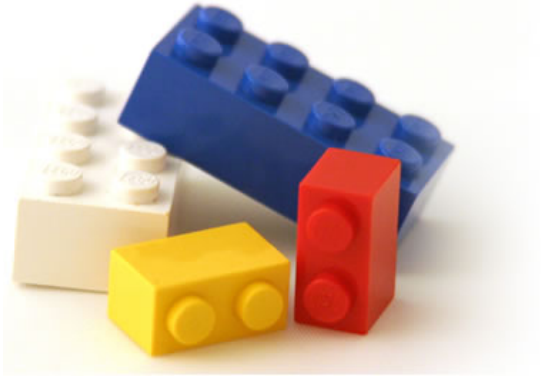
EECS 4340: Computer Hardware Design
Unit 3: Design Building Blocks



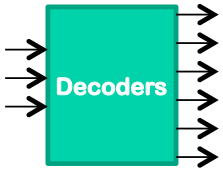
Prof. Simha Sethumadhavan

DRAM Illustrations from Memory Systems by Jacob, Ng, Wang

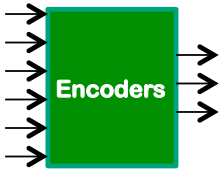
Columbia University

Hardware Building Blocks I: Logic

- Math units
  - Simple fixed point, floating point arithmetic units
  - Complex trigonometric or exponentiation units



- Decoders
  - N input wires
  - M output wires, M > N



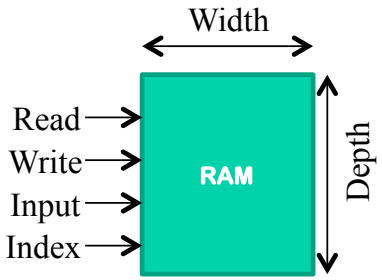
- Encoders
  - N input wires
  - M output wires, M < N, *e.g.*, first one detector

Designware Components

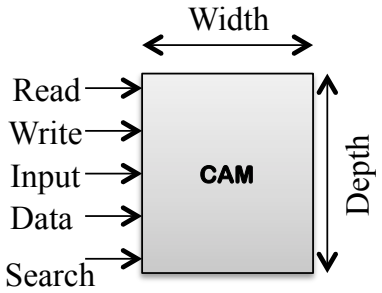
- dwbb_quickref.pdf (on courseworks)
  - Do not distribute

Hardware Building Blocks II: Memory

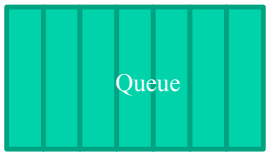
- Simple Register: Flop/latch groups



- Random Access Memory
  - Data = RAM[Index]
  - Parameters: Depth, Width, Portage

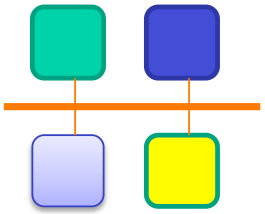


- Content Addressable Memory
  - Index = CAM [Data]
  - Parameters: Depth, Width, Portage
  - Operations supported (Partial Search, R/W)

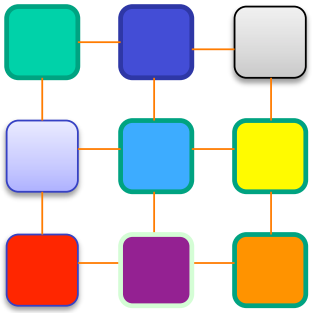


- Queues
  - Data = FIFO[oldest] or Data = LIFO[youngest]

Building Blocks III : Communication

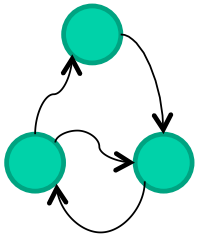


- Broadcast (Busses)
  - Message is sent to all clients
  - Only one poster at any time
  - Does not scale to large number of nodes



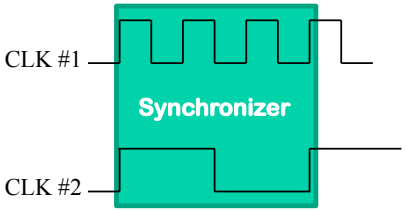
- Point-to-Point (Networks)
  - Message is sent to one node
  - Higher aggregate bandwidth
  - *De rigueur* in hardware design

Building Blocks IV: Controllers



Pipeline Controllers

- Pipeline controllers
  - Manages flow of data through pipeline regs
  - Synthesized as random logic



- Synchronizer
  - Chips have multiple clock domains
  - Handle data xfer between clock domains

- Memory, I/O controllers
  - Fairly complex blocks, *e.g.*, DRAM controller
  - Often procured as Soft-IP blocks

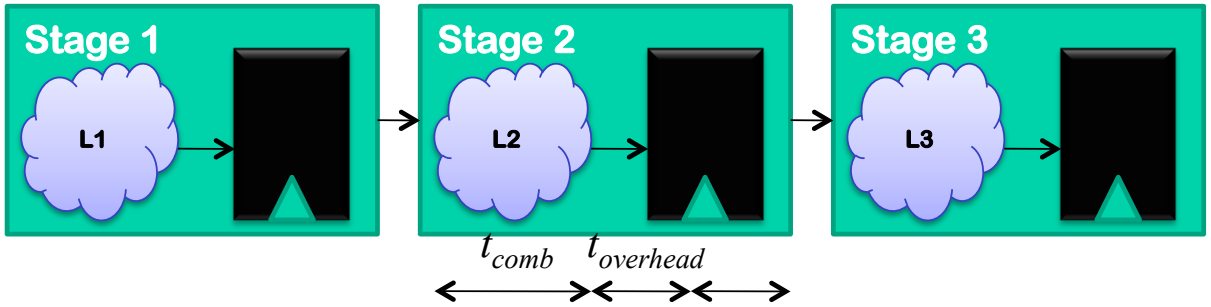
Rest of this Unit

More depth:

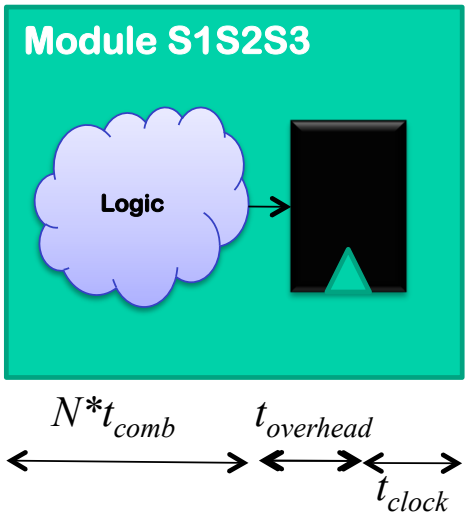
- **Pipelining/Pipeline Controllers**
- **SRAM**
- **DRAM**
- **Network on Chip**

Pipelining Basics

Clock Frequency = $1/(t_{comb} + t_{clock} + t_{overhead})$



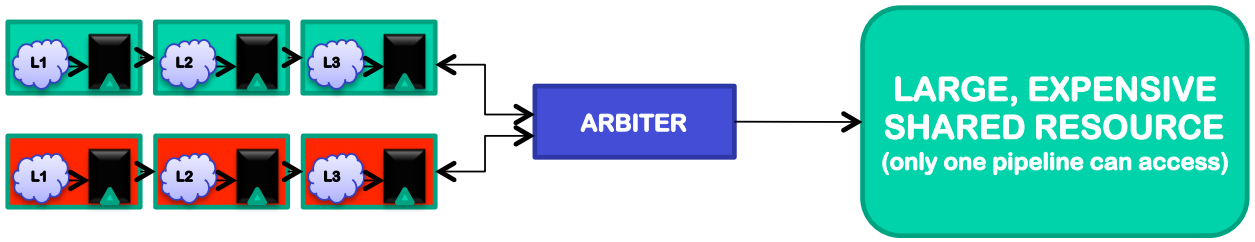
Unpipleined



Clock Frequency = $1/(N*t_{comb} + t_{clock} + t_{overhead})$

Pipelining Design Choices

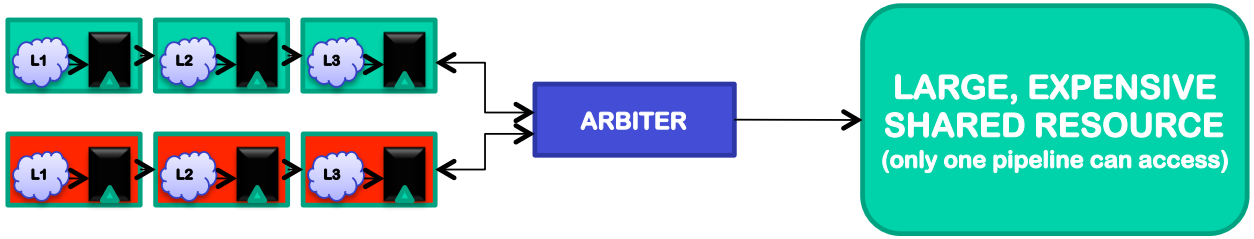
Consider:



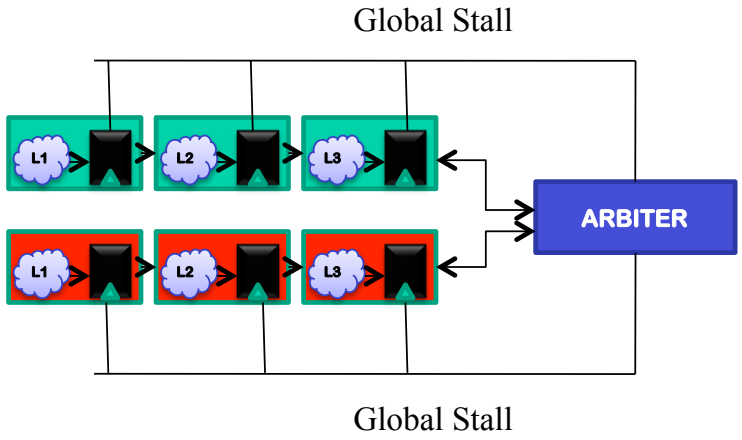
How should you orchestrate access to the shared resource?

Pipelining Design Choices

Consider:



Strategy1: GLOBAL STALL

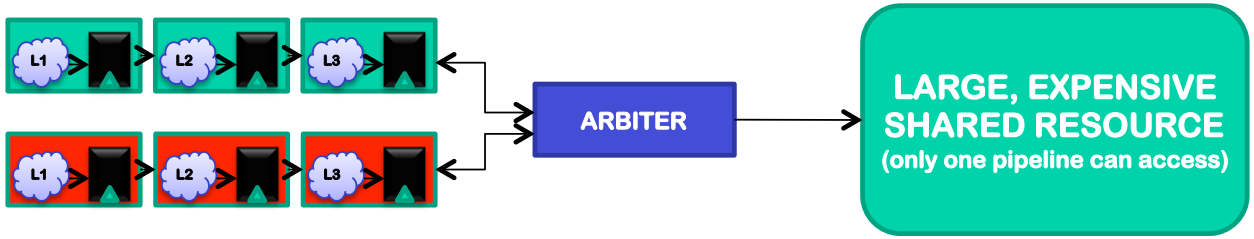


Pros: Intellectually Easy!

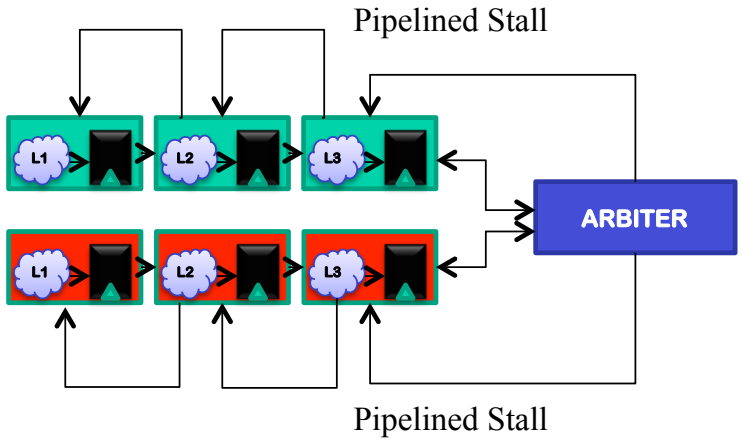
Cons: SLOW!
- Delay α Length of wire
- α Fanout

Pipelining Design Choices

Consider:



Strategy 2: PIPELINED STALL



Pros: Fast!

Cons: Area
- One cycle delay for stalls => one additional latch per stage!
- More complex arbitration

Space-Time Diagram

Stage	Cycle N	Cycle N + 1	Cycle N + 3	Cycle N + 4
K	A (Stall due to ARB)			
K – 1	B			
K – 2	C			
K – 3	D			
K – 4	E			

Space-Time Diagram

Stage	Cycle N	Cycle N + 1	Cycle N + 3	Cycle N + 4
K	A (Stall due to ARB)			
K – 1	B	Stall reached K-1		
K – 2	C			
K – 3	D			
K – 4	E			

Space-Time Diagram

Stage	Cycle N	Cycle N + 1	Cycle N + 3	Cycle N + 4
K	A (Stall due to ARB)	AB		
K – 1		Stall reached K-1; C		
K – 2		D		
K – 3		E		
K – 4				

Space-Time Diagram

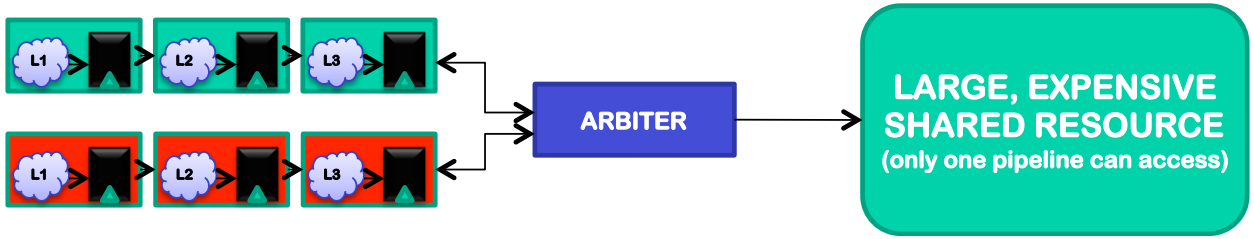
Stage	Cycle N	Cycle N + 1	Cycle N + 3	Cycle N + 4
K	A (Stall due to ARB)	AB (Stall)	AB (Stall)	B (unstall)
K – 1		Stall reached K-1;	CD (Stall)	CD (Stall)
K – 2			Stall reached K-2. E	
K – 3				
K – 4				

Space-Time Diagram

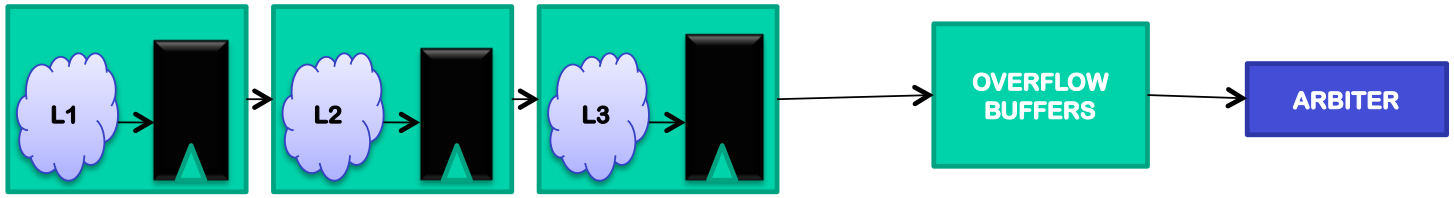
Stage	Cycle N	Cycle N + 1	Cycle N + 3	Cycle N + 4	Cycle N+5
K	A (Stall due to ARB)	AB (Stall)	AB (Stall)	B (unstall)	C
K – 1		Stall reached K-1;	CD (Stall)	D(Stall)	(unstall)
K – 2			Stall reached K-2. E		
K – 3					
K – 4					

Pipelining Design Choices

Consider:



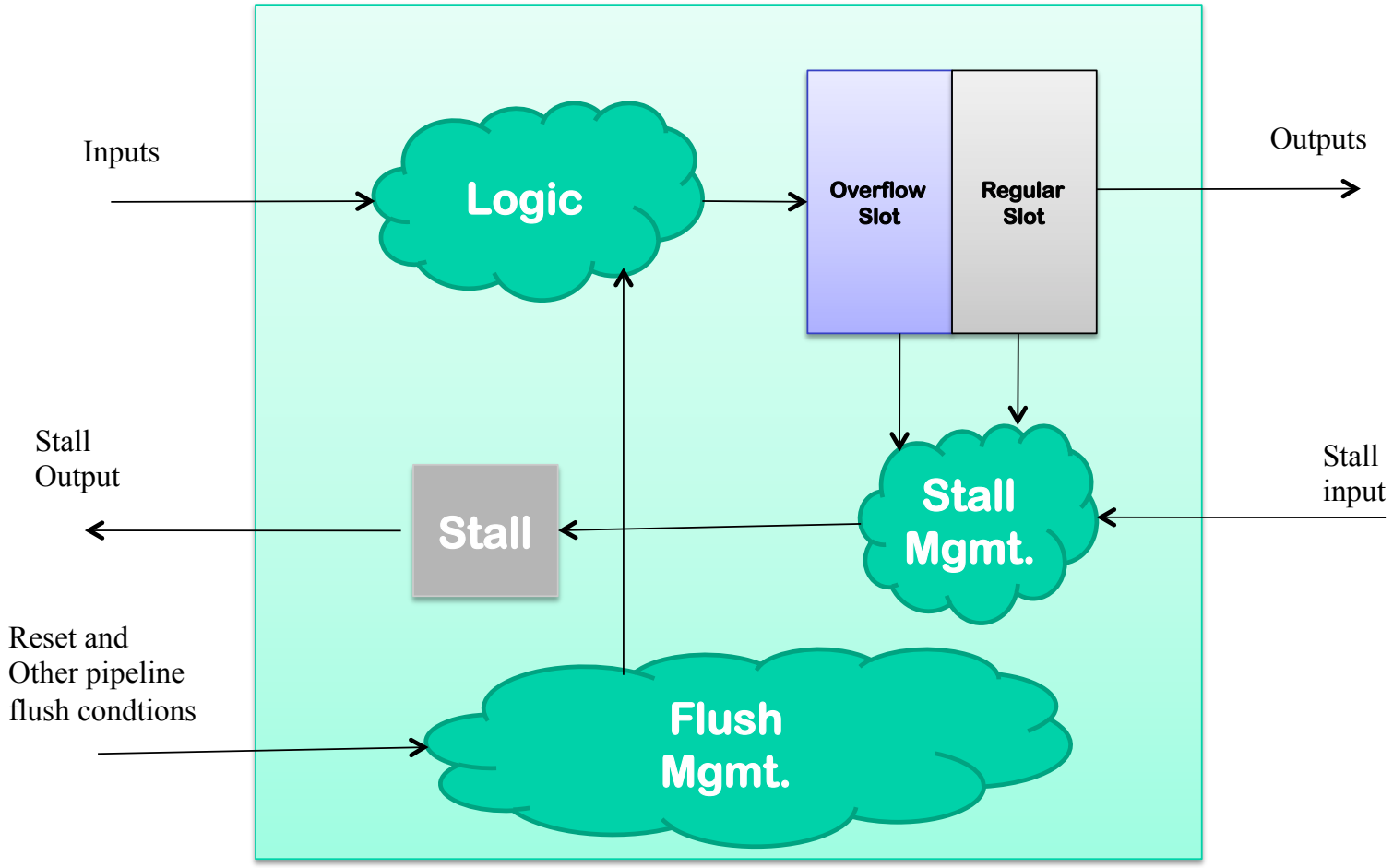
Strategy 3: OVERFLOW BUFFERS AT THE END



Pros: Very fast, useful when pipelines are self-throttling

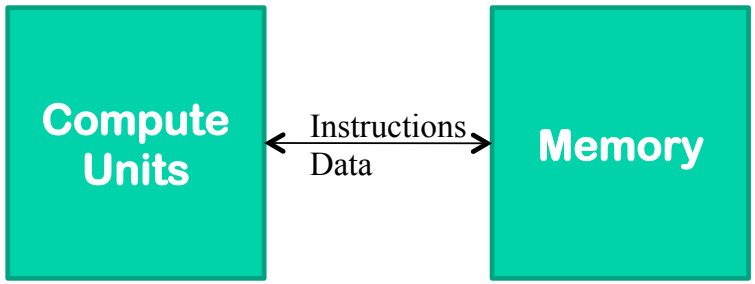
Cons:
- Area!
- Complex arbitration logic at the arb.

