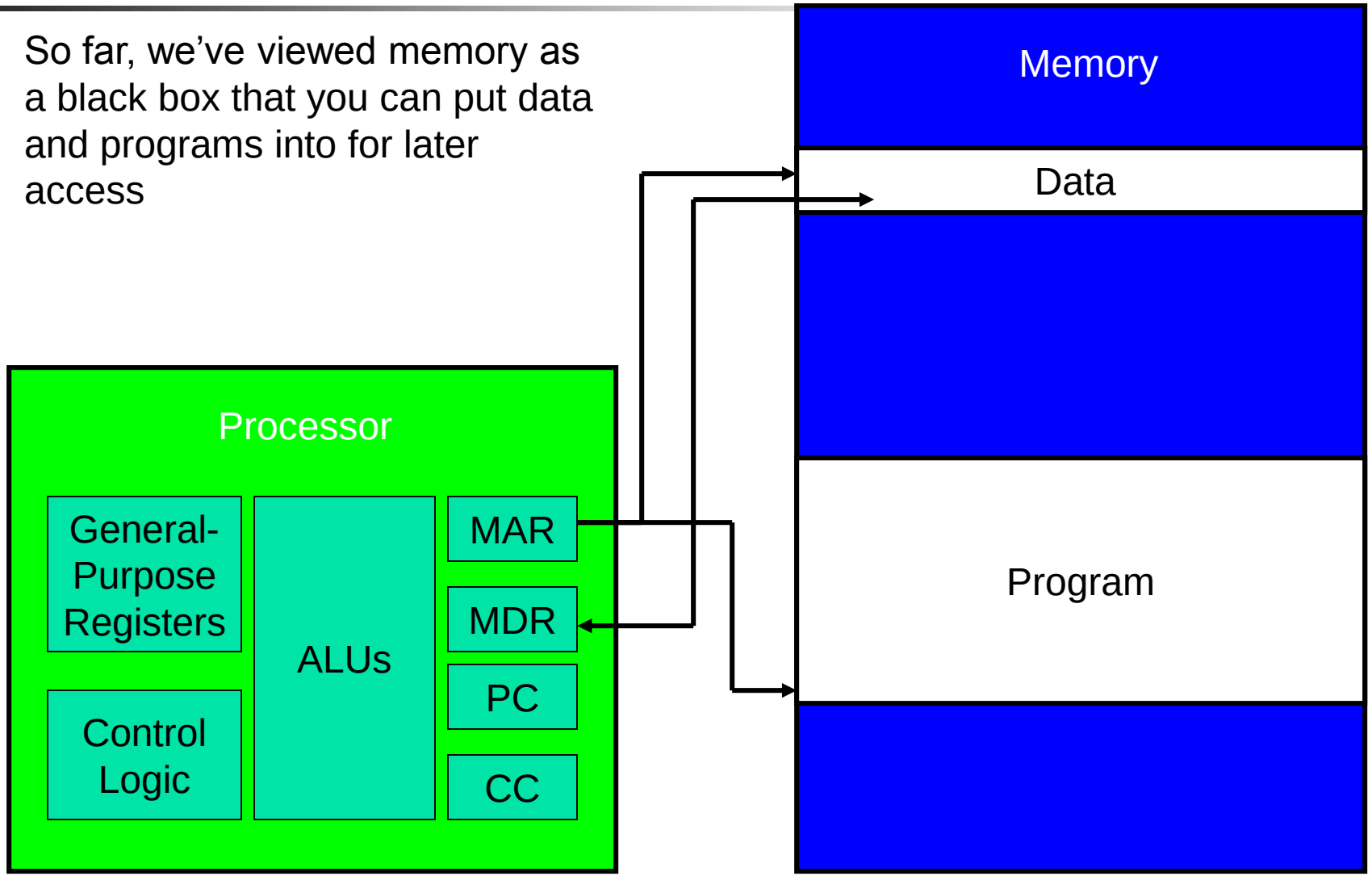


Memory Hierarchy

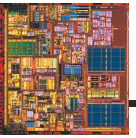


Physical Memory Systems

So far, we've viewed memory as a black box that you can put data and programs into for later access



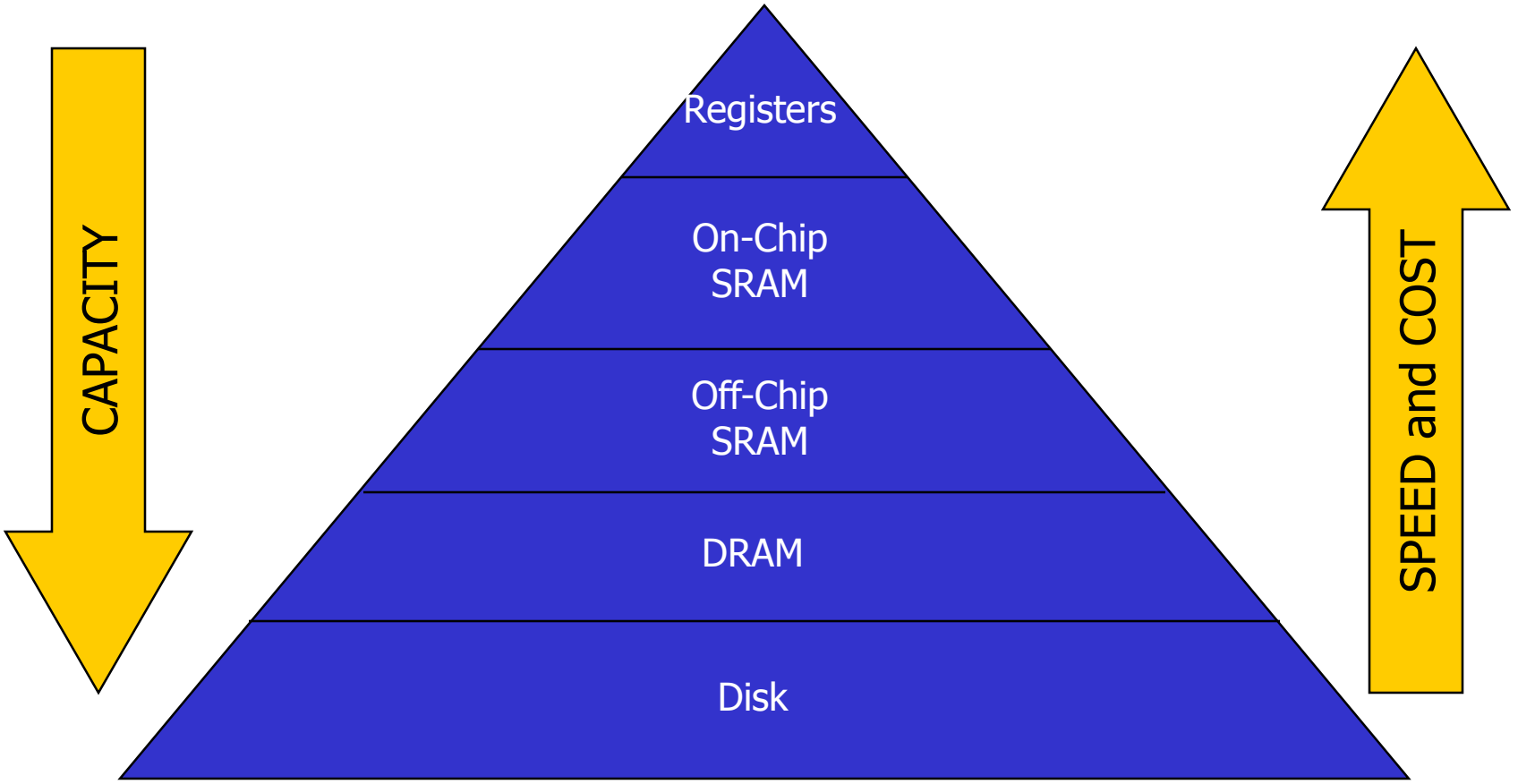
Types of Memories



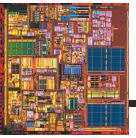
Type	Size	Latency	Cost/bit
Register	< 1KB	< 1ns	$$$$
On-chip SRAM	8KB-6MB	< 2ns	$$$
Off-chip SRAM	1Mb – 16Mb	< 10ns	$$
DRAM	64MB – 1TB	< 100ns	$
Disk (SSD, HD)	40GB – 1PB	< 20ms	< $1/GB

- SSD: $0.75 /GB, HD: $0.1-0.2/GB
- DRAM: $20-25/GB
- More on Bandwidth later

Memory Hierarchy

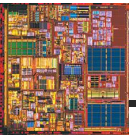


Why Does a Hierarchy Work?