Generic Pipeline Module



Introduction to Memories

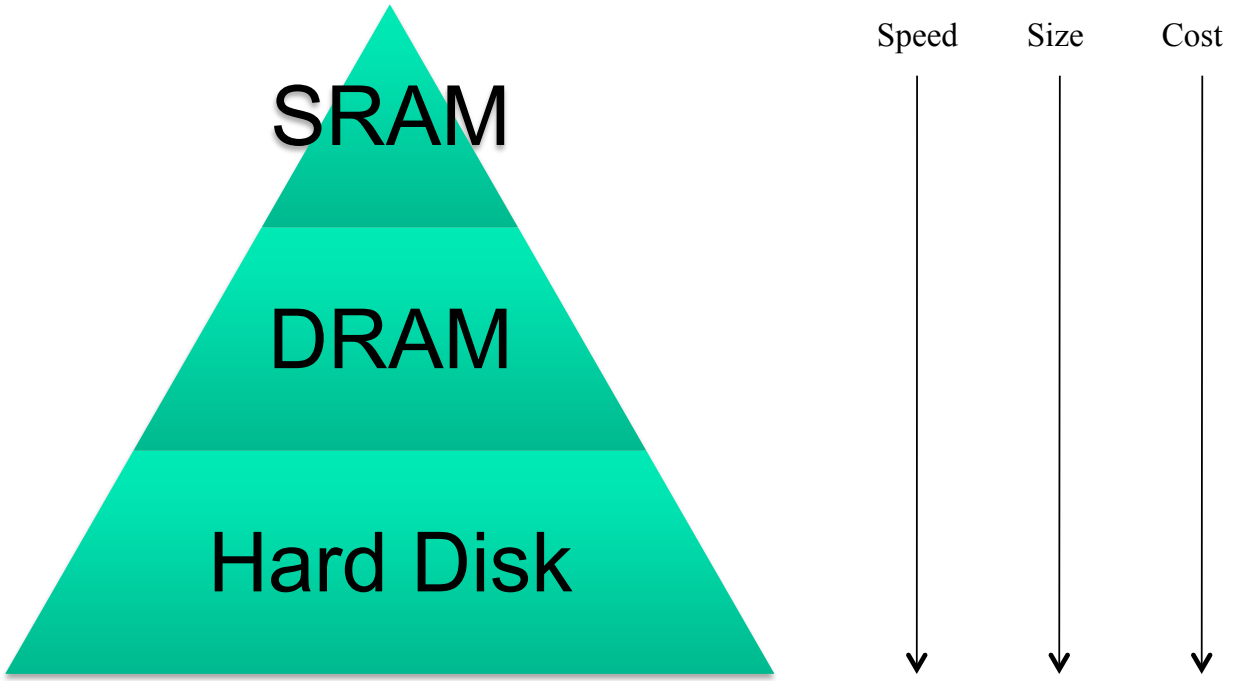
- Von-neumann model
  - "Instructions are data"



- Memories are primary determiners of performance
  - Bandwidth
  - Latency

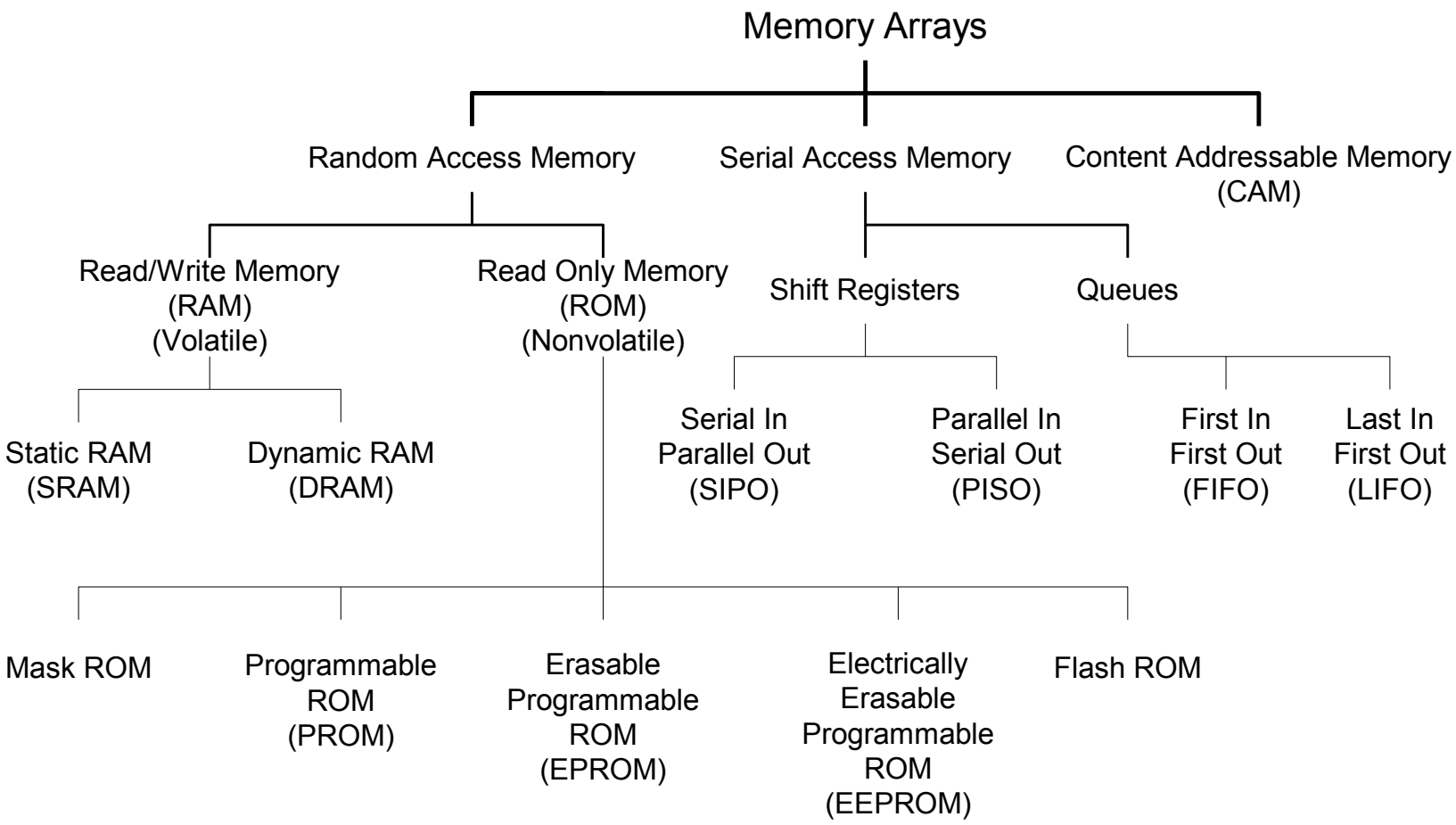
Memory Hierarchies

- Invented to ameliorate some memory system issues

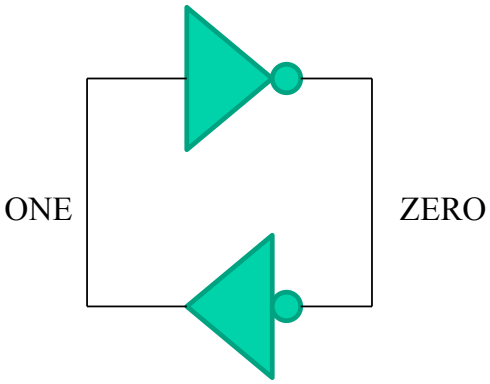


- A fantastic example of how architectures can address technology limitations

Types of Memories

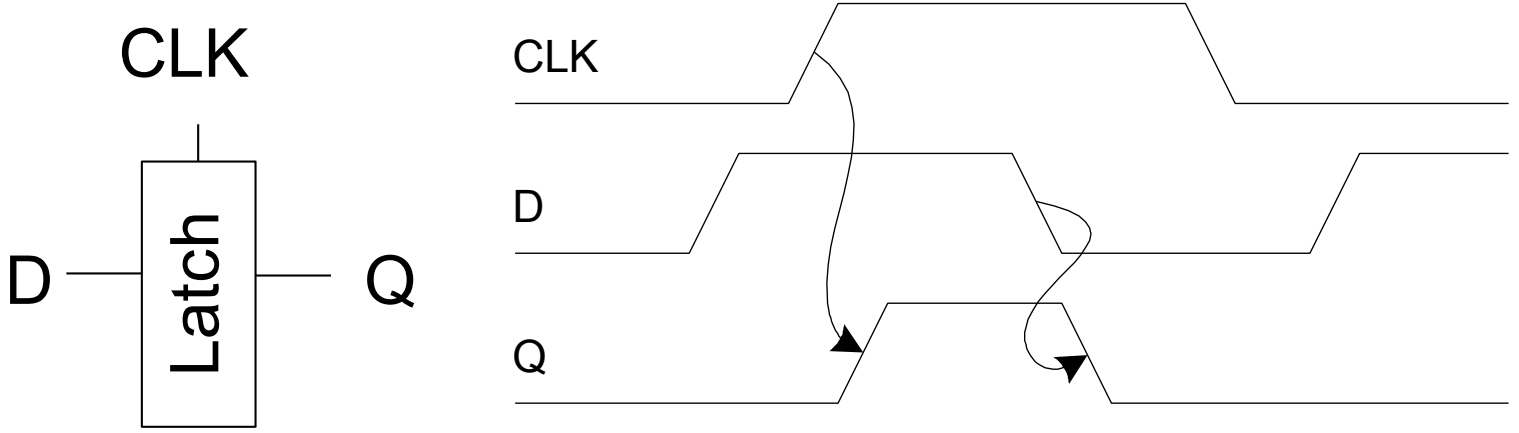


Memory



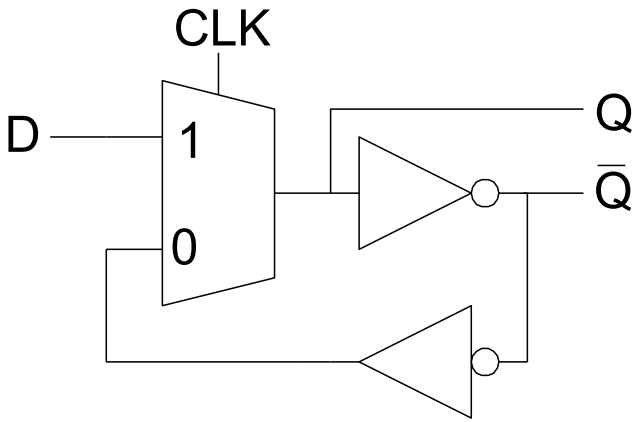
D Latch

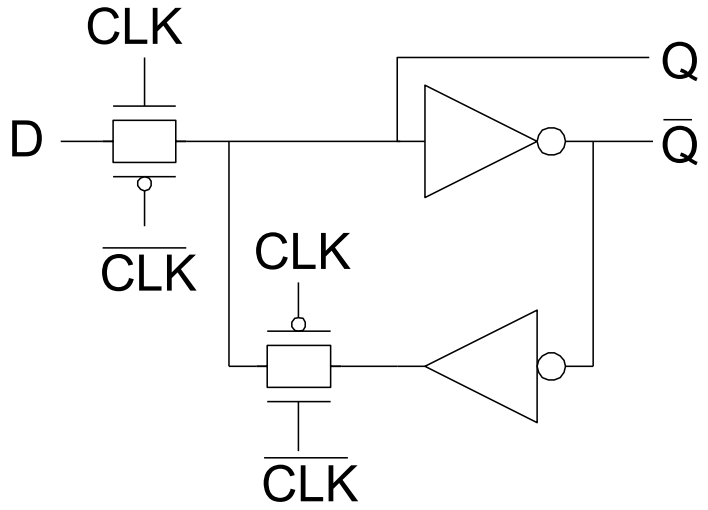
- When CLK = 1, latch is transparent
  - D flows through to Q like a buffer
- When CLK = 0, the latch is opaque
  - Q holds its old value independent of D
- a.k.a. transparent latch or level-sensitive latch



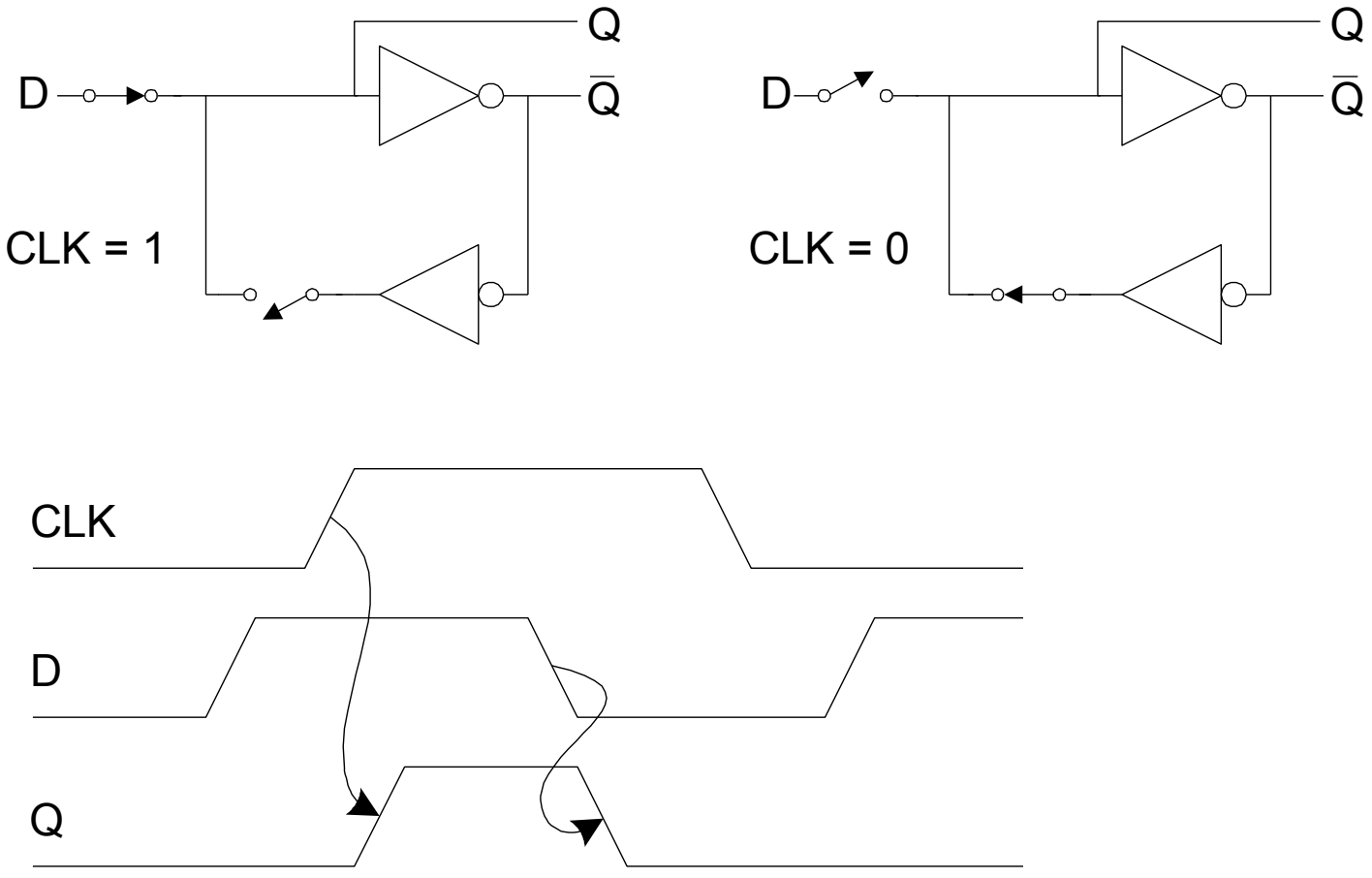
D Latch Design

- **Multiplexer chooses D or old Q**



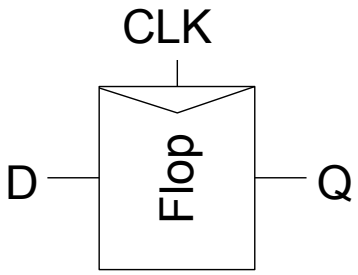


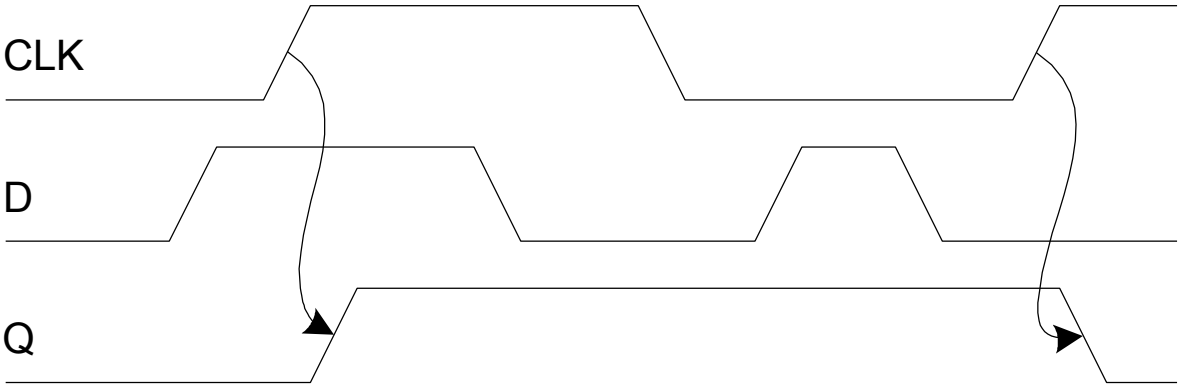
D Latch Operation



D Flip-flop

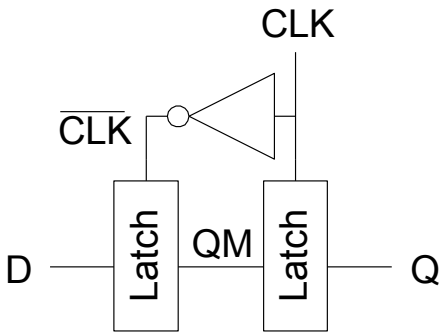
- When CLK rises, D is copied to Q
- At all other times, Q holds its value
- a.k.a. *positive edge-triggered flip-flop*, *master-slave flip-flop*

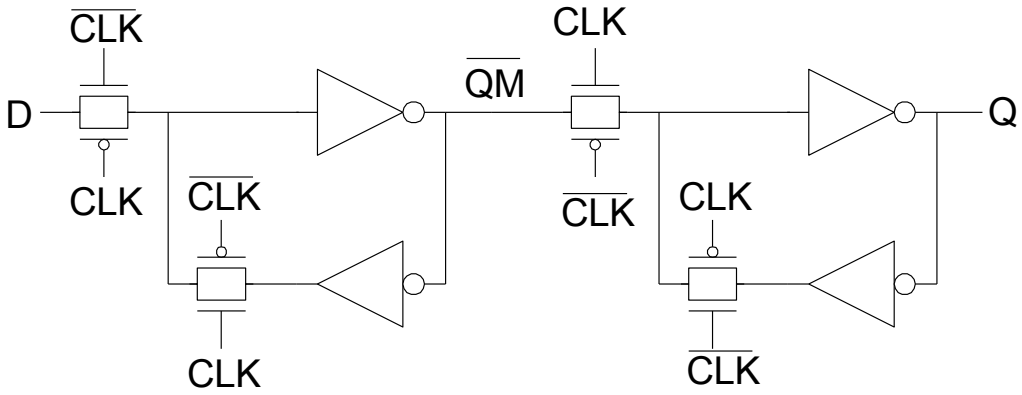




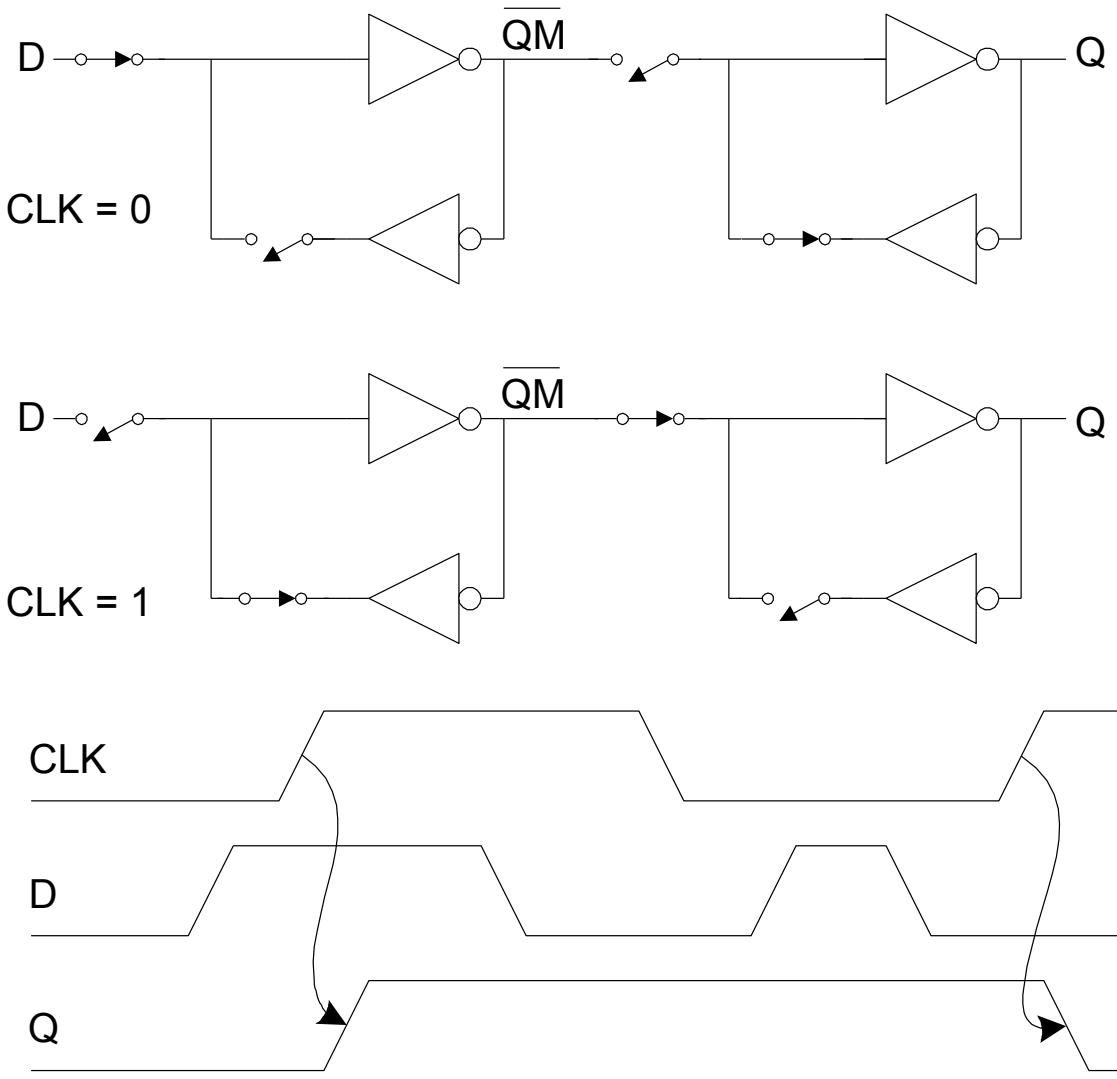
D Flip-flop Design

- **Built from master and slave D latches**





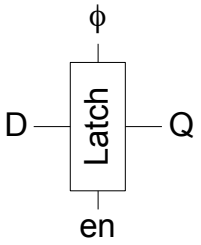
D Flip-flop Operation



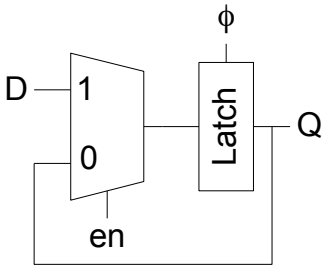
Enable

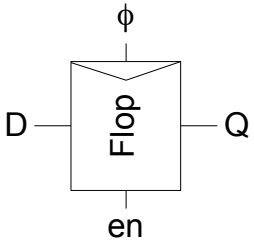
- **Enable: ignore clock when en = 0**
  - **Mux: increase latch D-Q delay**