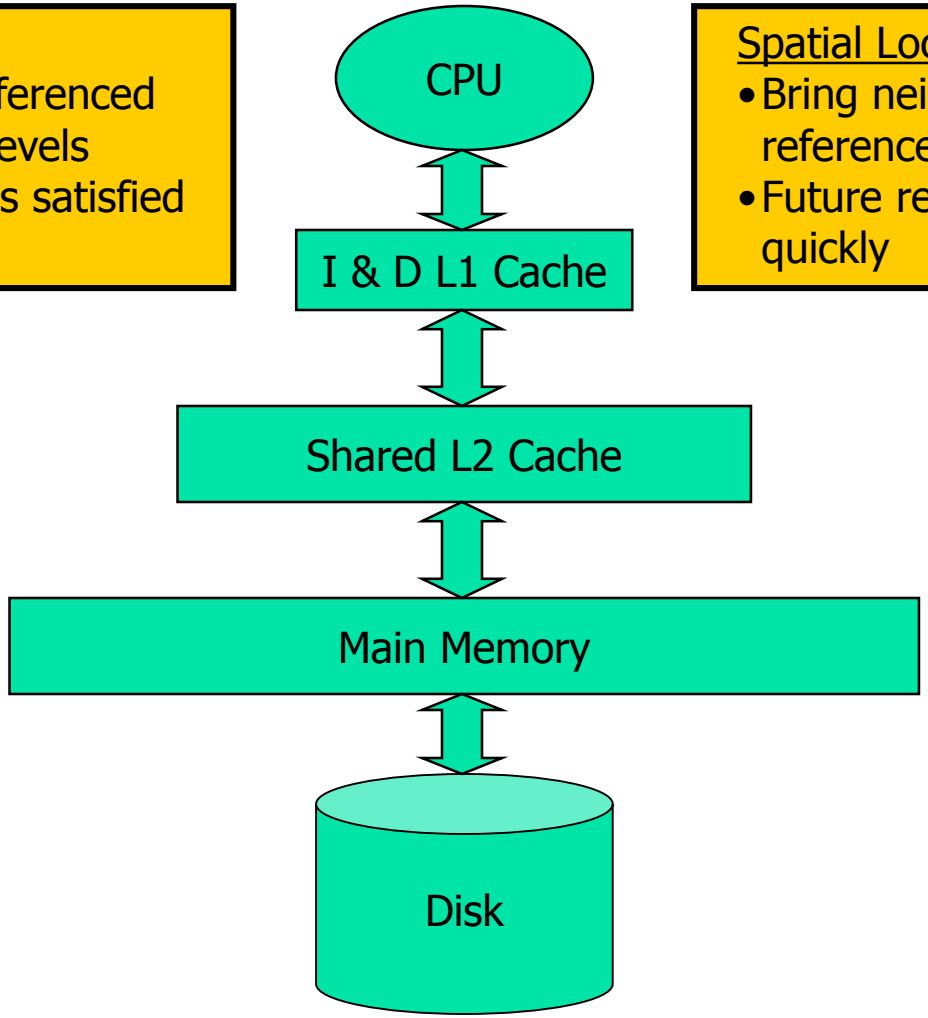


- Locality of reference
  - ***Temporal locality***
    - Reference same memory location many times (close together, in time)
  - ***Spatial locality***
    - Reference near neighbors around the same time

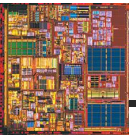
Memory Hierarchy



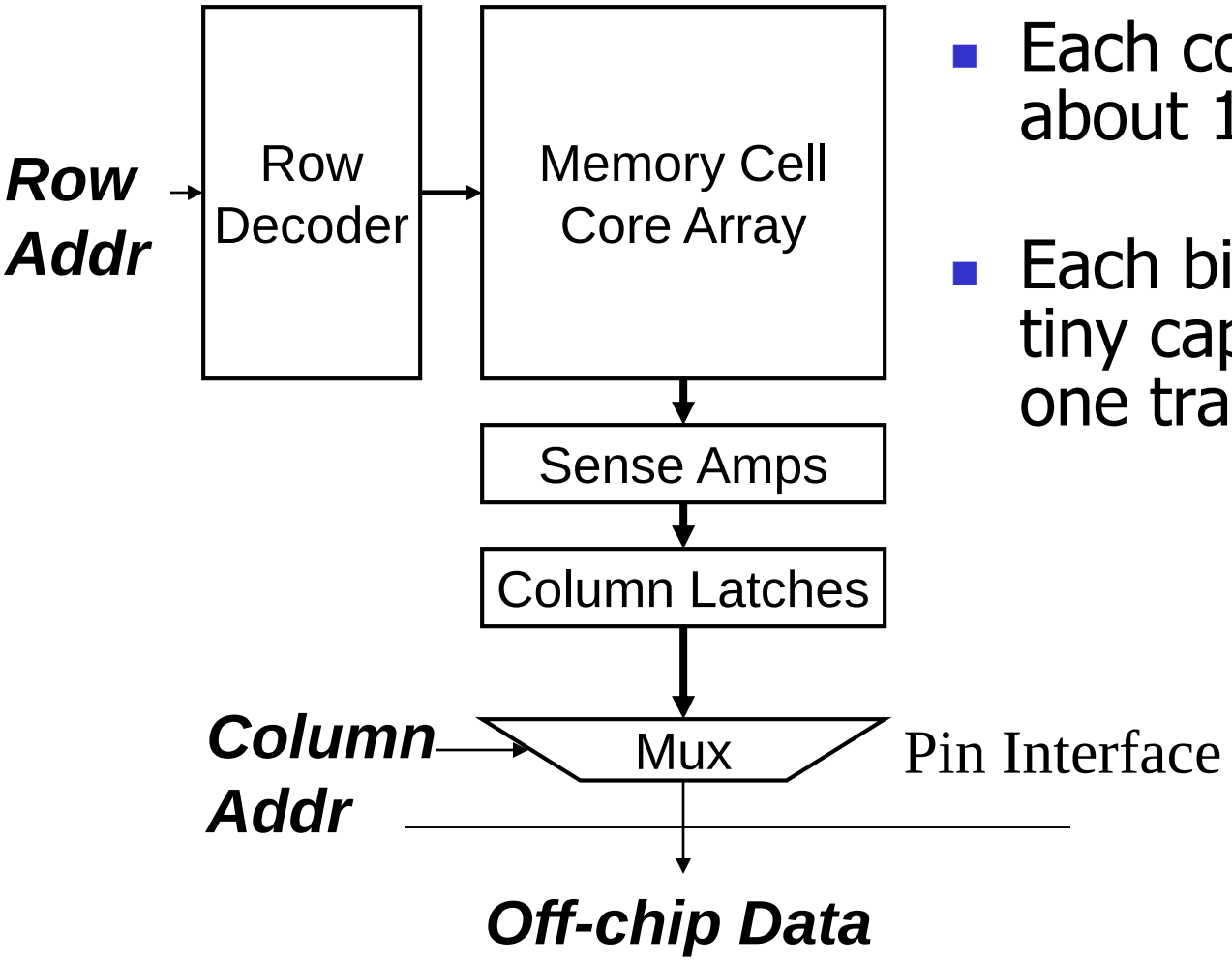
Temporal Locality
- Keep recently referenced items at higher levels
- Future references satisfied quickly



Spatial Locality
- Bring neighbors of recently referenced to higher levels
- Future references satisfied quickly

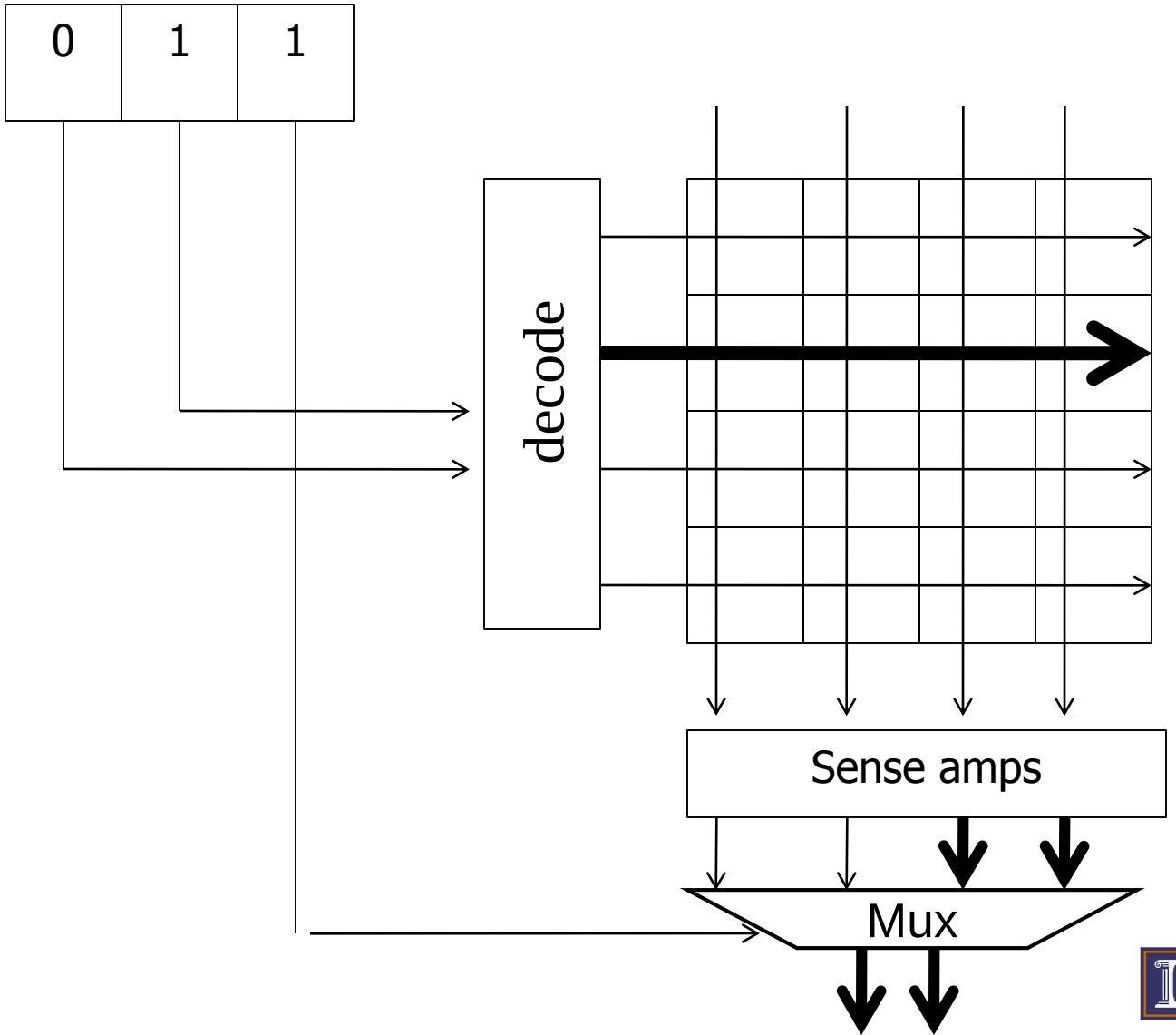


DRAM Bank Organization



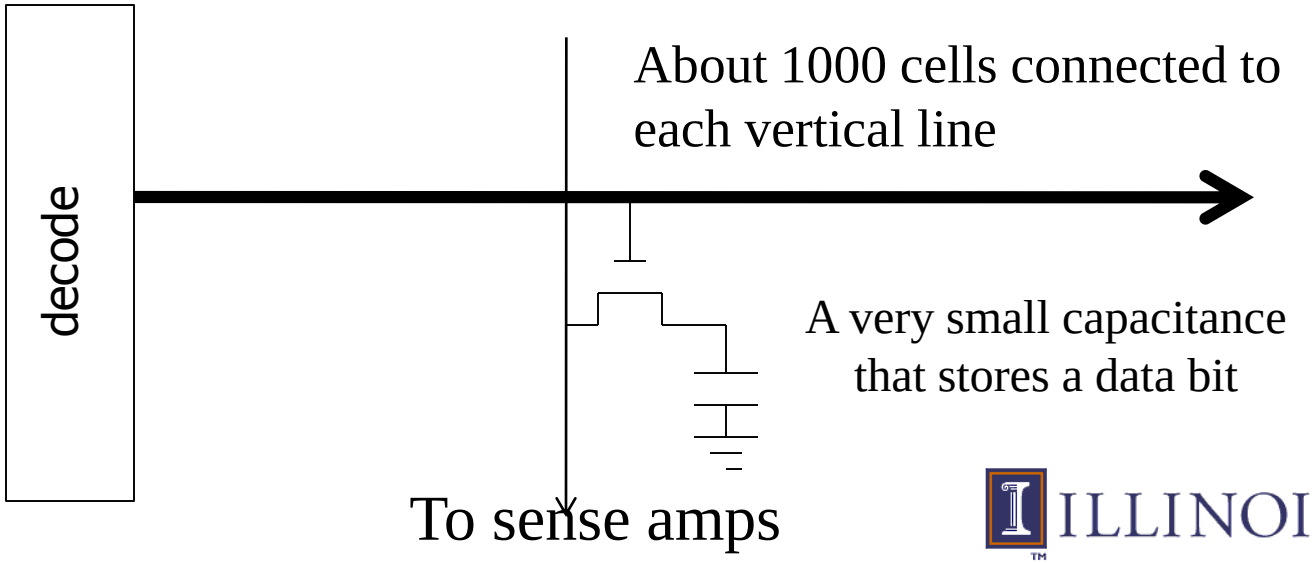
- Each core array has about 1M bits
- Each bit is stored in a tiny capacitor, made of one transistor

A very small (8x2 bit) DRAM Bank



DRAM core arrays are slow.