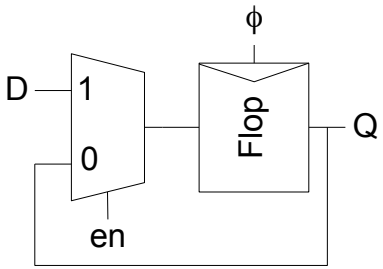
Symbol



Multiplexer Design

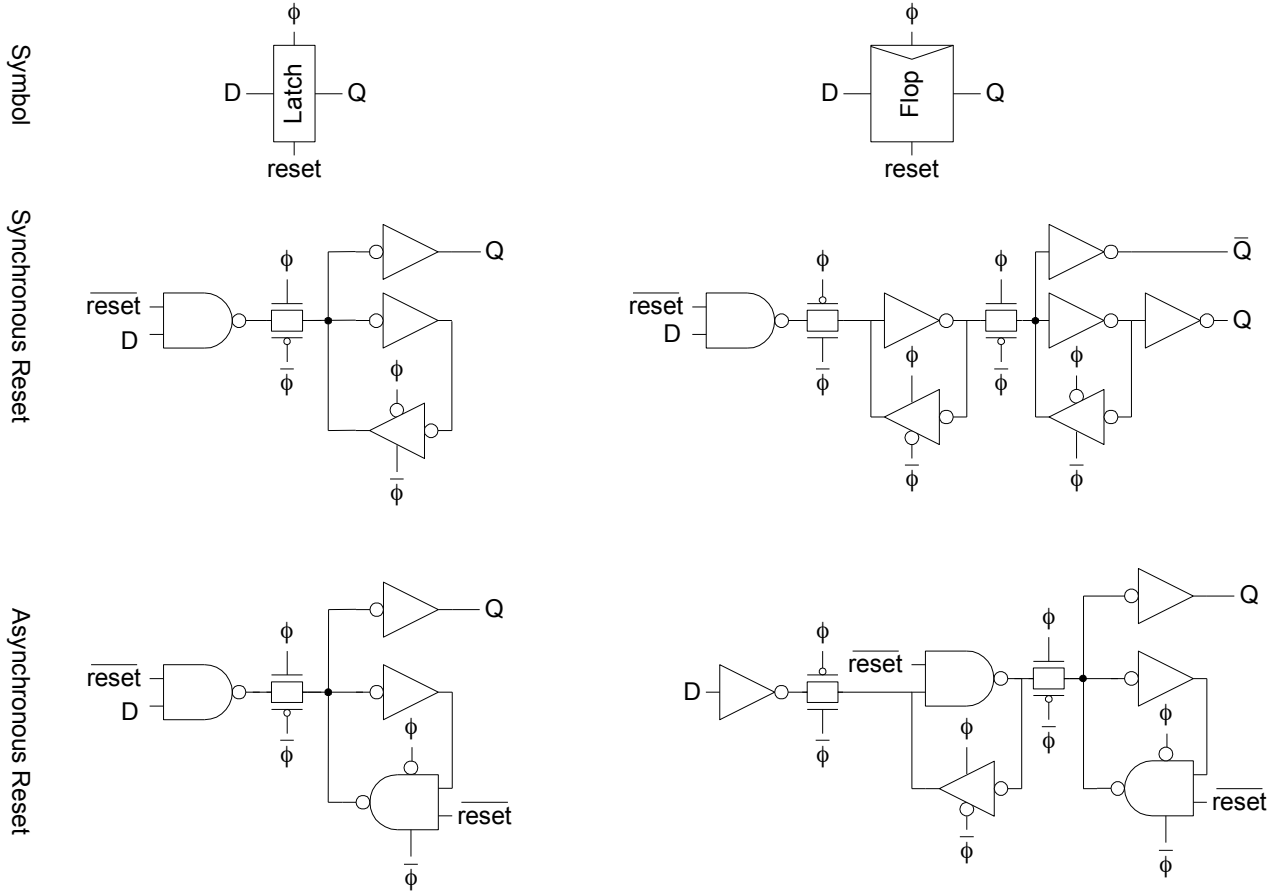






Reset

- Force output low when reset asserted
- Synchronous vs. asynchronous



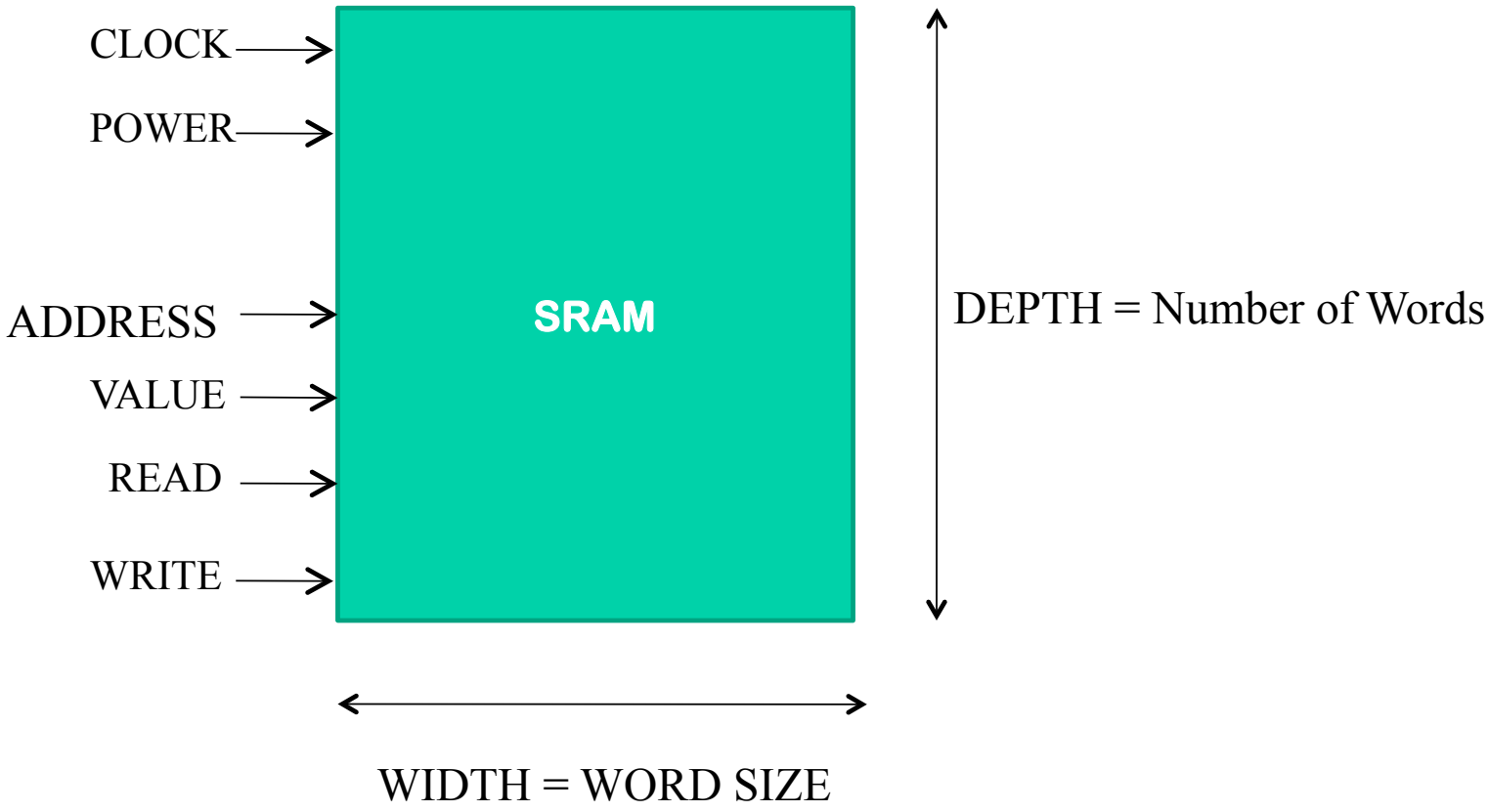
Building Small Memories

- We have done this already
- Latch vs Flip Flop tradeoffs
  - Count number of transistors
  - Examine this tradeoff in your next homework

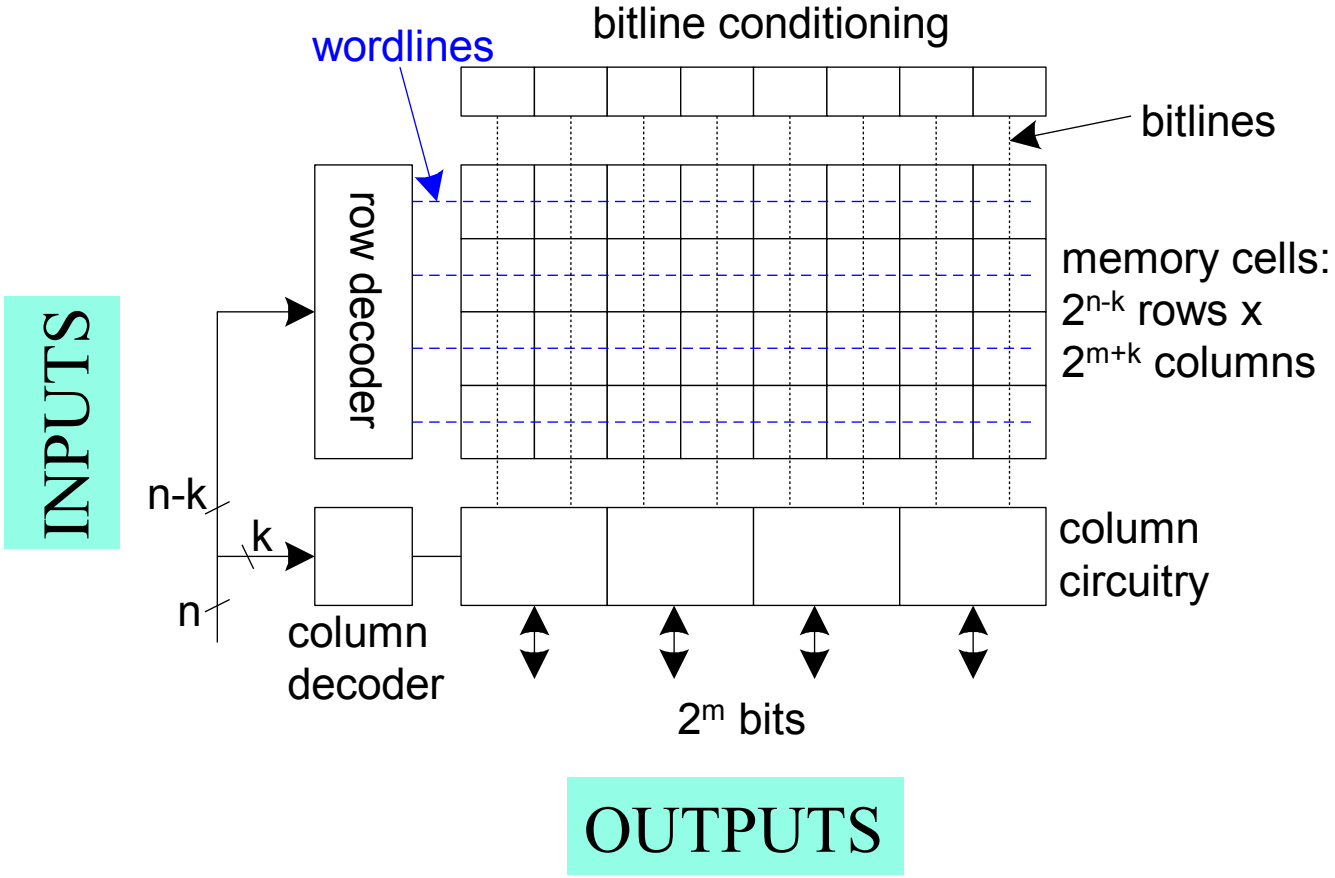
SRAMs

- Operations
  - Reading/Writing
- Interfacing SRAMs into your design
  - Synthesis

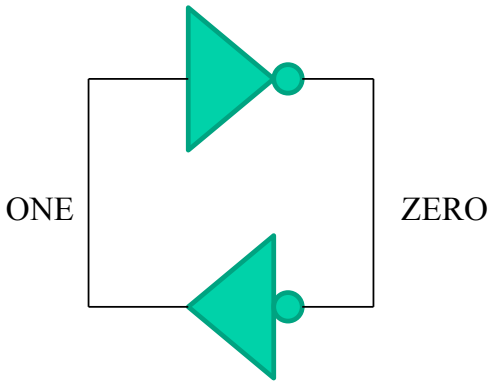
SRAM Abstract View



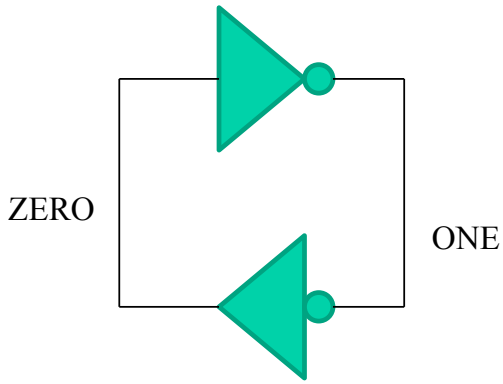
SRAM Implementation



Bistable CMOS

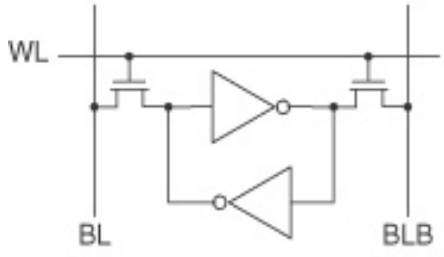


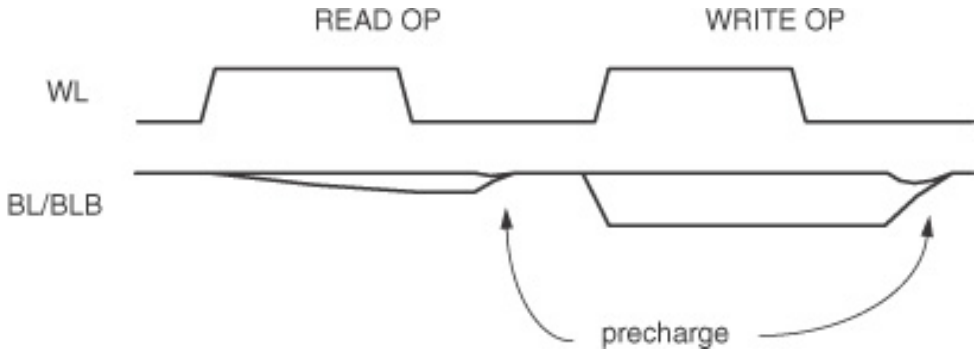
1



0

The “6T” - SRAM CELL

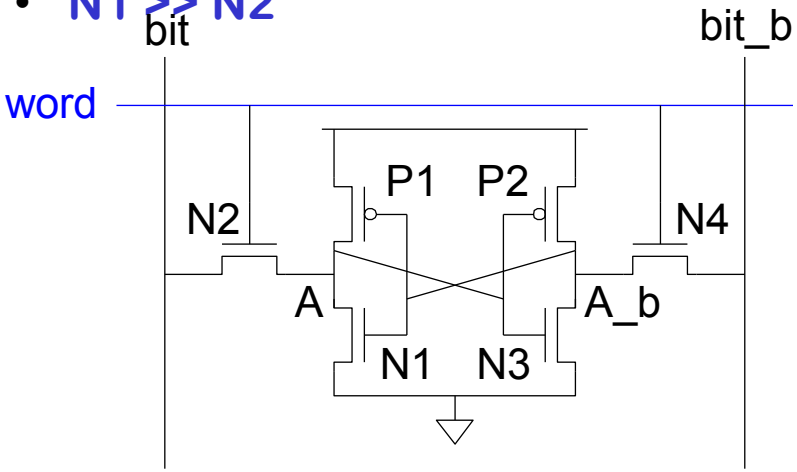


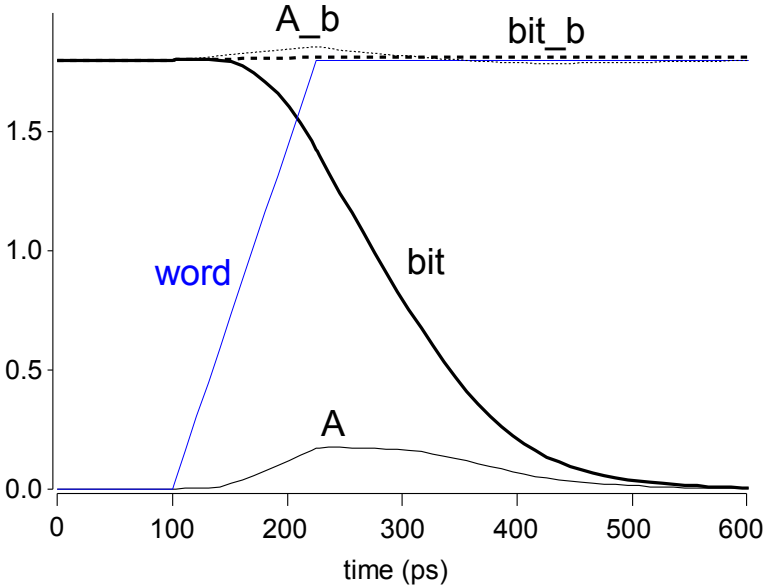


- 6T SRAM Cell
  - Used in most commercial chips
  - Data stored in cross-coupled inverters
- Read:
  - Precharge BL, BLB (Bit LINE, Bit LINE BAR)
  - Raise WL (Wordline)
- Write:
  - Drive data onto BIT, BLB
  - Raise WL

SRAM Read

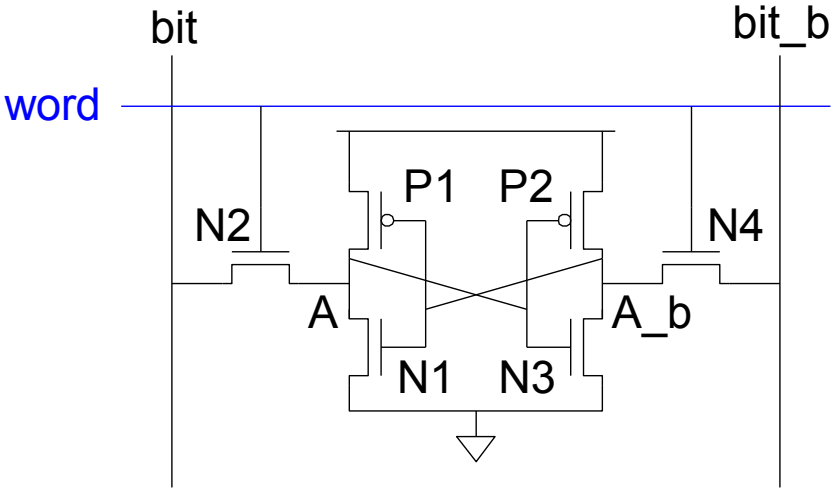
- Precharge both bitlines high
- Then turn on wordline
- One of the two bitlines will be pulled down by the cell
- Ex: A = 0, A_b = 1
  - bit discharges, bit_b stays high
  - But A bumps up slightly
- *Read stability*
  - A must not flip
  - N1 >> N2

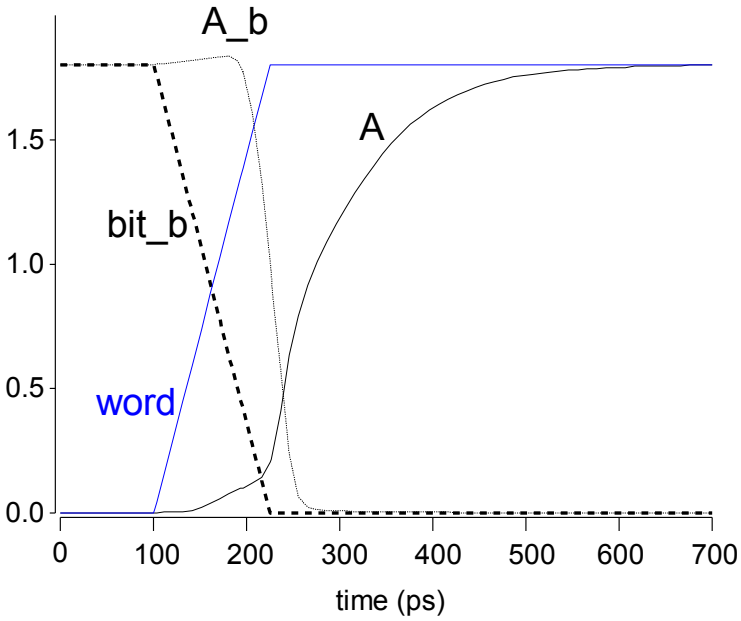




SRAM Write

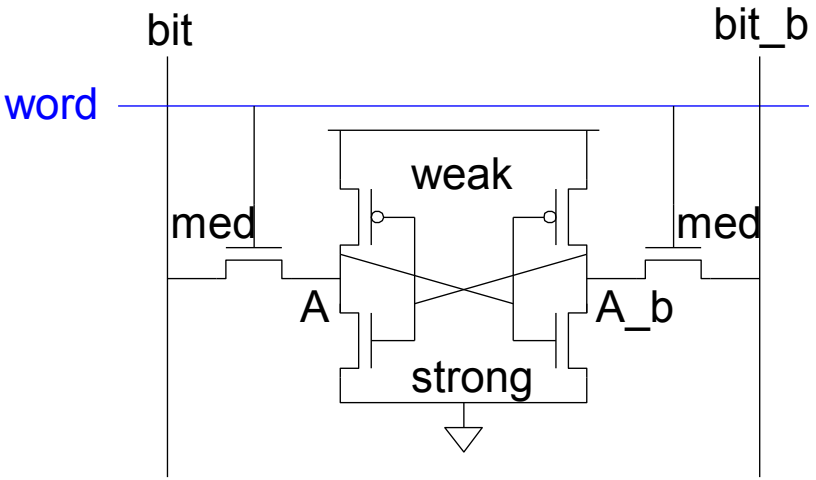
- Drive one bitline high, the other low
- Then turn on wordline
- Bitlines overpower cell with new value
- Ex: A = 0, A_b = 1, bit = 1, bit_b = 0
  - Force A_b low, then A rises high
- *Writability*
  - Must overpower feedback inverter
  - N2 >> P1





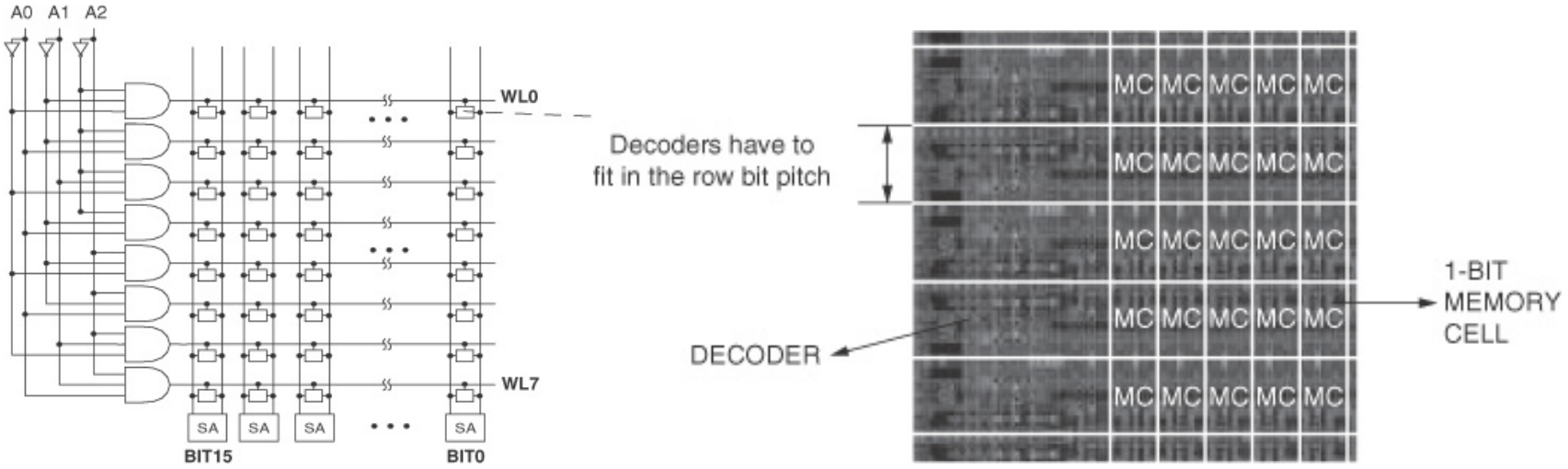
SRAM Sizing

- High bitlines must not overpower inverters during reads
- But low bitlines must write new value into cell



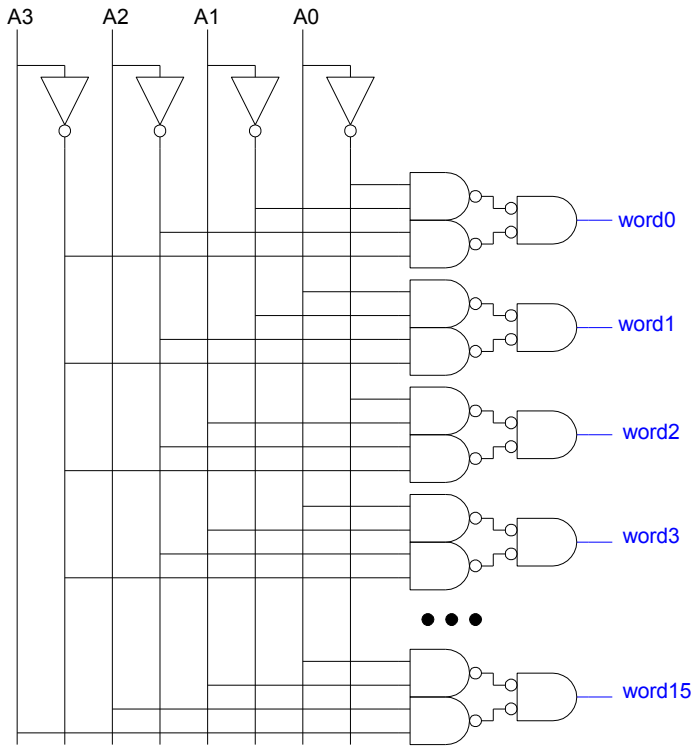
Decoders

- n:2^n decoder consists of 2^n n-input AND gates
  - One needed for each row of memory
- Naïve decoding requires large fan-in AND gates
  - Also gates must be pitch matched with SRAM logic



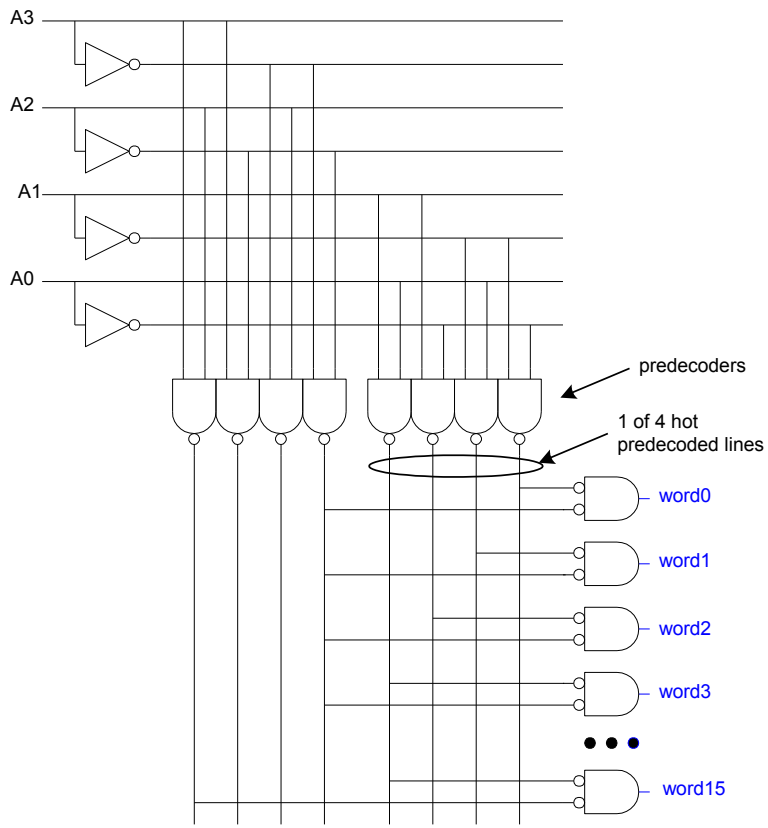
Large Decoders

- For n > 4, NAND gates become slow
  - Break large gates into multiple smaller gates



Predecoding

- Many of these gates are redundant
  - Factor out common gates into predecoder
  - Saves area



Timing

- Read timing
  - Clock to address delay
  - Row decode time
  - Row address driver time
  - Bitline sense time
  - Setup time to capture data on to latch
- Write timing
  - Usually faster because bit lines are actively driven

Drawbacks of Monolithic SRAMS

- As number of SRAM cells increases, the number of transistors increase, increasing the total capacitance and therefore the resulting delay and power increase

- Increasing number of cells results in physical lengthening of the SRAM array, increasing the wordline wire length and the wiring capacitance

- More power in the bitlines is wasted each cycle because more and more columns are activated by a single word line even though a subset of these columns are activated.

Banking SRAMs

- Split the large array into small subarrays or banks
- Tradeoffs
  - Additional area overhead for sensing and peripheral circuits
  - Better speed
- Divided wordline Architecture
  - Predecoding
  - Global wordline decoding
  - Local wordline decoding
- Additional optimization – divided bitlines

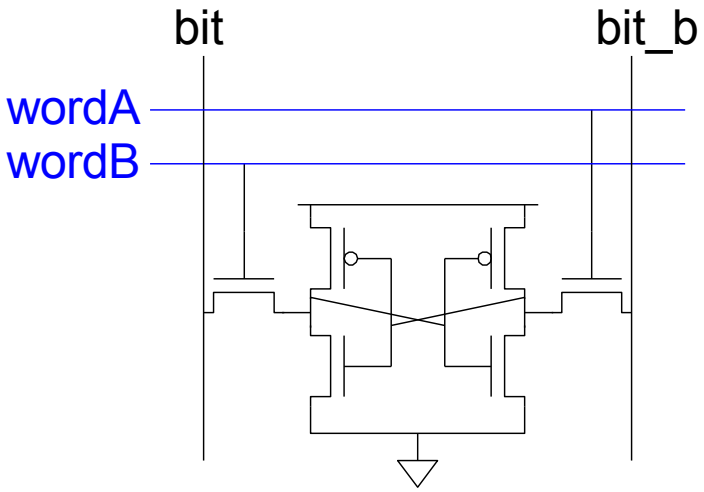
Multiple Ports

- We have considered single-ported SRAM
  - One read or one write on each cycle

- *Multiported* SRAM are needed in several cases
  Examples:
  - Multicycle MIPS must read two sources or write a result on some cycles
  - Pipelined MIPS must read two sources and write a third result each cycle
  - Superscalar MIPS must read and write many sources and results each cycle

Dual-Ported SRAM

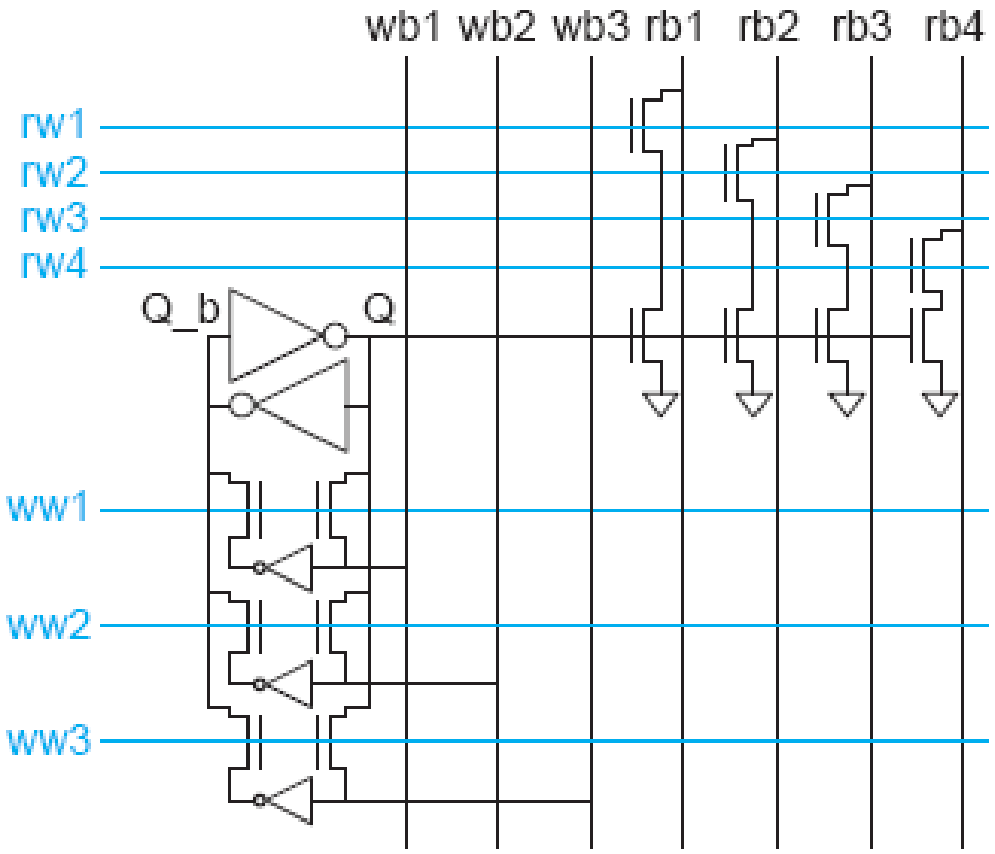
- Simple dual-ported SRAM
  - Two *independent* single-ended reads
  - Or one differential write



- Do two reads and one write by time multiplexing
  - Read during ph1, write during ph2

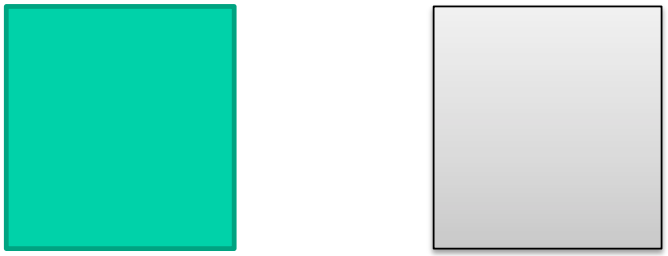
Multi-Ported SRAM

- Adding more access transistors hurts read stability
- Multiported SRAM isolates reads from state node
- Single-ended bitlines save area

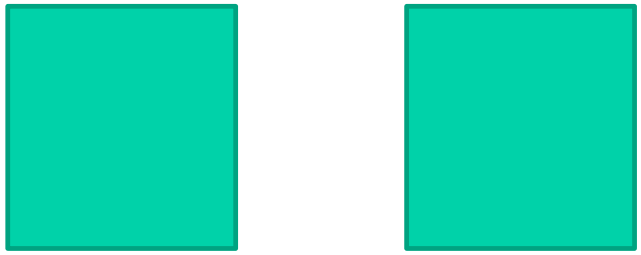


Microarchitectural Alternatives

- **Banking**



- **Replication**



In this class

- Step 1: Try to use SRAM macros
  - /proj/castl/development/synopsys/SAED_EDK90nm/Memories/doc/databook/Memories_Rev1_6_2009_11_30.pdf
  - Please do not e-mail, distribute or post online
- Step 2: Build small memories (1K bits) using flip-flops
- Step 3: Use memory compiler for larger non-standard sizes (instructions provided before project)

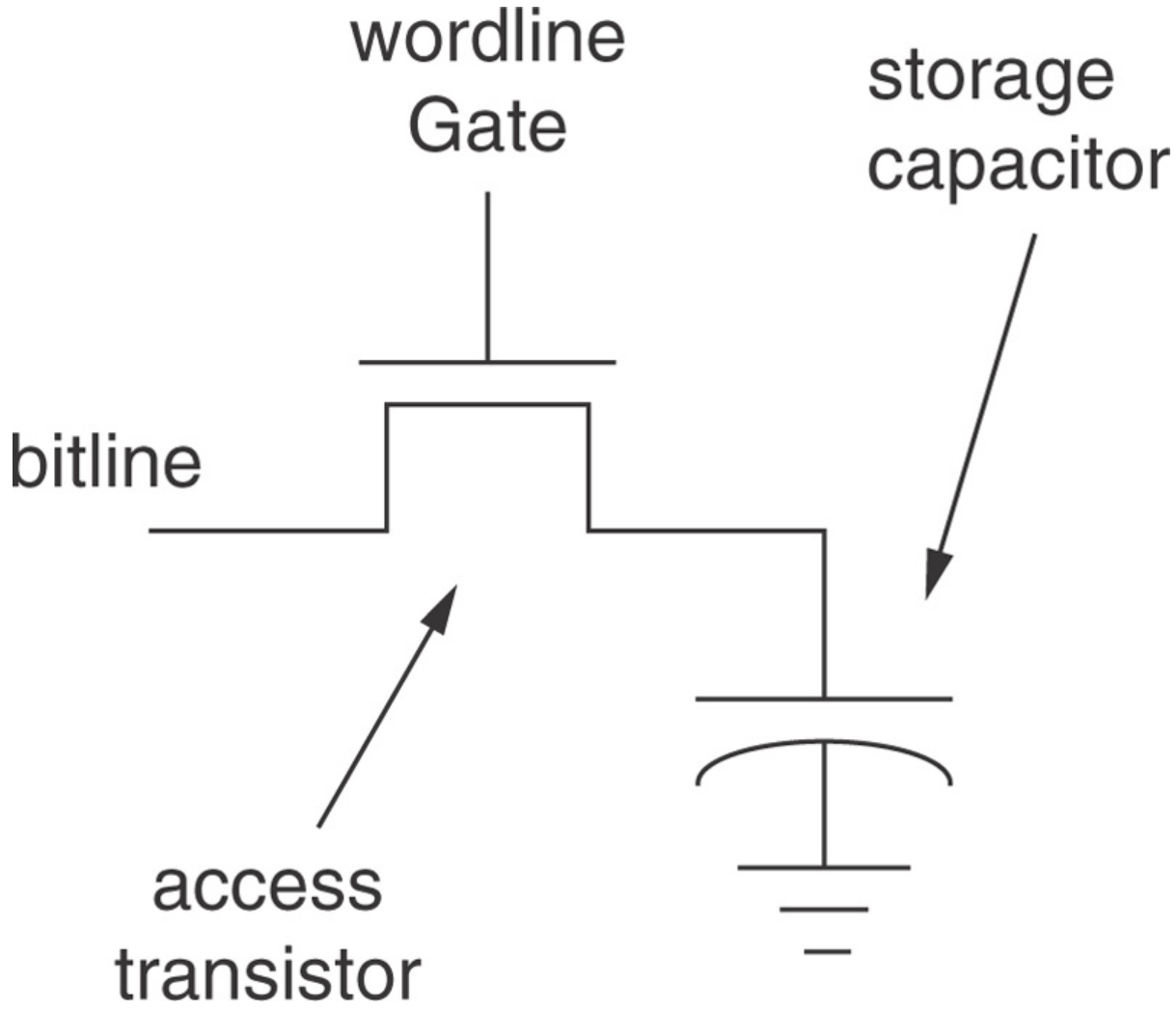
DRAM Reference

Reference Chapters:

Chapter 8 (353 – 376)
Chapter 10 (409 - 424)
Chapter 11 (425 - 456

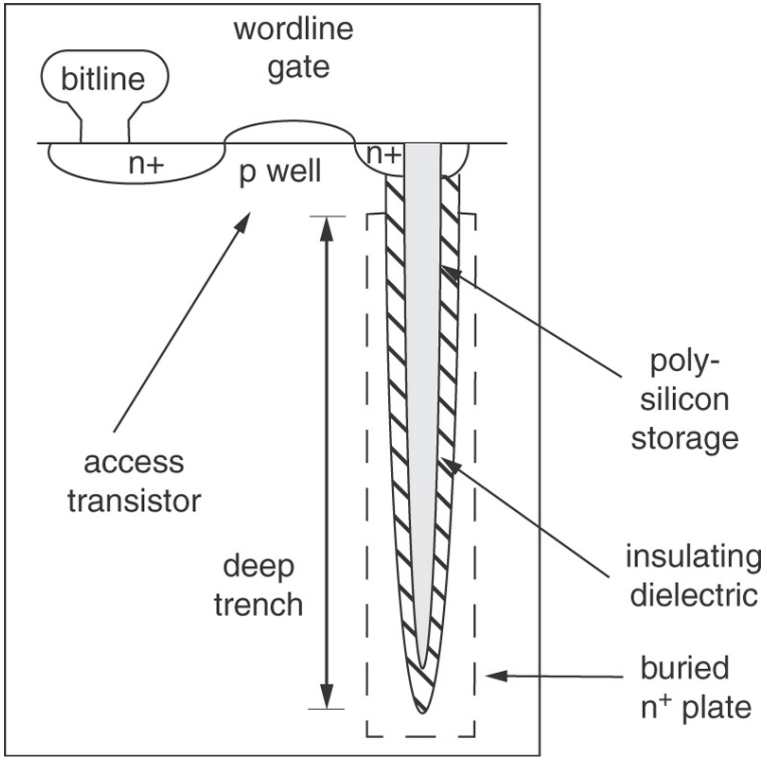
Or listen to lectures and take notes

DRAM: Basic Storage Cell

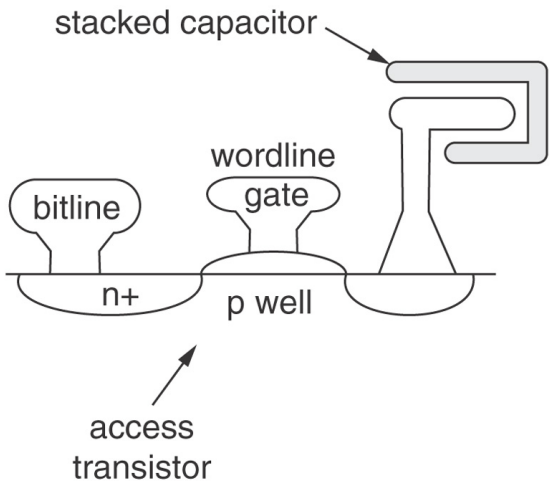


DRAM Cell Implementations

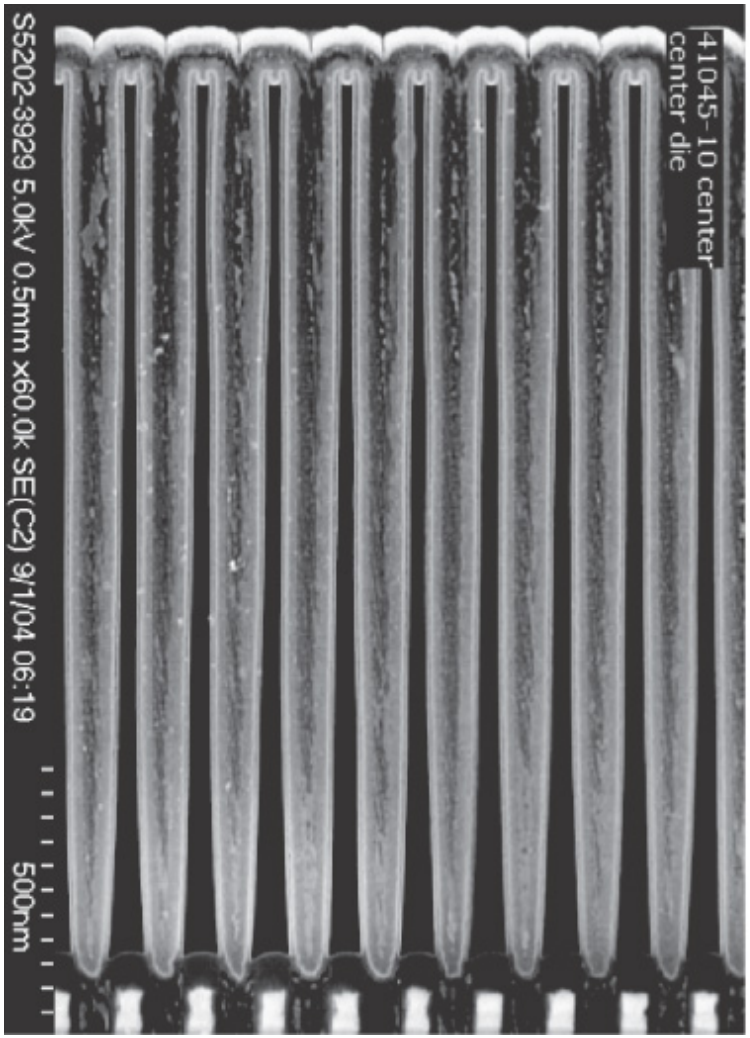
Trench

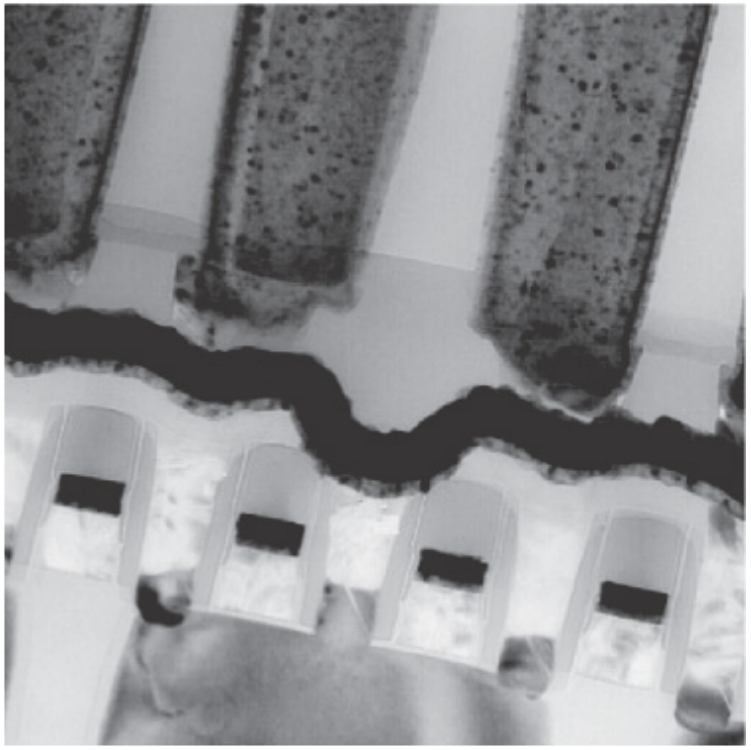


Stacked

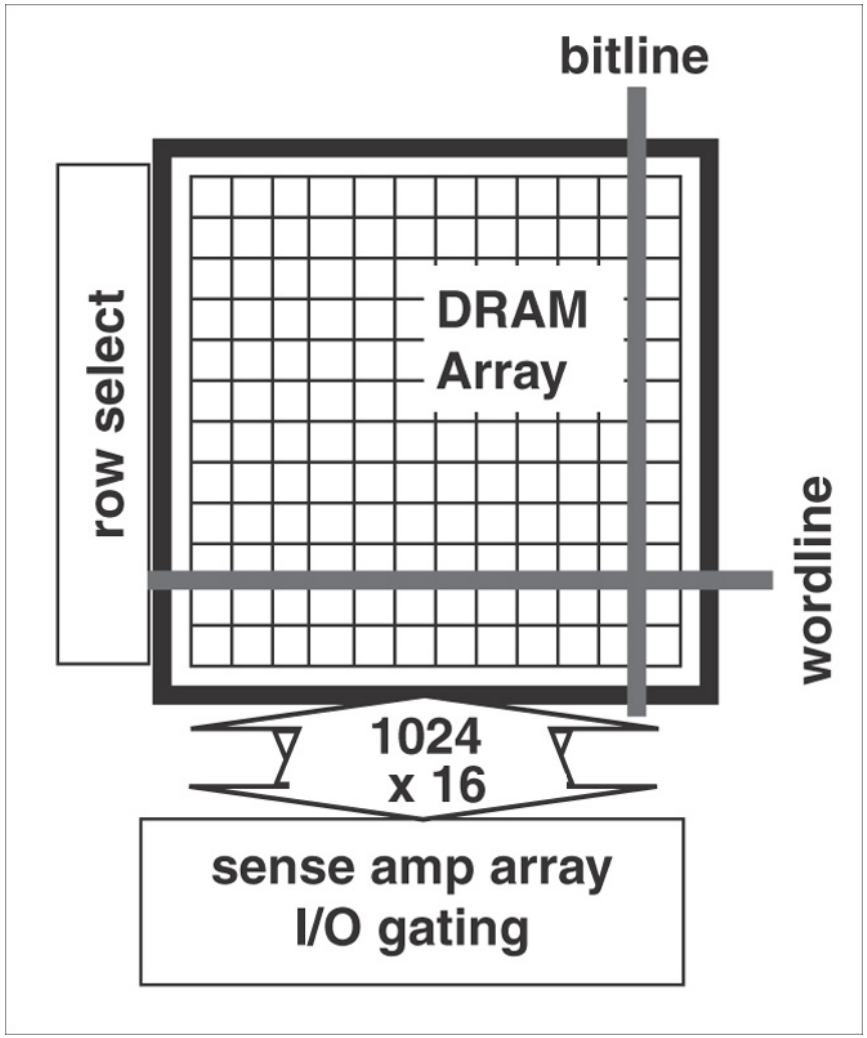


DRAM Cell Implementations

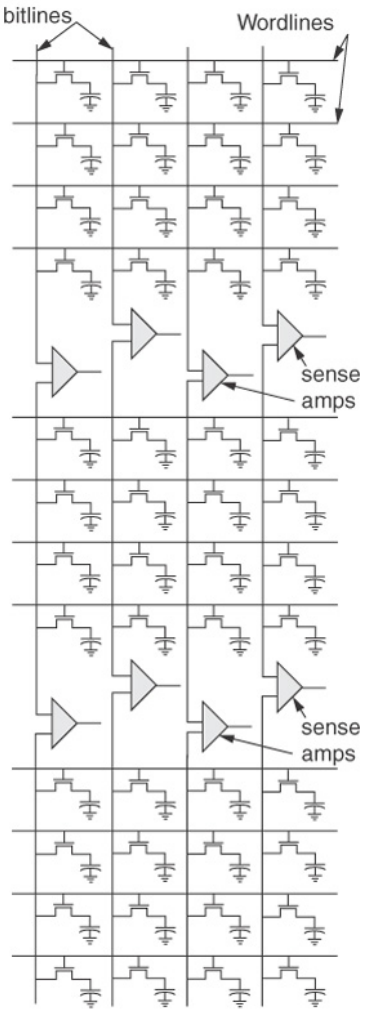




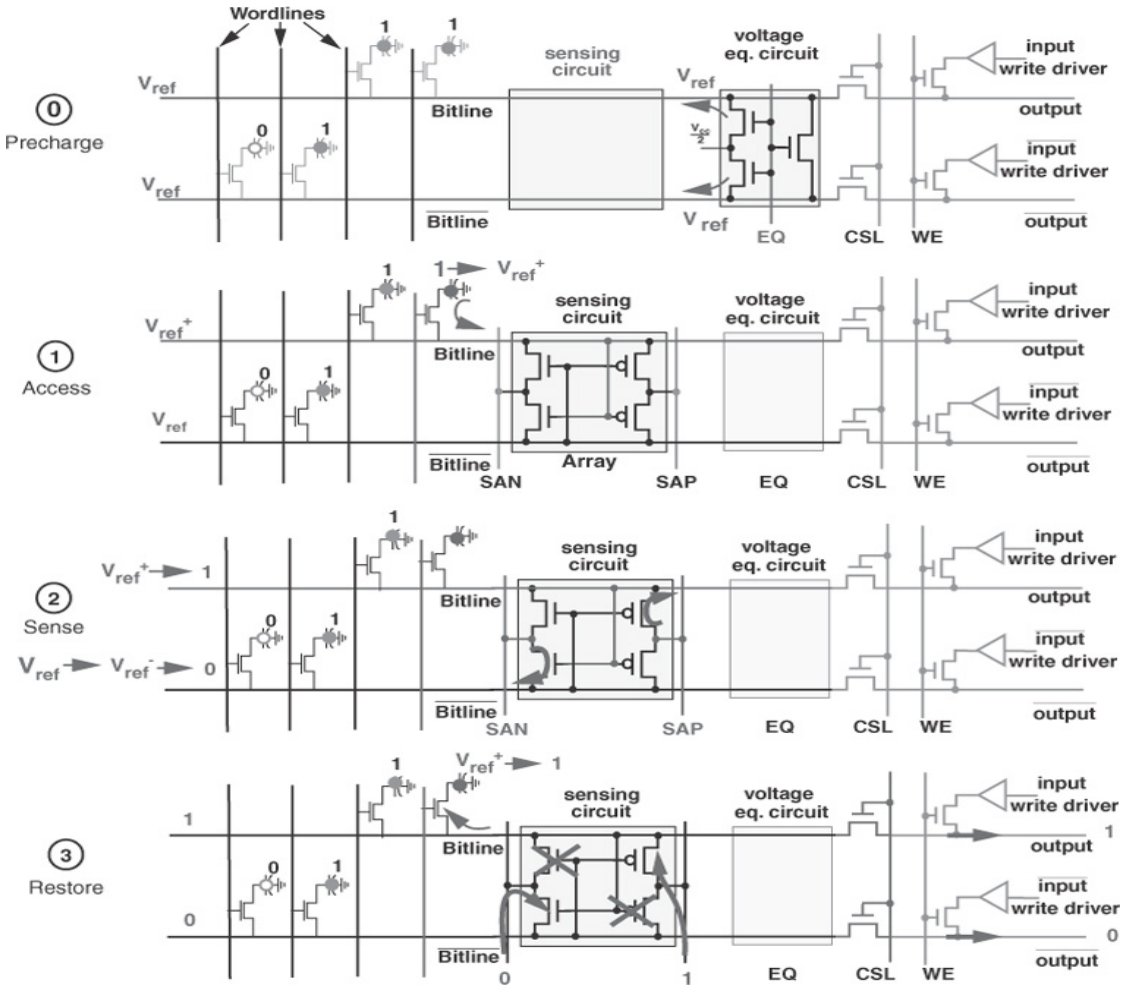
DRAM Array


bitline
row select
DRAM
Array
wordline
1024
x 16
sense amp array
I/O gating

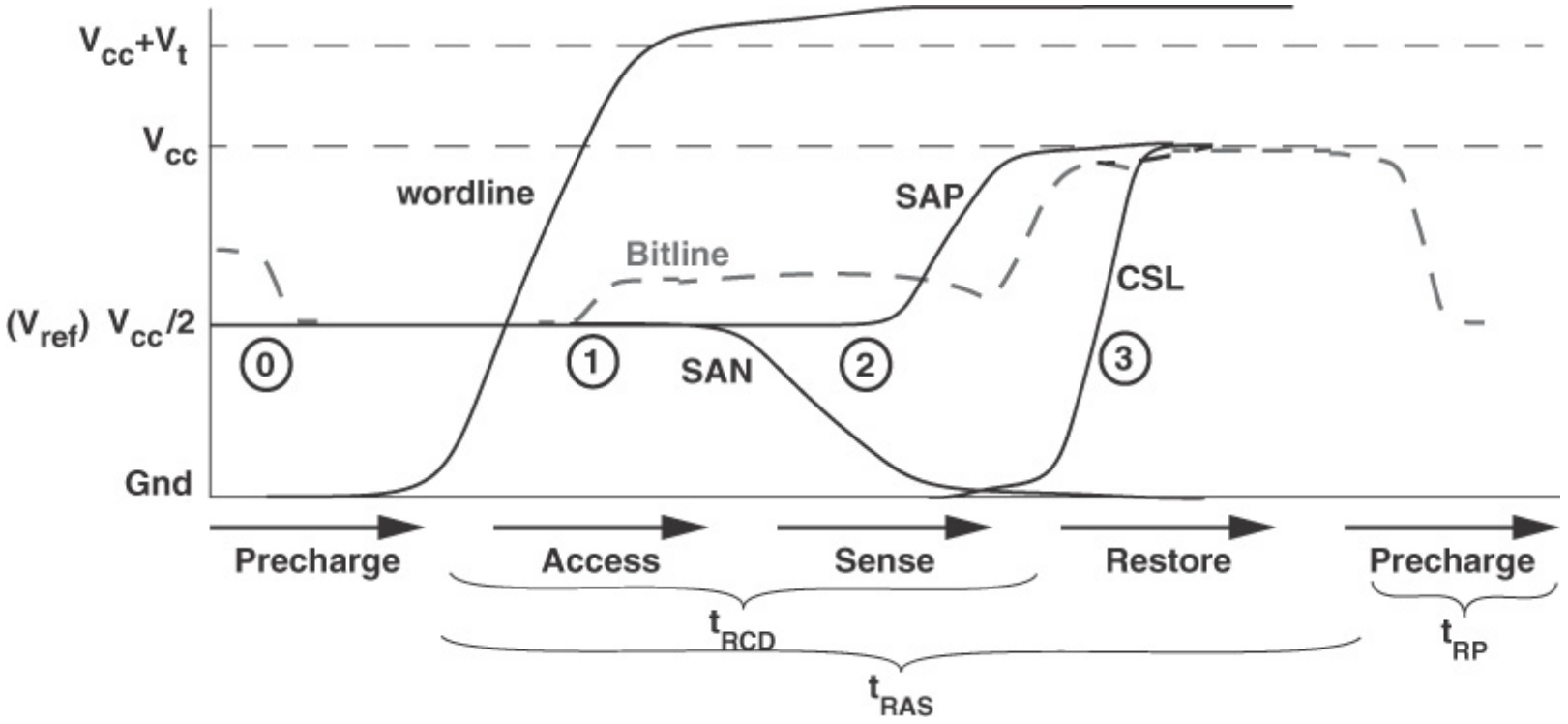
Bitlines with Sense Amps



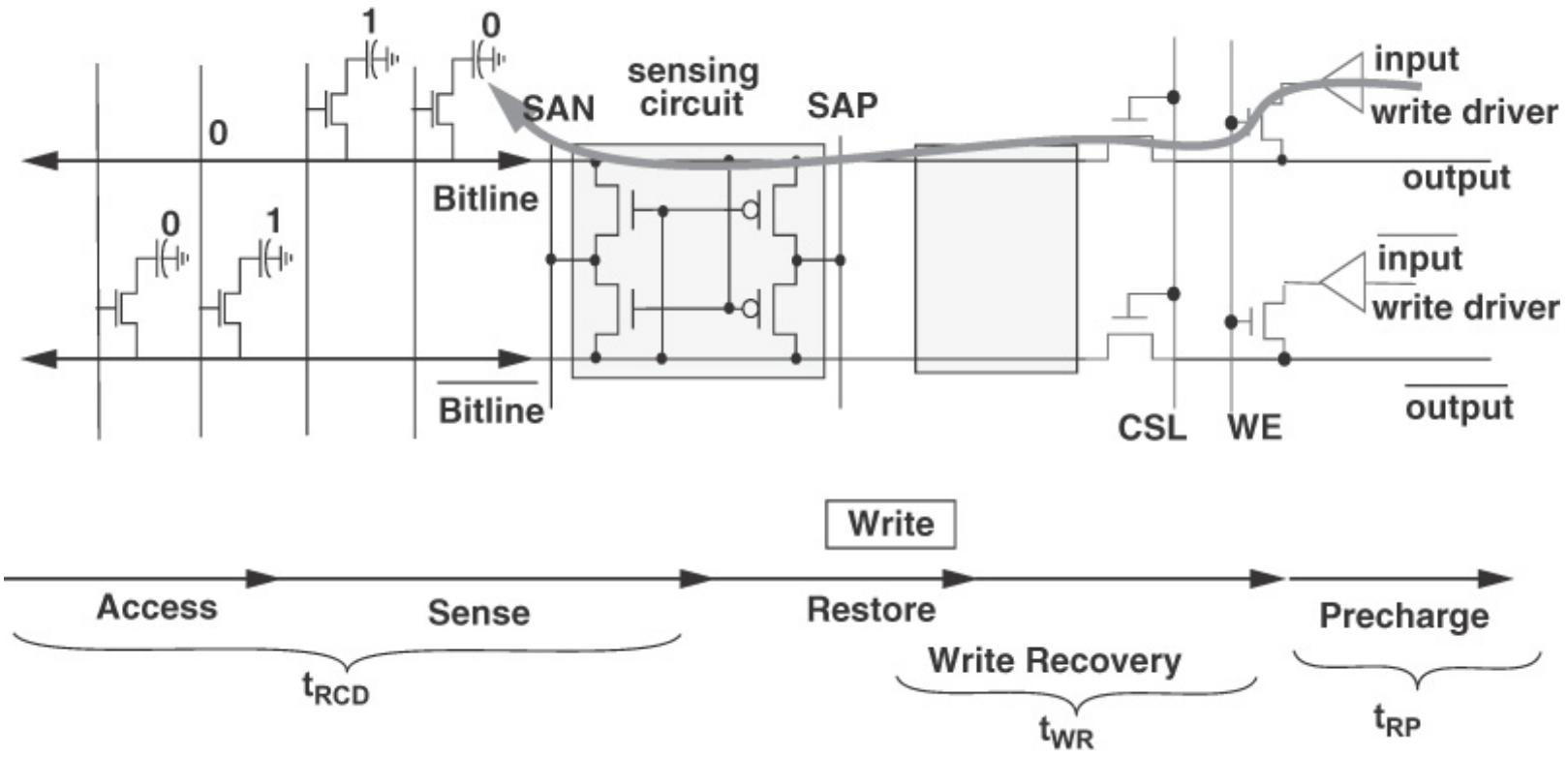
Read Operation



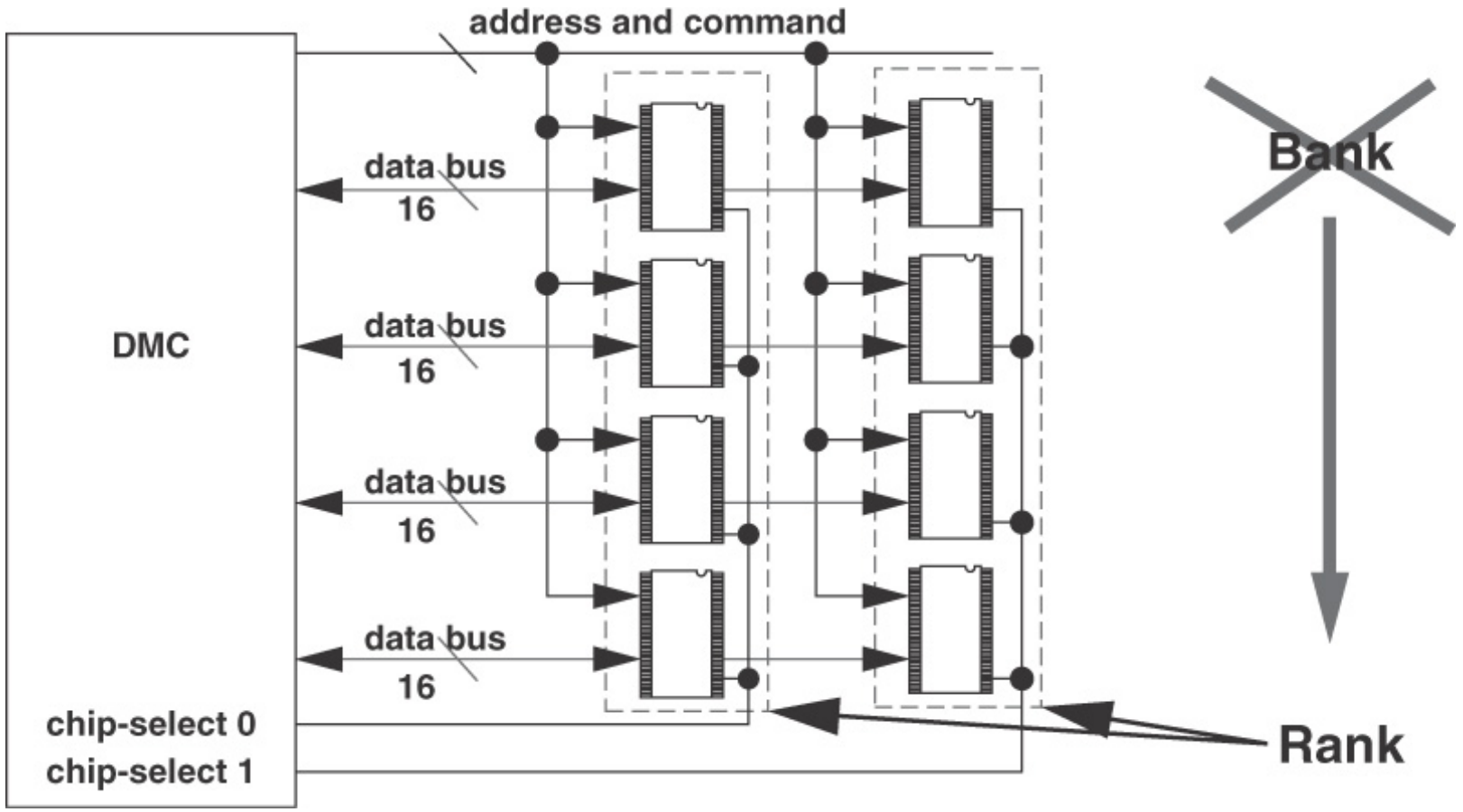
Timing of Reads



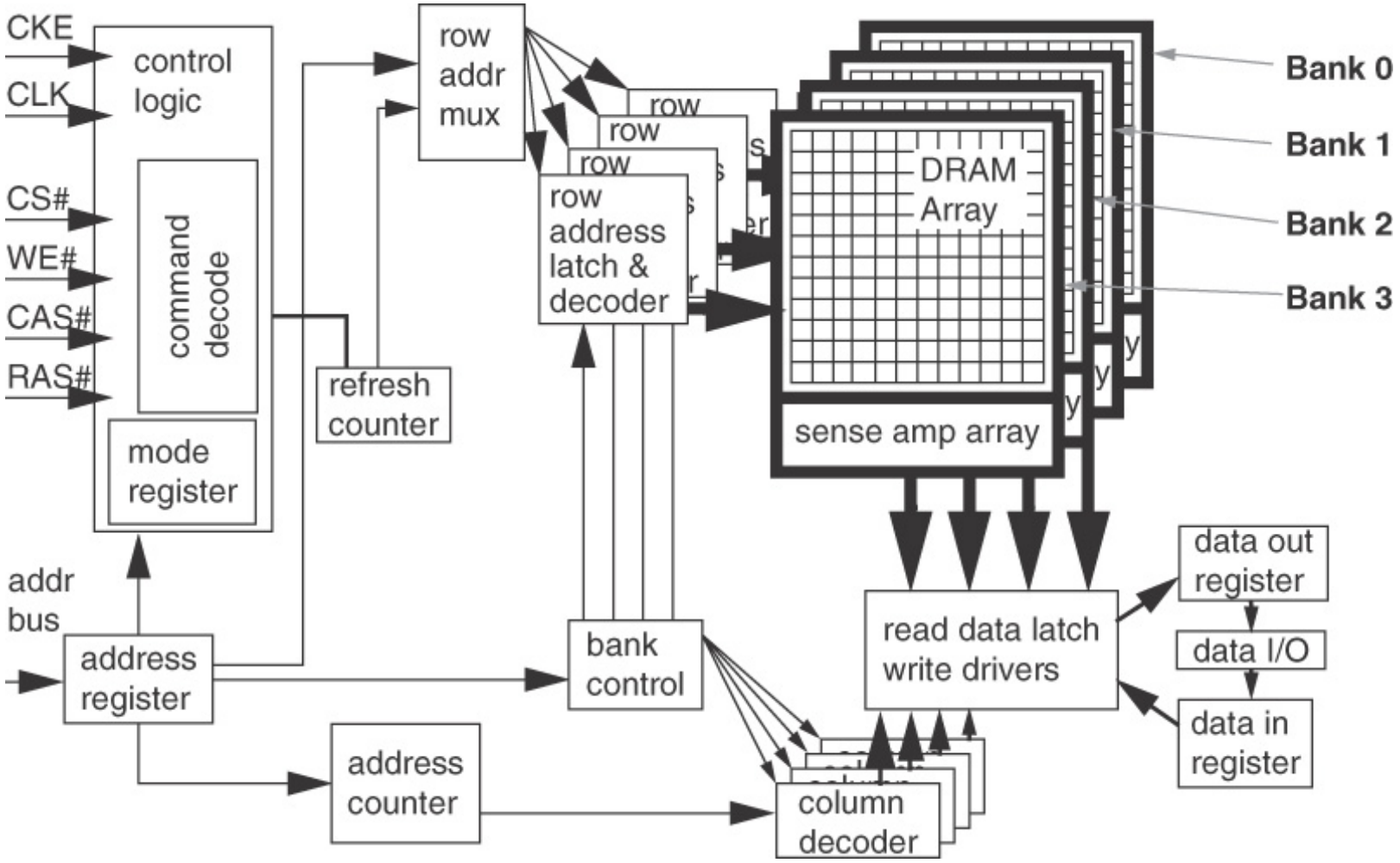
Timing of Writes



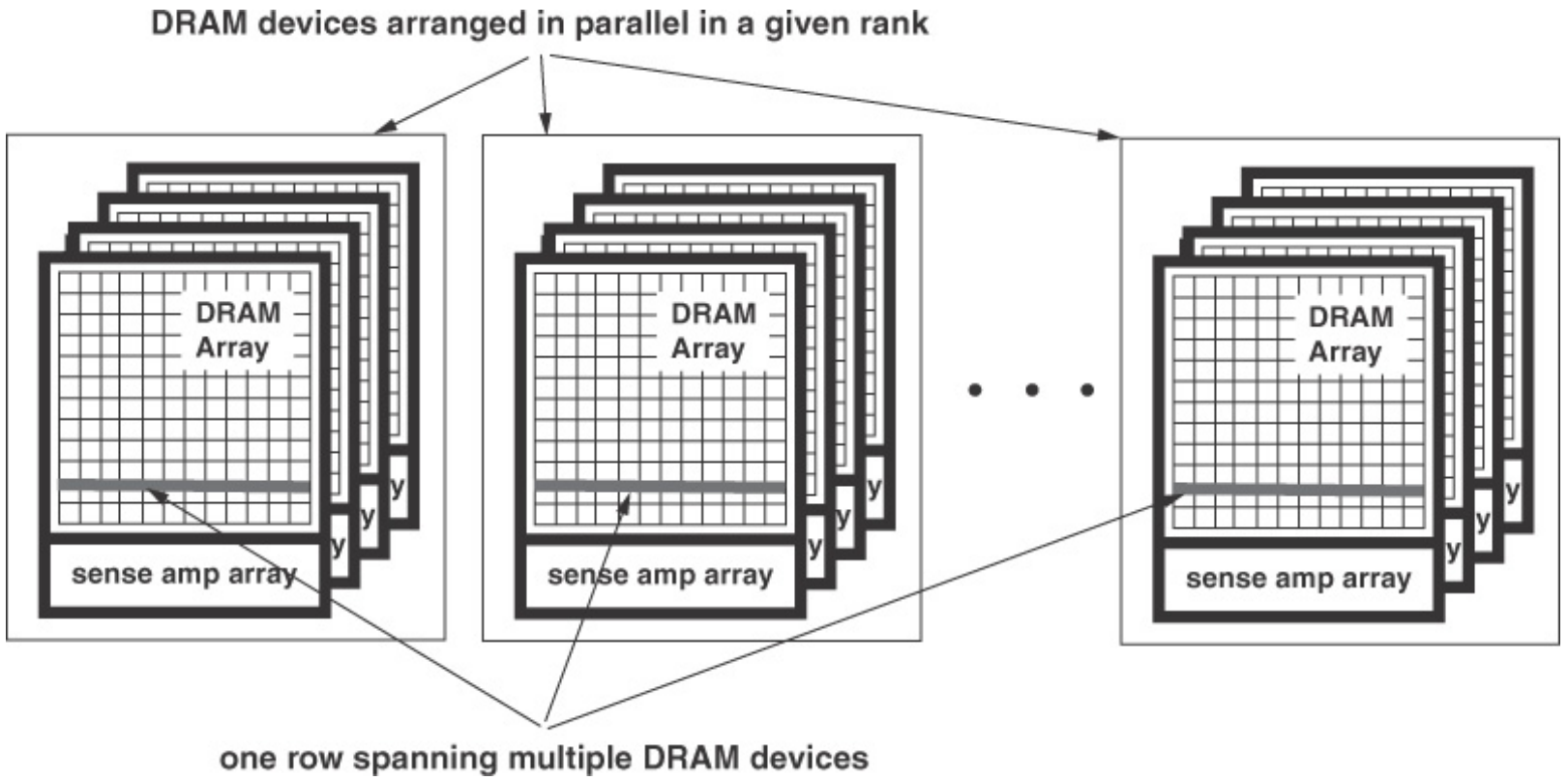
DRAM System Organization



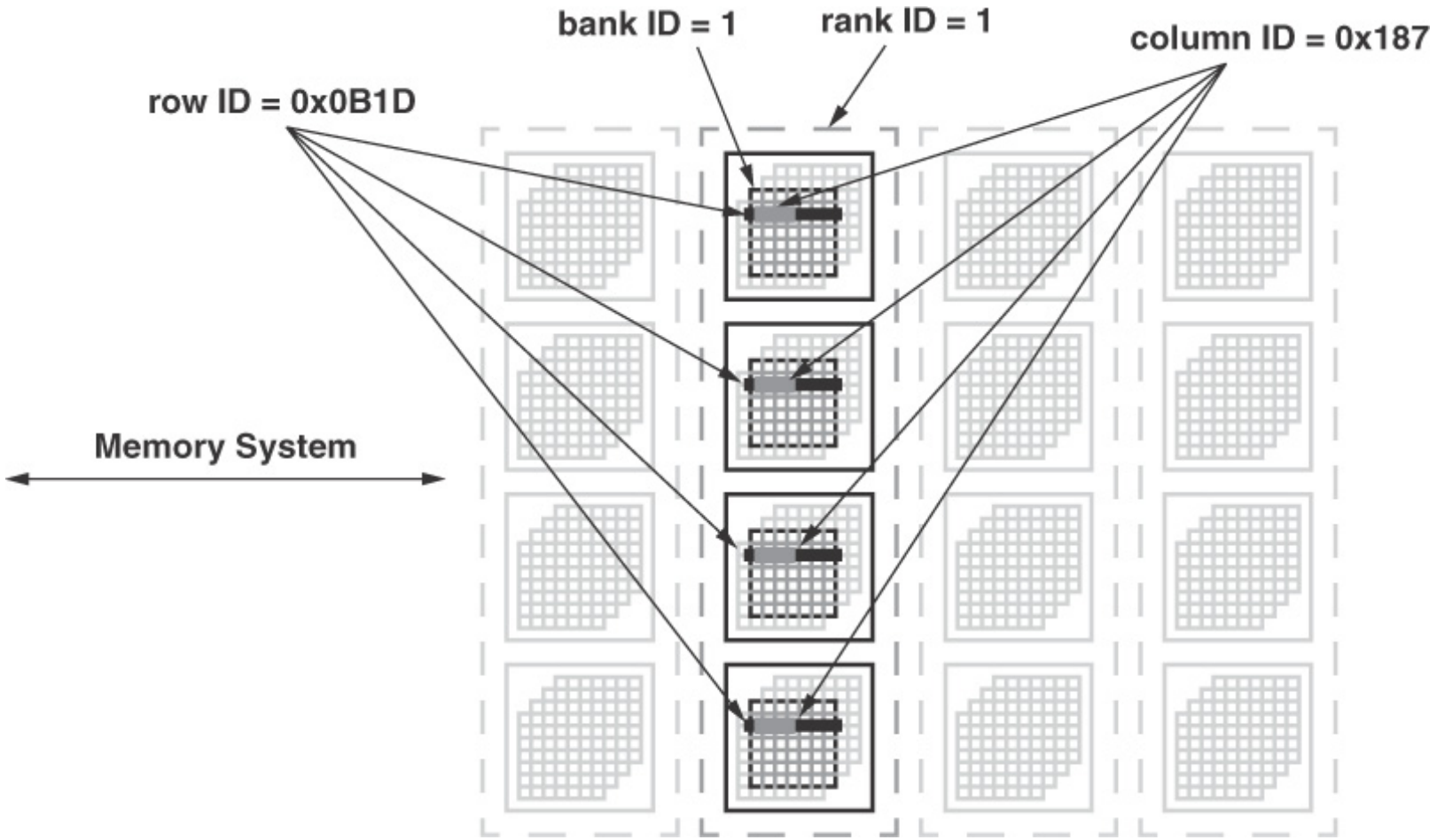