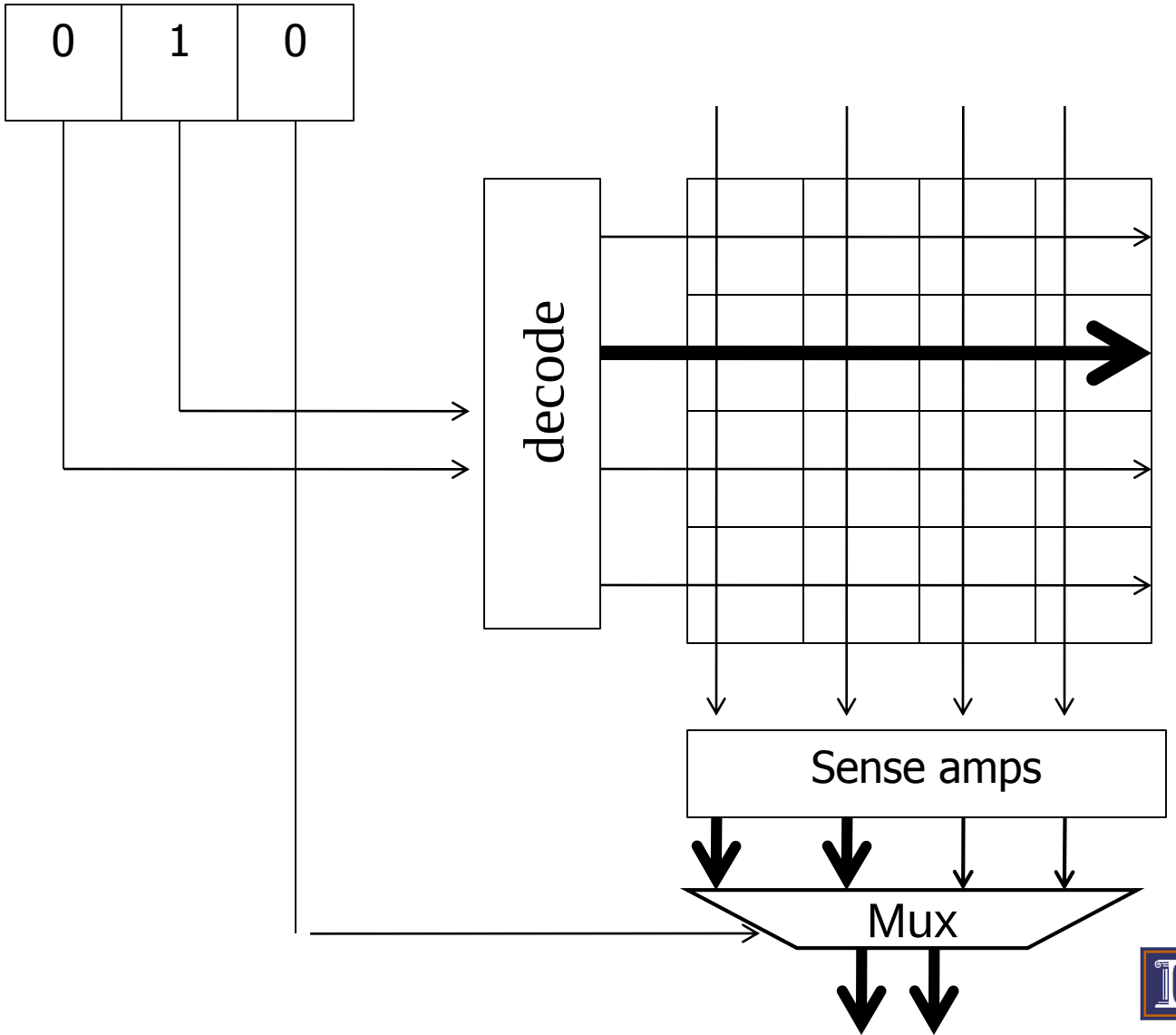
- Reading from a cell in the core array is a very slow process
  - DDR: Core speed = ½ interface speed
  - DDR2/GDDR3: Core speed = ¼ interface speed