SDRAM Device Internals



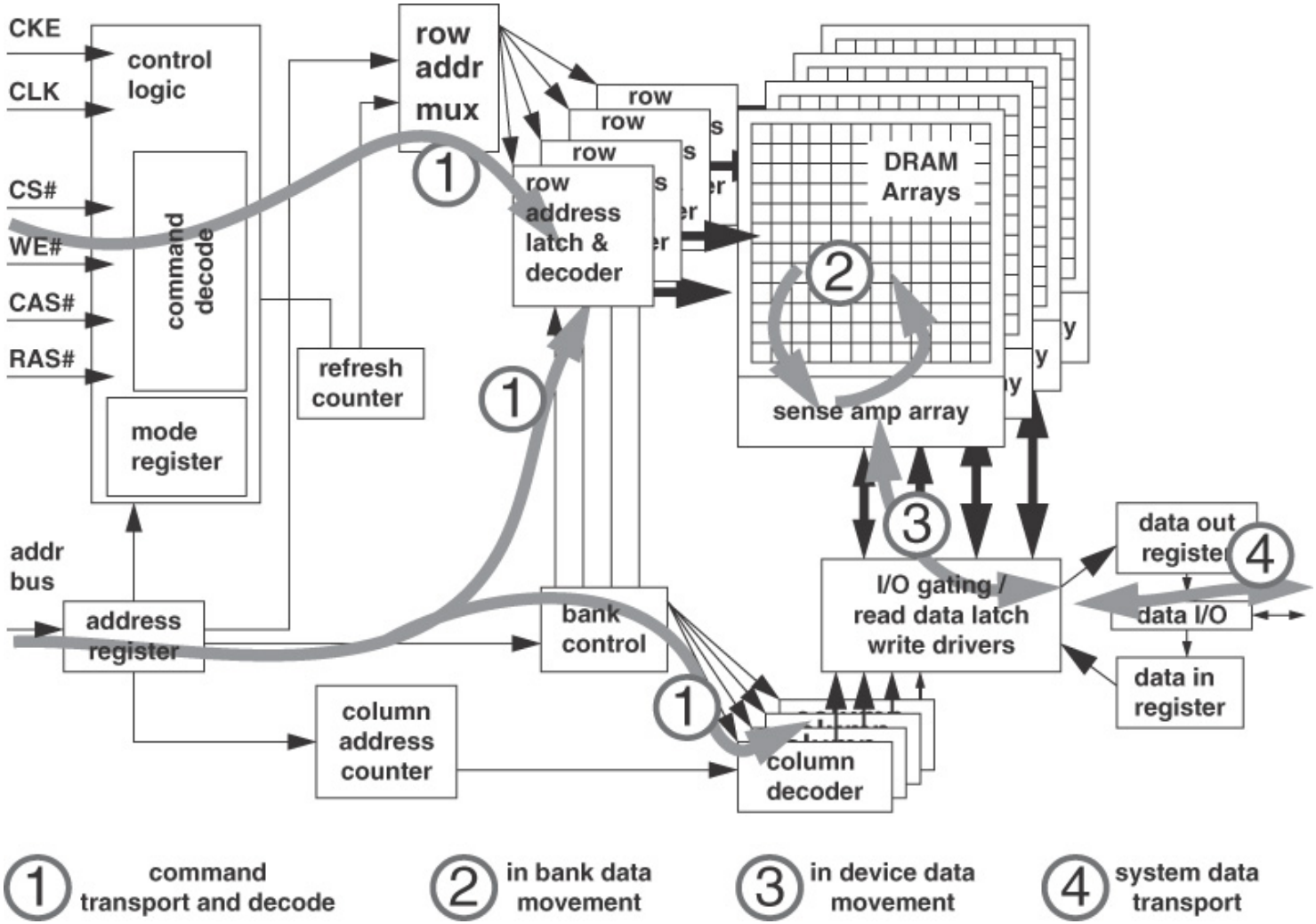
More Nomenclature



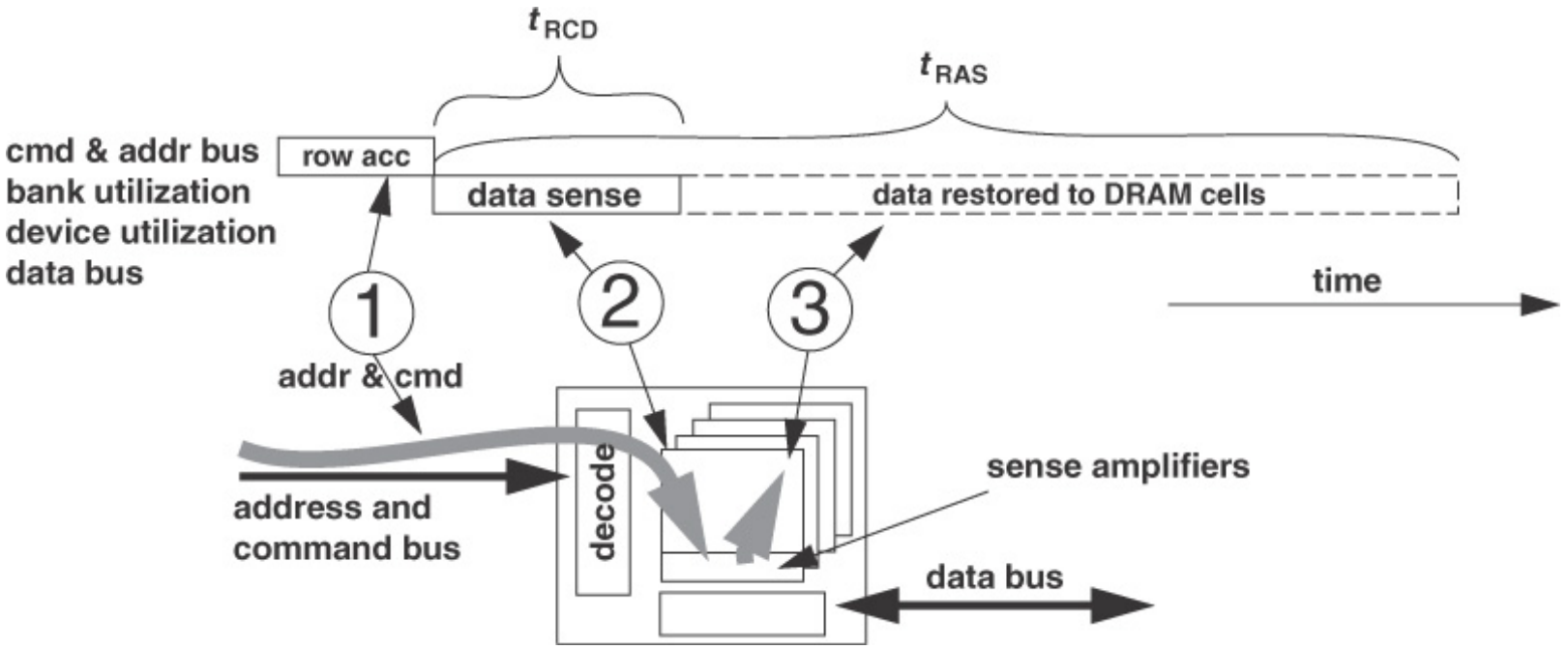
Example Address Mapping



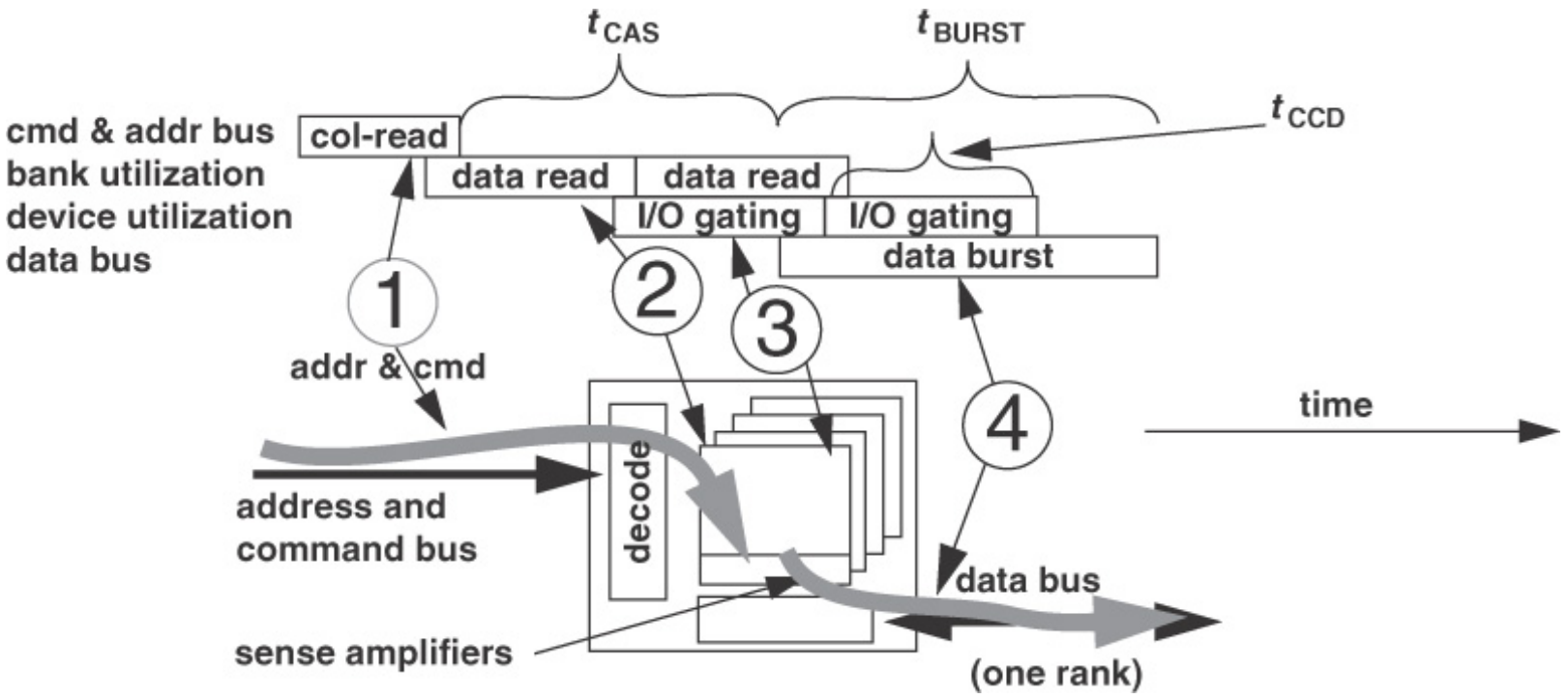
Command and Data Movement



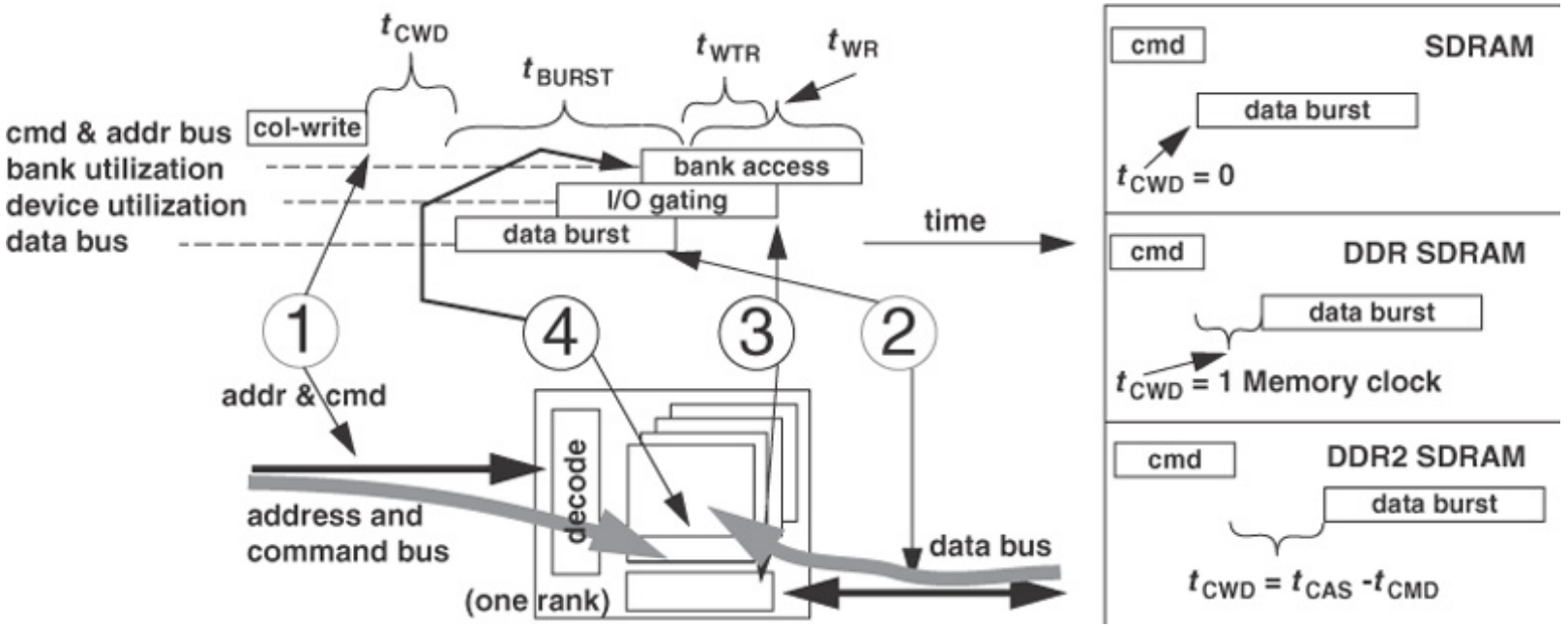
Timing: Row Access Command



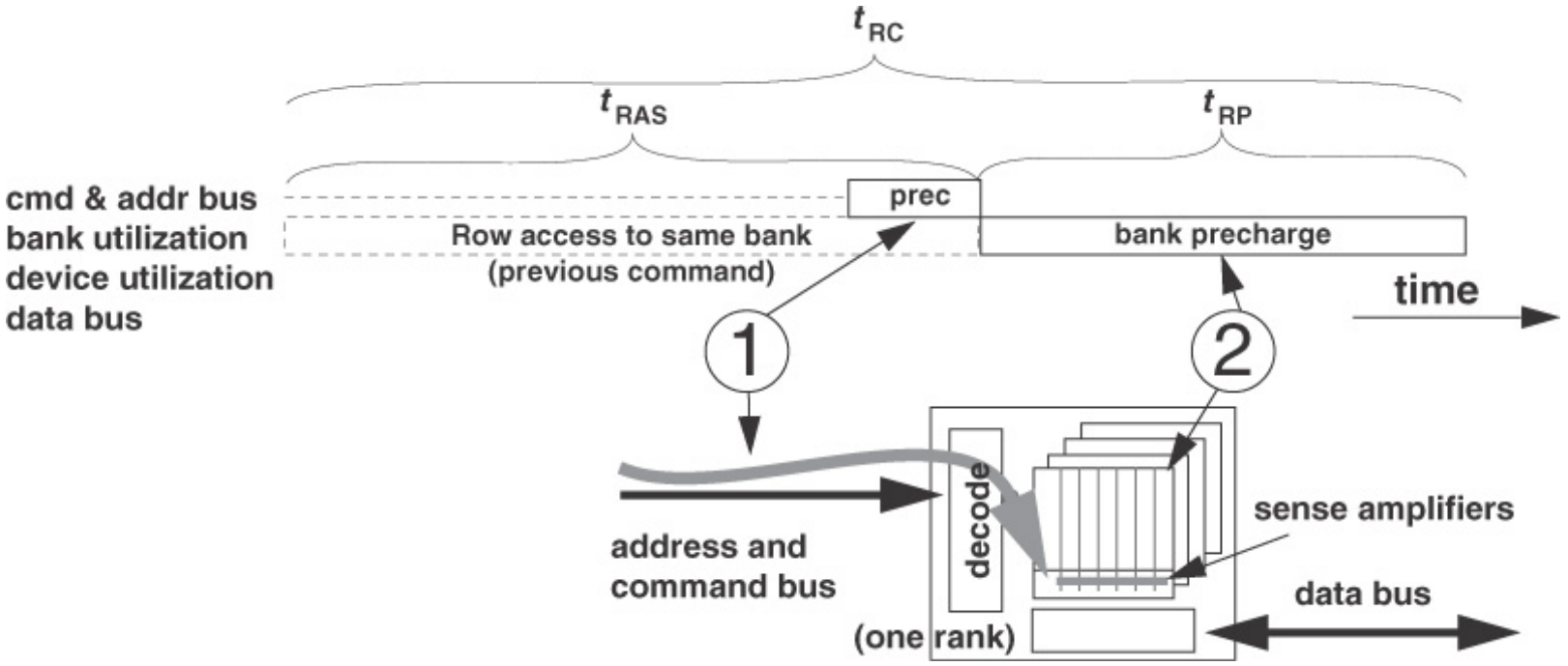
Timing: Column-Read Command



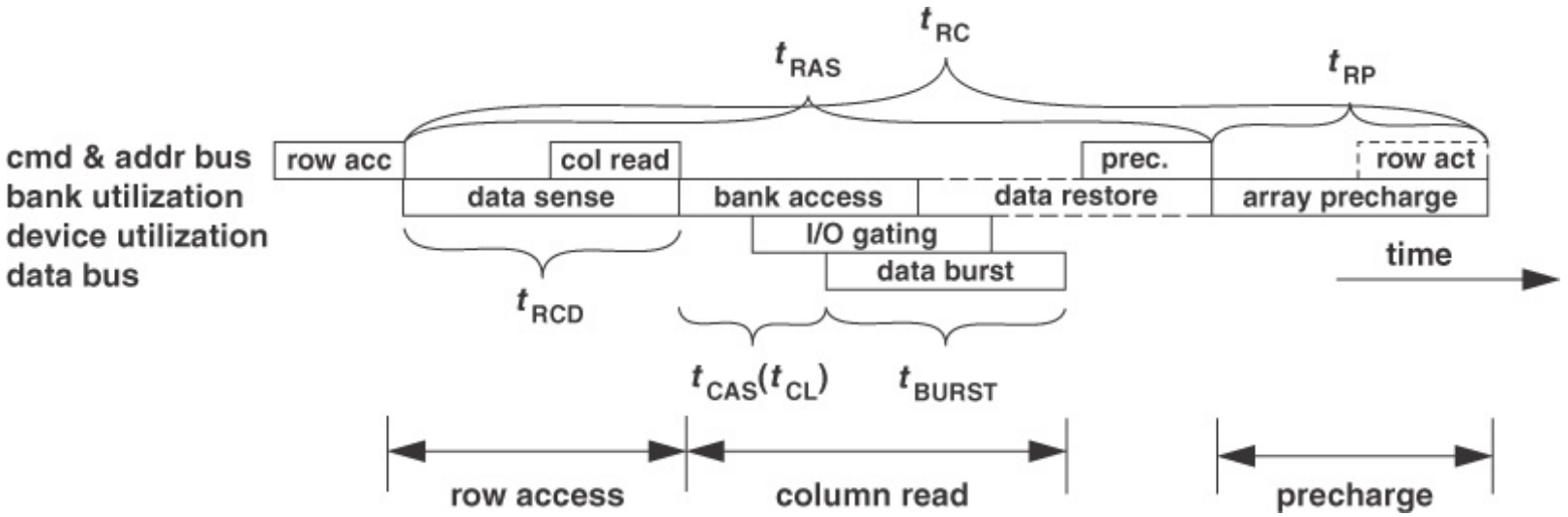
Protocols: Column-Write Command



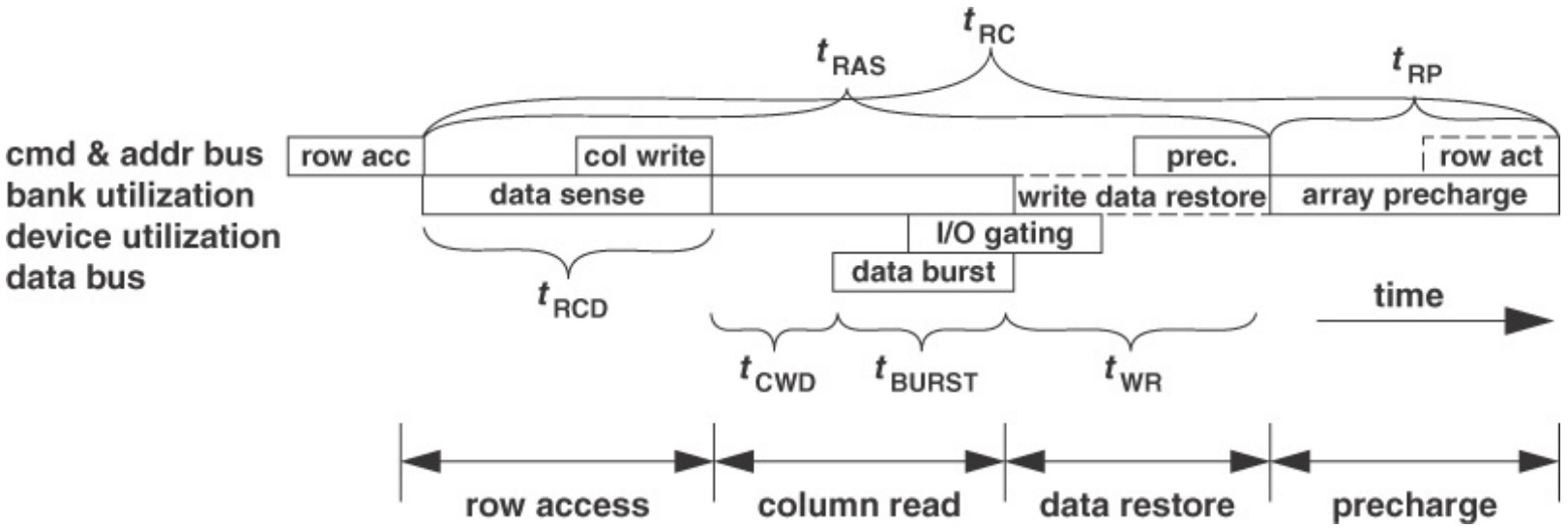
Timing: Row Precharge Command



One Read Cycle



One Write Cycle



Common Values for DDR2/DDR3

For more information refer to Micron or Samsung Datasheets. Uploaded to class website.

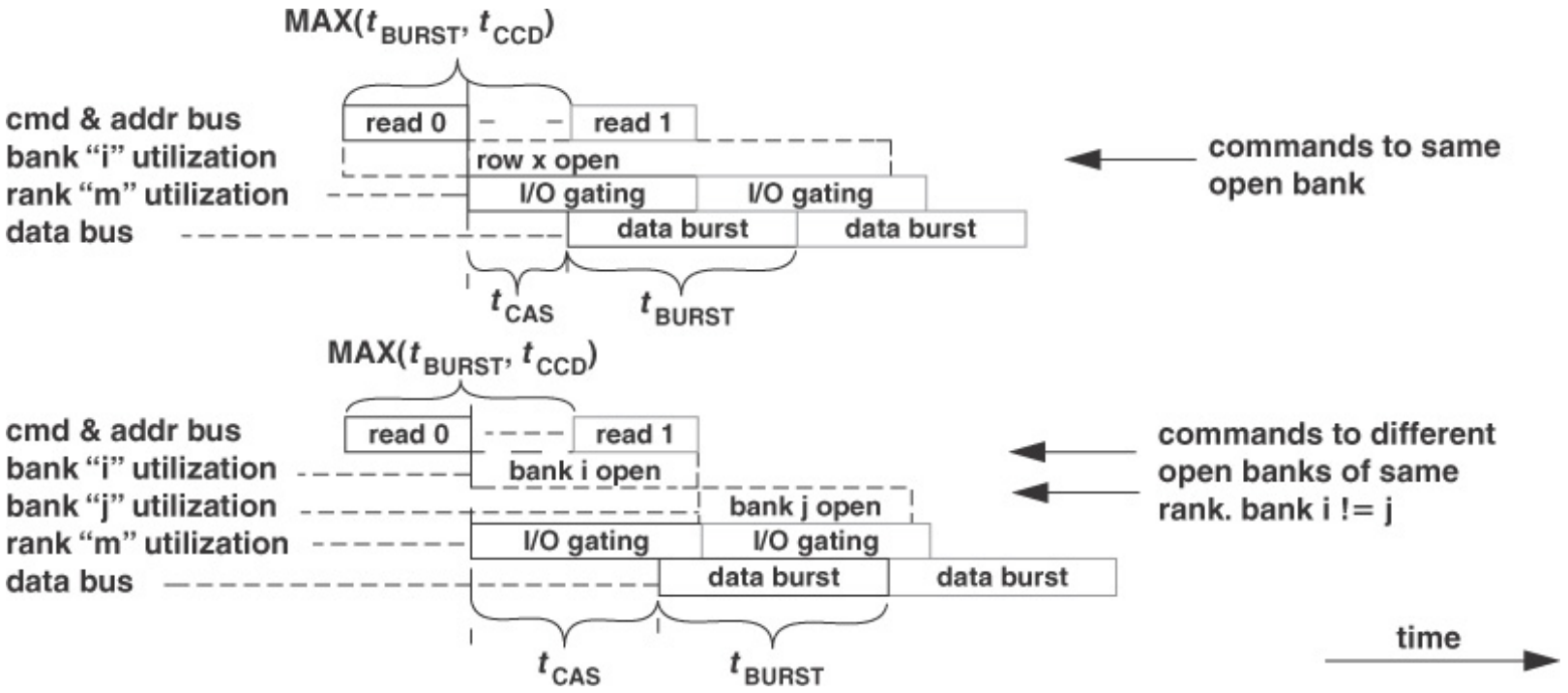
Technology Parameter	DDR2 [30]	DDR3 [31, 33]	LVDDR3 [31, 33]	LPDDR [21, 37]	LPDDR2 [35, 37]
Operating Voltage	1.8V	1.5V	1.35V	1.8V	1.2V
Operating Frequency	400MHz	800MHz	400MHz	200MHz	400MHz
Typical Device Width (pins)	4	8	8	16	16
Peak Channel Bandwidth (sequential)	6.4GBps	12.8GBps	6.4GBps	3.2GBps	6.4GBps
Dynamic					
Timing (CAS, RAS, RC)	12, 40, 55ns	15, 38, 50ns	15, 38, 50ns	12, 40, 54ns	15, 42, 57ns
Active Current (read, write)	160, 160mA	180, 185mA	125, 130mA	130, 130mA	210, 175mA
Energy per bit (peak, typical)	111, 266mW/Gbps	70, 160 mW/Gbps	110, 190 mW/Gbps	110, 140 mW/Gbps	40, 50 mW/Gbps
Static					
Idle current (power-down, standby)	50, 70mA	35, 45mA	22, 32mA	3.6, 20mA	1.6, 23mA
Min power-down period	84ns	90ns	90ns	20ns	20ns
Slow Powerdown Exit latency	20ns	24ns	24ns	7.5ns	7.5ns

SOURCE:

Krishna T. Malladi, Benjamin C. Lee, Frank A. Nothaft, Christos Kozyrakis, Karthika Periyathambi, and Mark Horowitz. 2012. Towards energy-proportional datacenter memory with mobile DRAM. SIGARCH Comput. Archit. News 40, 3 (June 2012), 37-48. DOI=10.1145/2366231.2337164 http://doi.acm.org/10.1145/2366231.2337164