  - DDR3/GDDR4: Core speed = ⅛ interface speed
  - … likely to be worse in the future



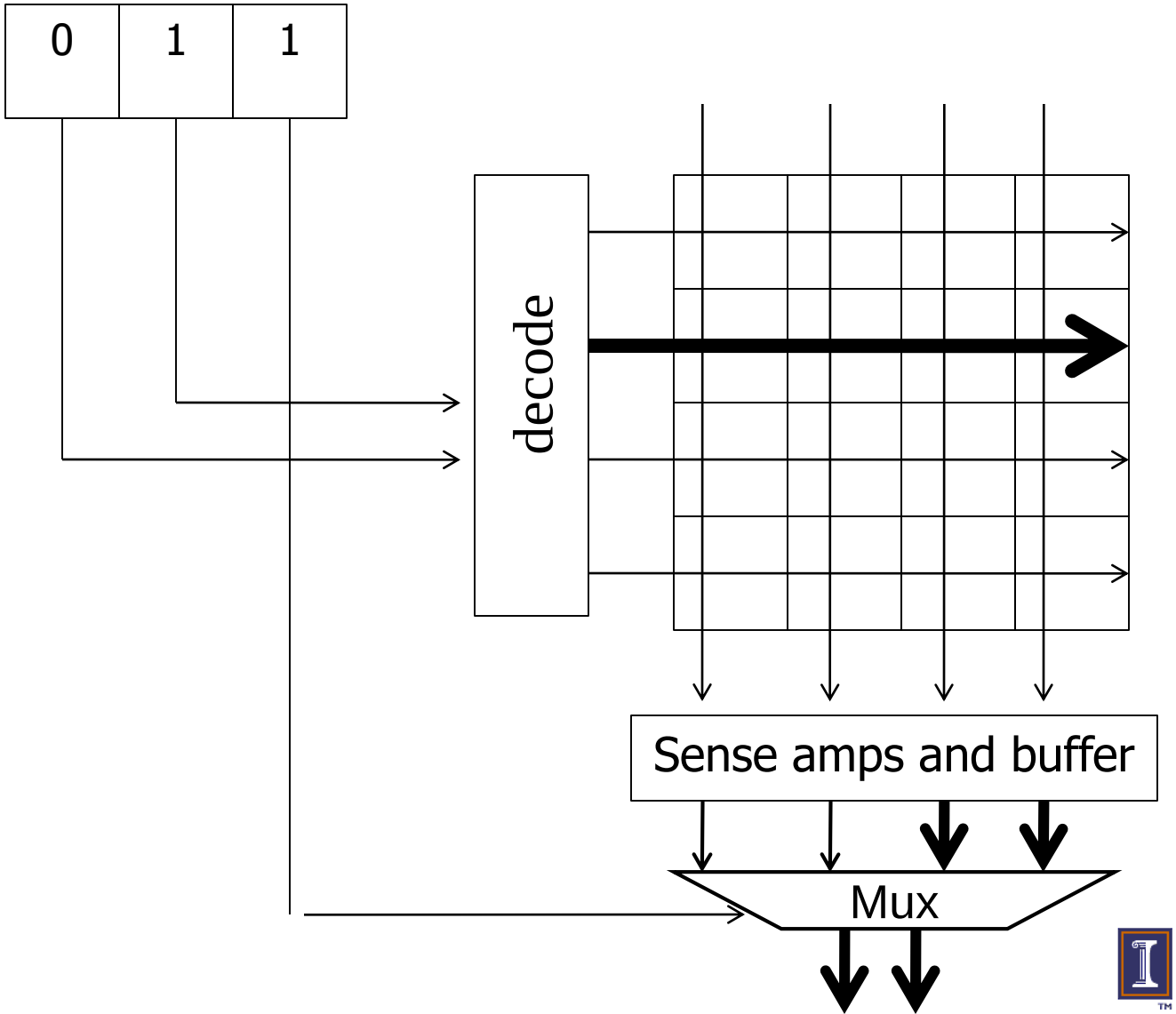
DRAM Bursting.

- For DDR{2,3} SDRAM cores clocked at 1/N speed of the interface:
  - Load (N × interface width) of DRAM bits from the same row at once to an internal buffer, then transfer in N steps at interface speed
  - DDR2/GDDR3: buffer width = 4X interface width

DRAM Bursting



DRAM Bursting



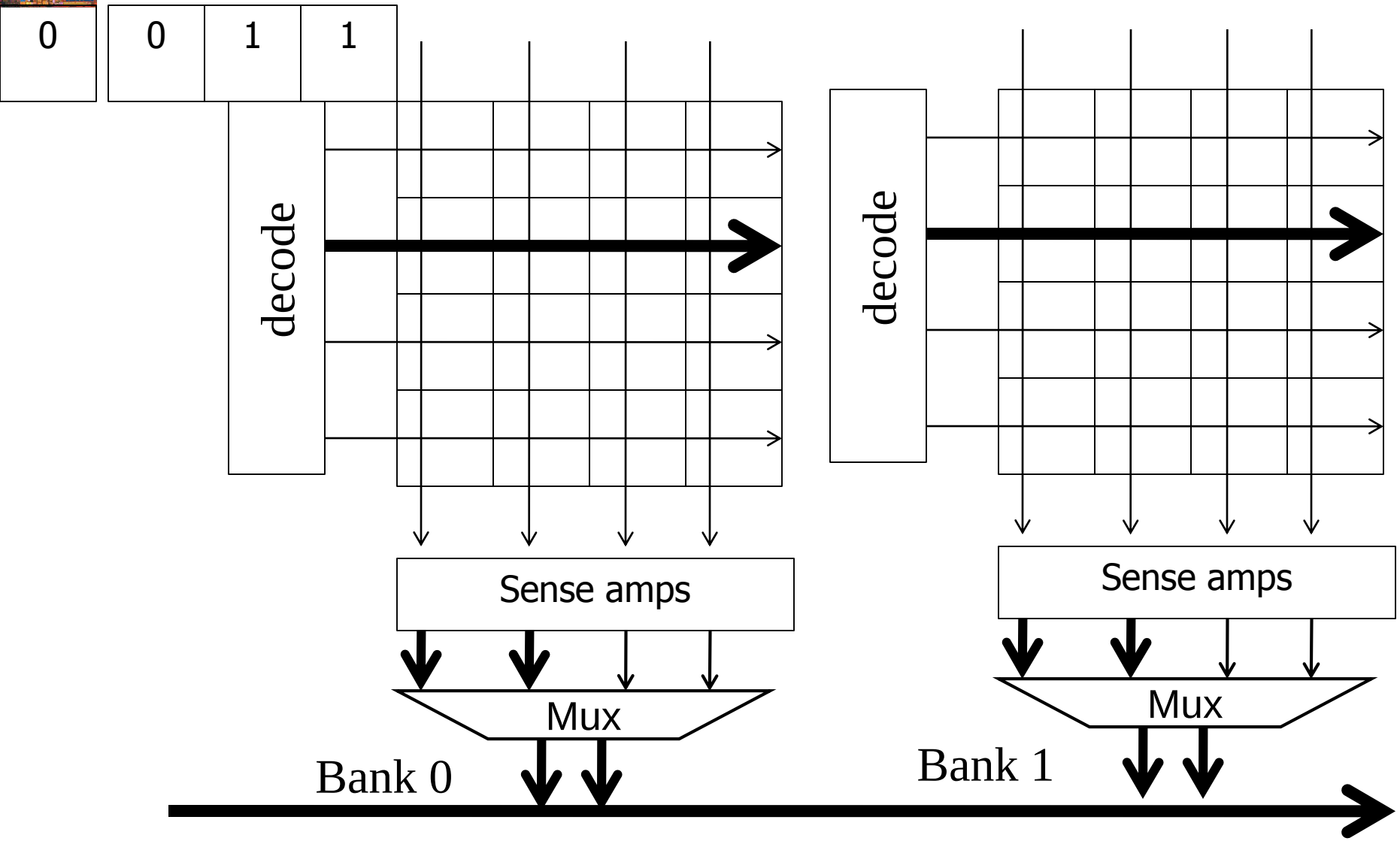
DRAM Bursting for the 8x2 Bank

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.

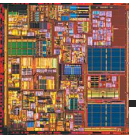
Multiple DRAM Banks



DRAM Bursting for the 8x2 Bank

Single-Bank burst timing, dead time on interface

Multi-Bank burst timing, reduced dead time



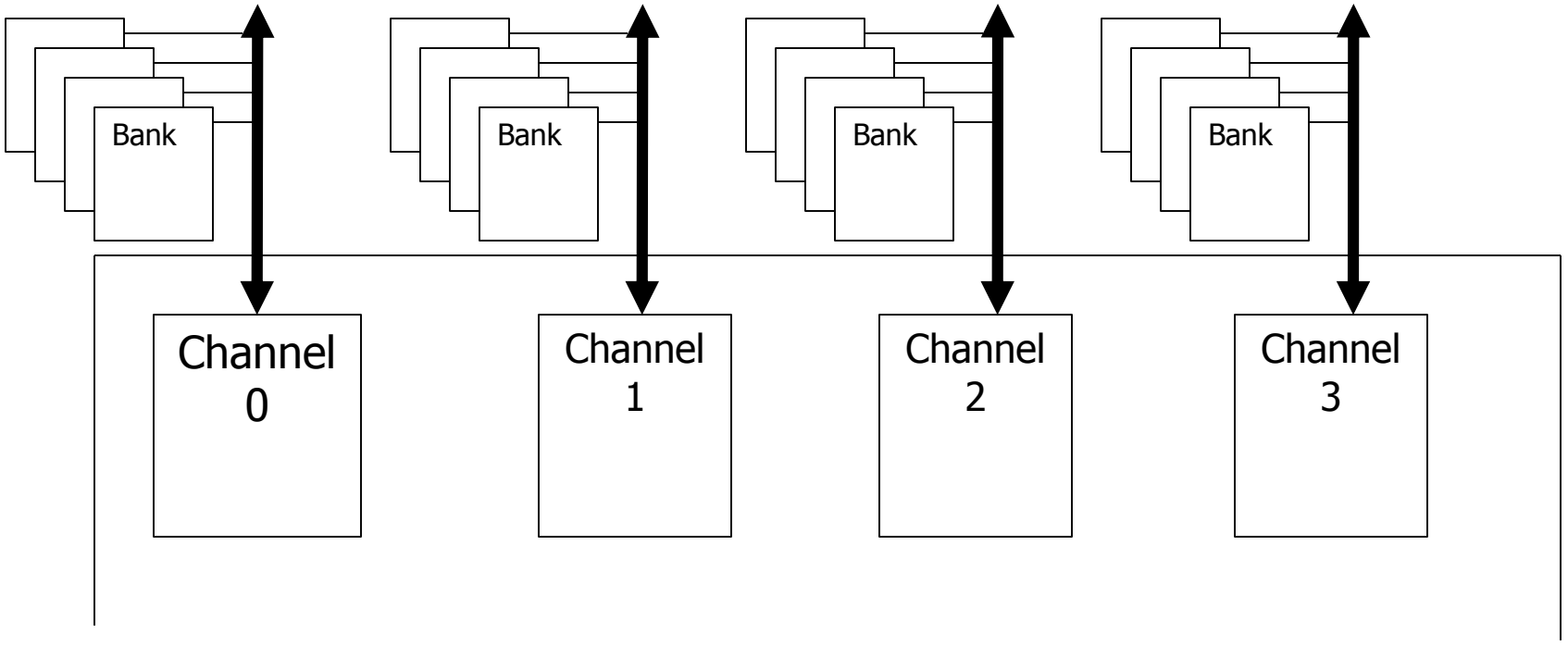
FA GPU off-chip DRAM subsystem

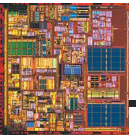
- nVidia GTX280 GPU:
  - Peak global memory bandwidth = 141.7GB/s

- Global memory (GDDR3) interface @ 1.1GHz
  - (Core speed @ 276Mhz)
  - For a typical 64-bit interface, we can sustain only about 17.6 GB/s (Recall DDR - 2 transfers per clock)
  - We need a lot more bandwith (141.7 GB/s) – thus 8 memory channels

Multiple Memory Channels

- Divide the memory address space into N parts
  - N is number of memory channels
  - Assign each portion to a channel





READ CHAPTERS 3 AND 4
FINISH MP 1