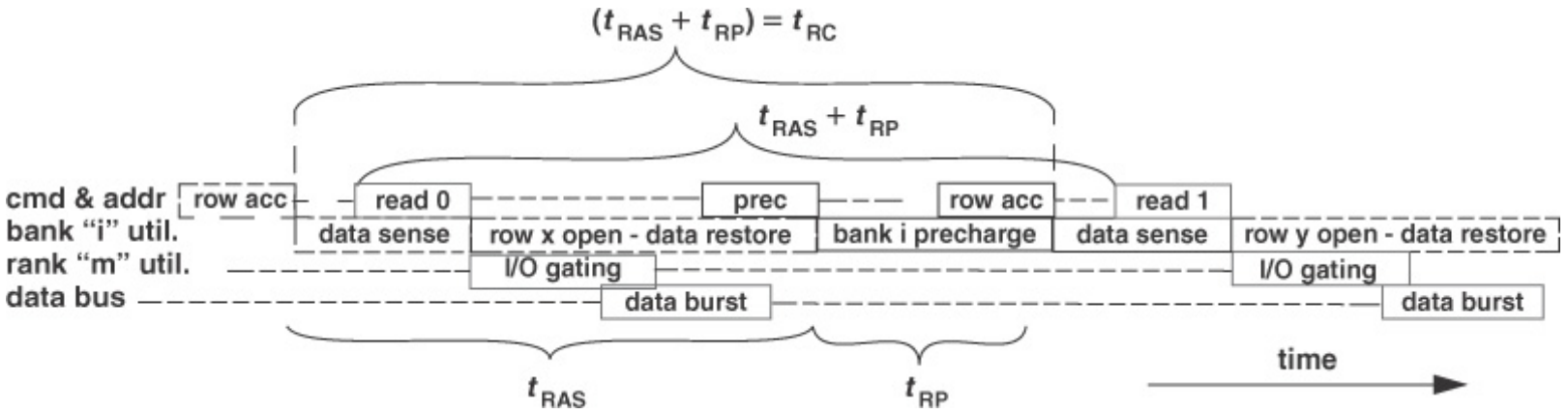
Multiple Command Timing

Consecutive Rds to Same Rank



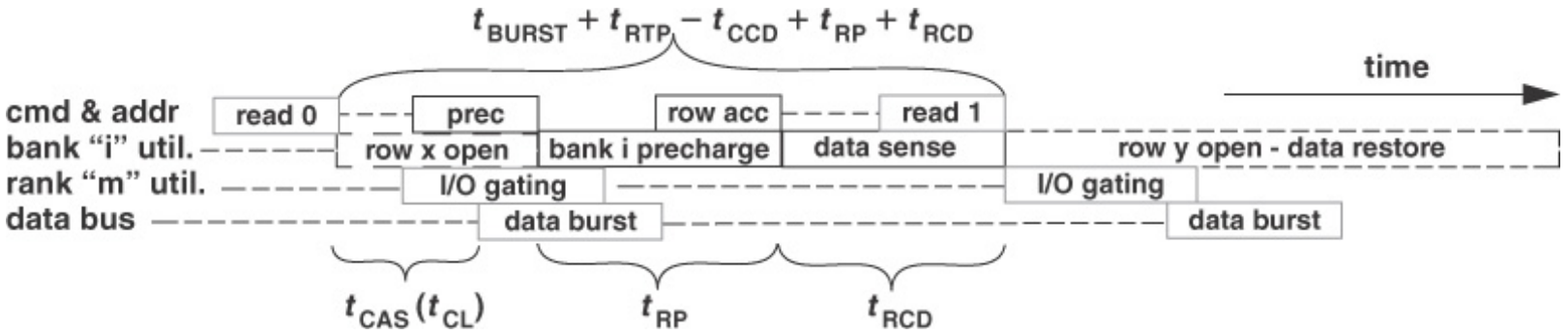
Consecutive Rds: Same Bank, Diff Row

Worst Case



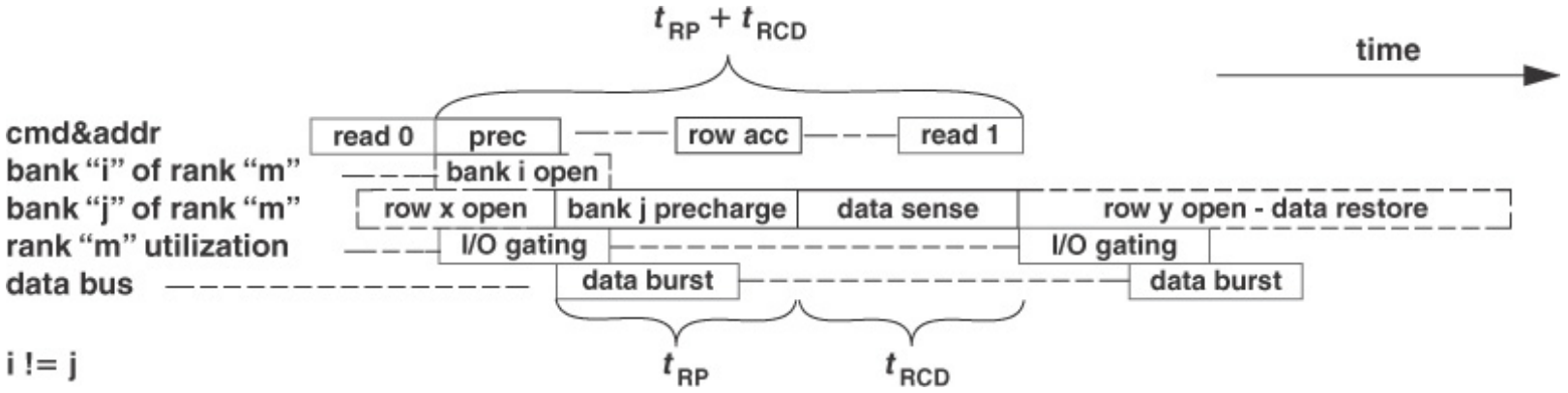
Consecutive Rds: Same Bank, Diff Row

Best Case



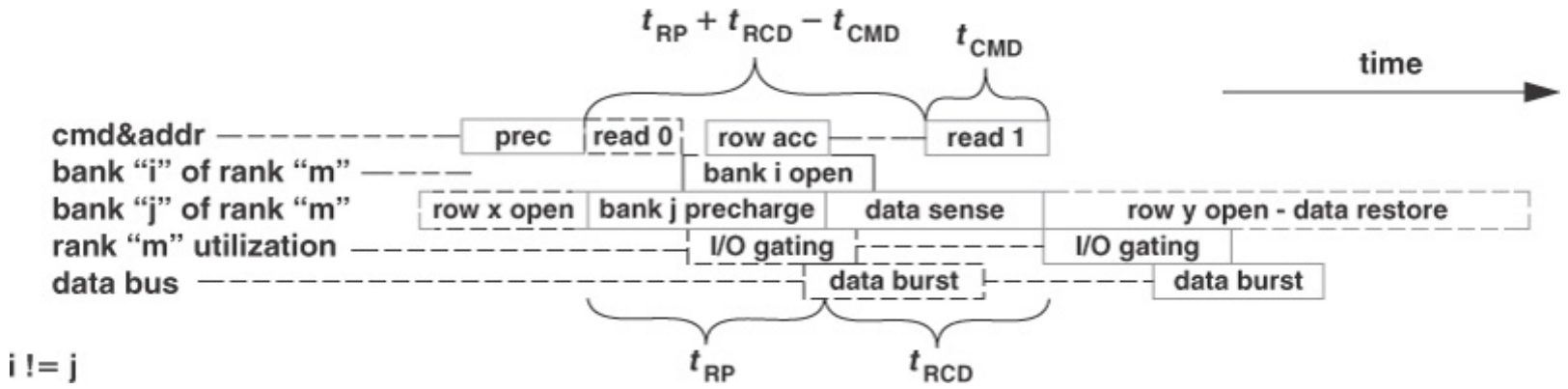
Consecutive Rd's to Diff Banks w/ Conflict

w/o command reordering



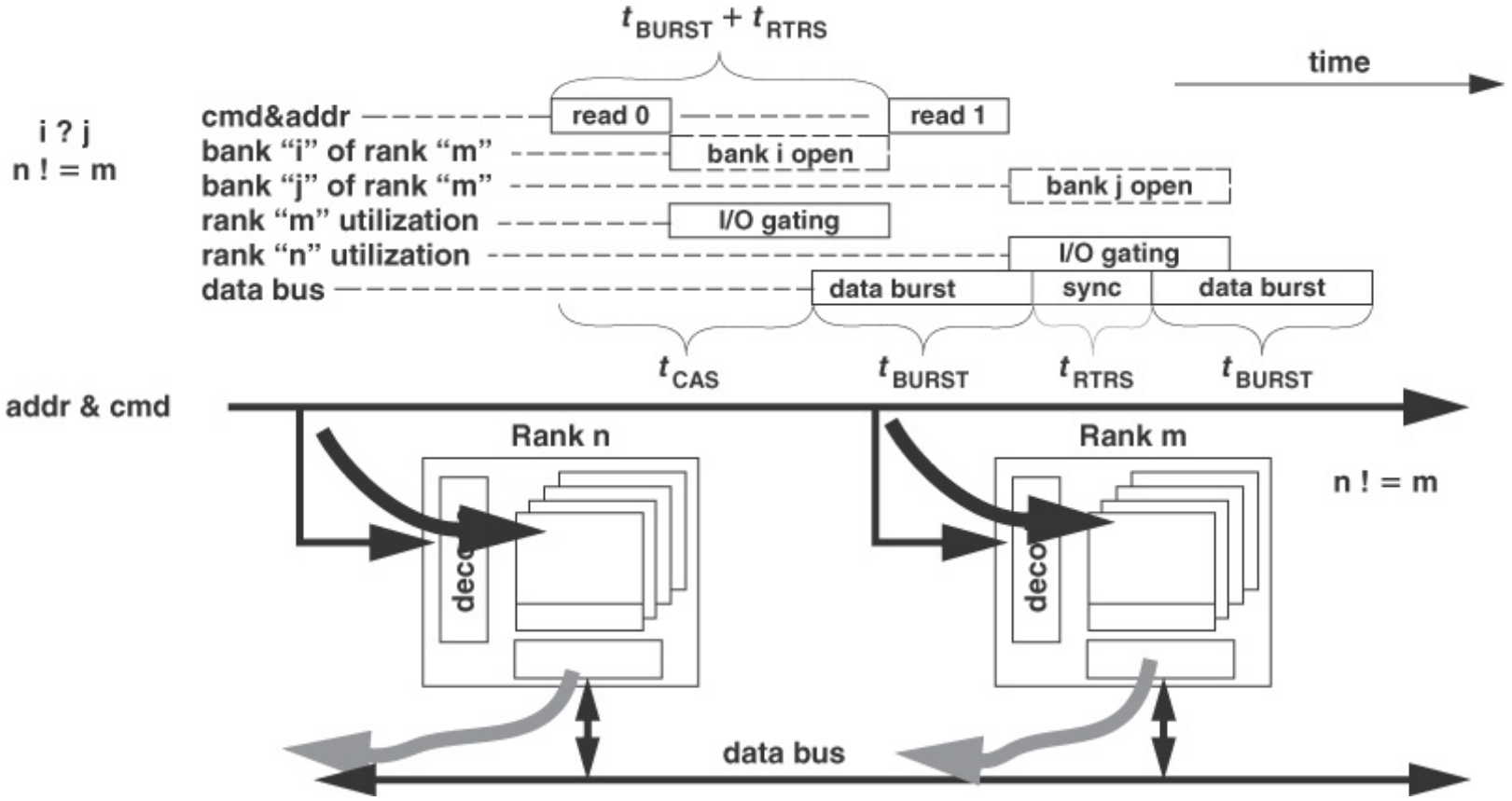
Consecutive Rd's to Diff Banks w/ Conflict

w/ command reordering

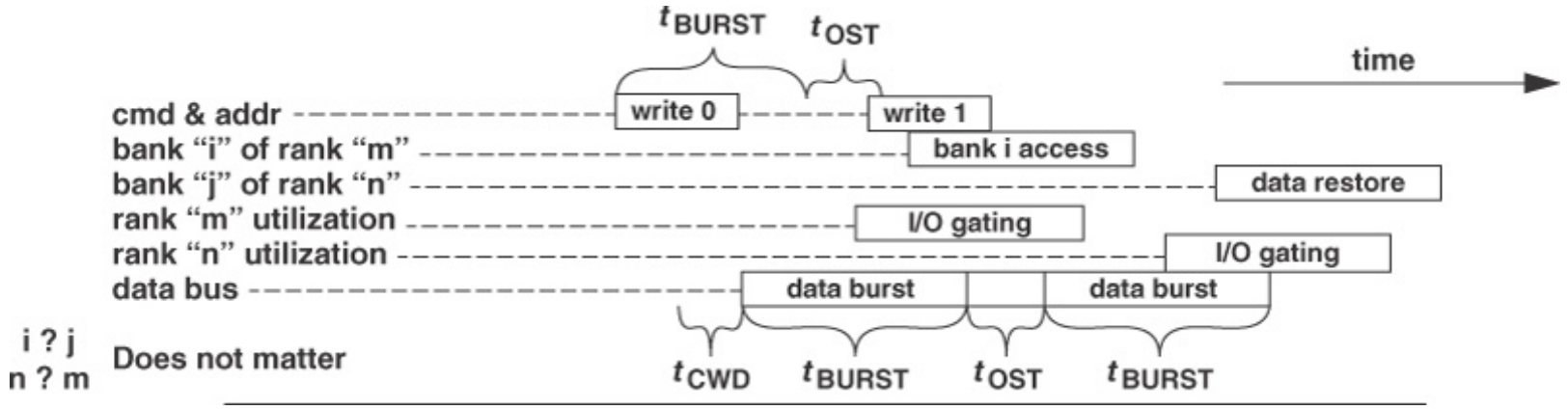


Consecutive Col Rds to Diff Ranks

SDRAM: RTRs = 0
DDRx = non-zero

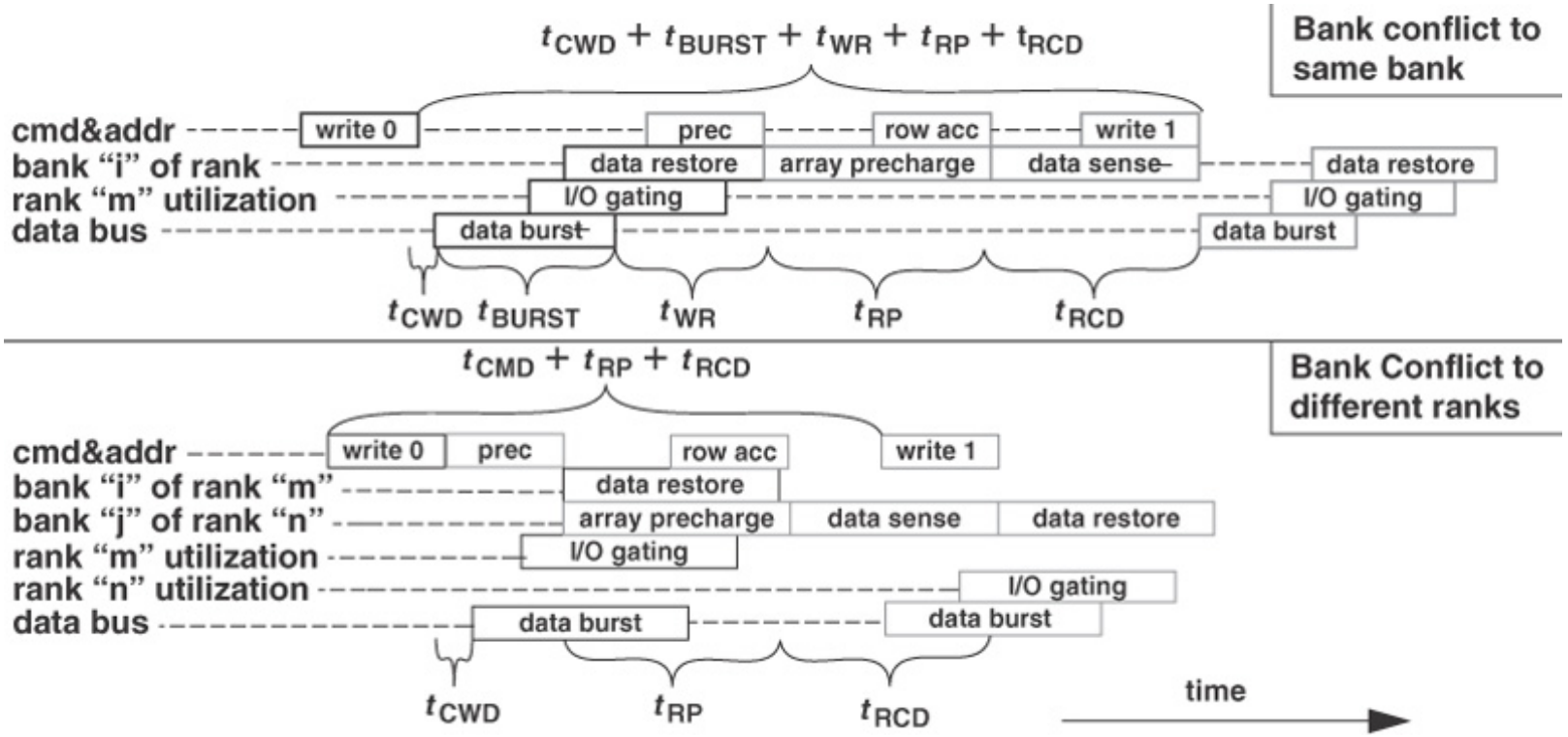


Consecutive Col-WRs to Diff Ranks

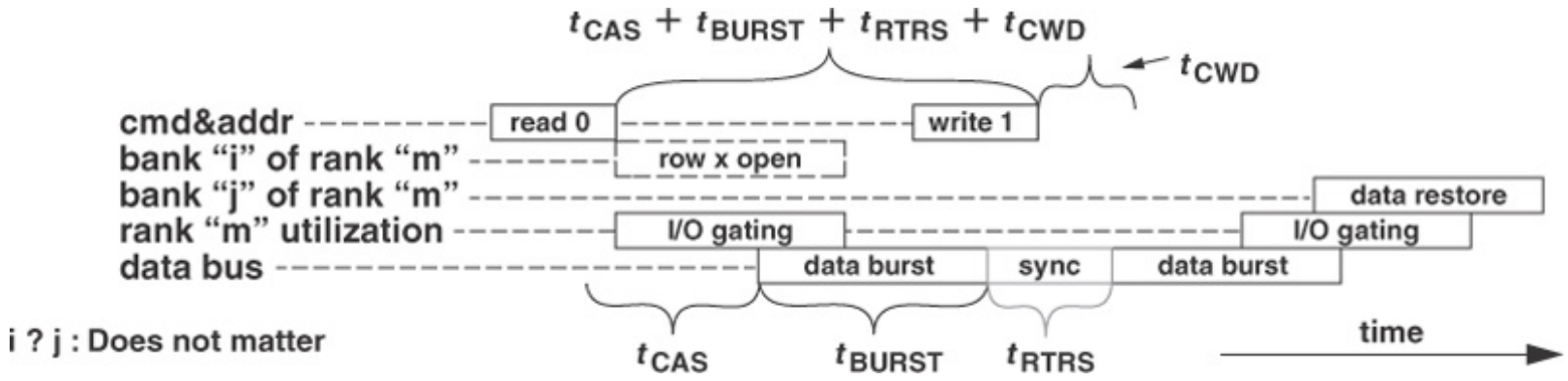


On your Own

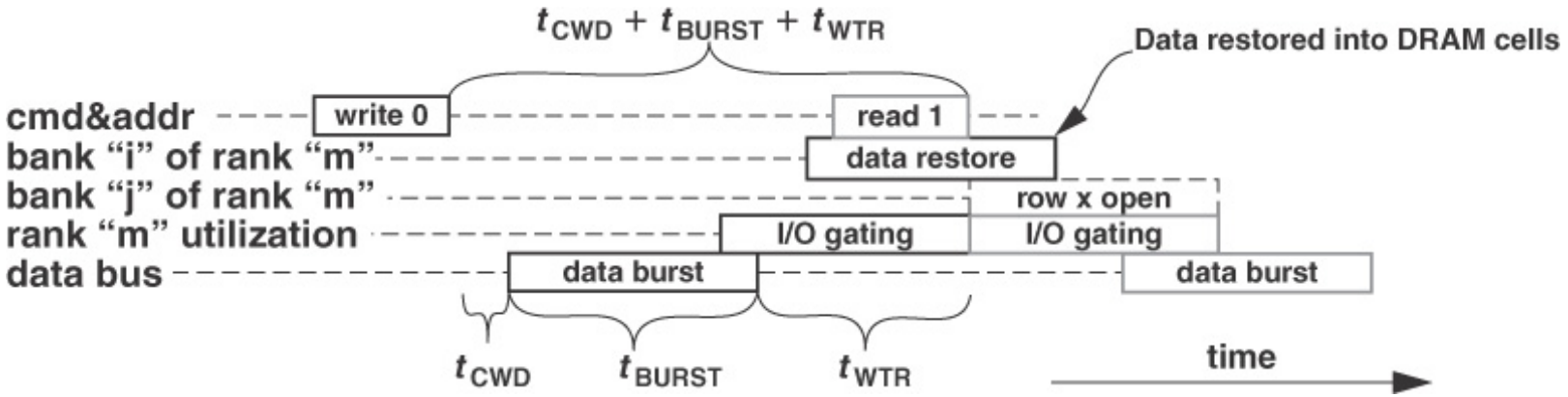
Consecutive WR Reqs: Bank Conflicts



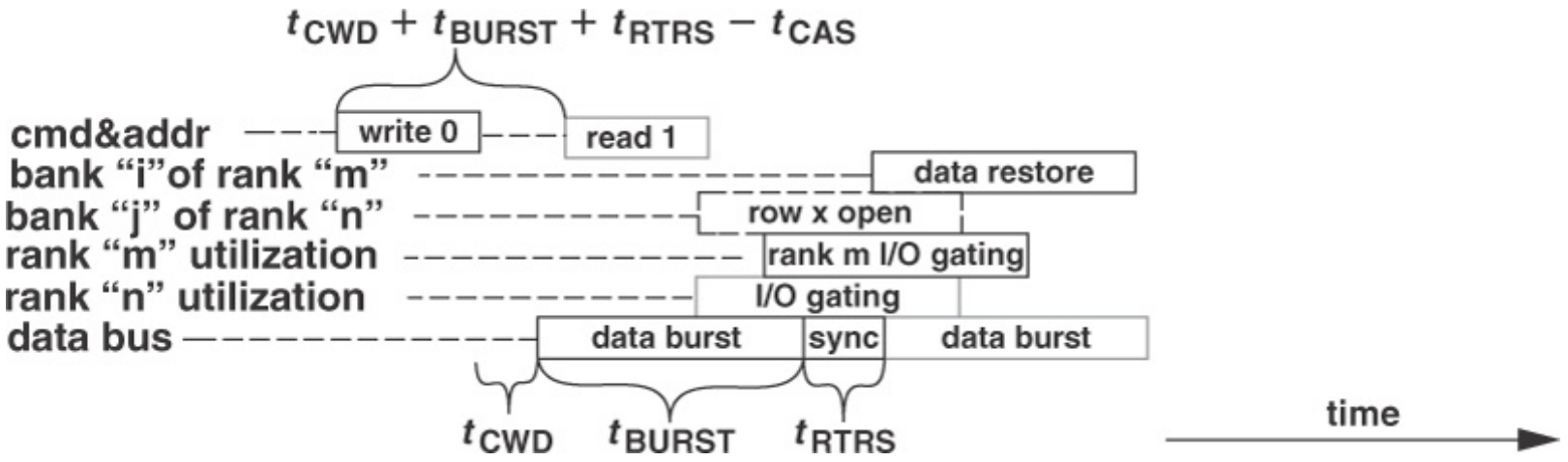
WR Req. Following Rd. Req: Open Banks



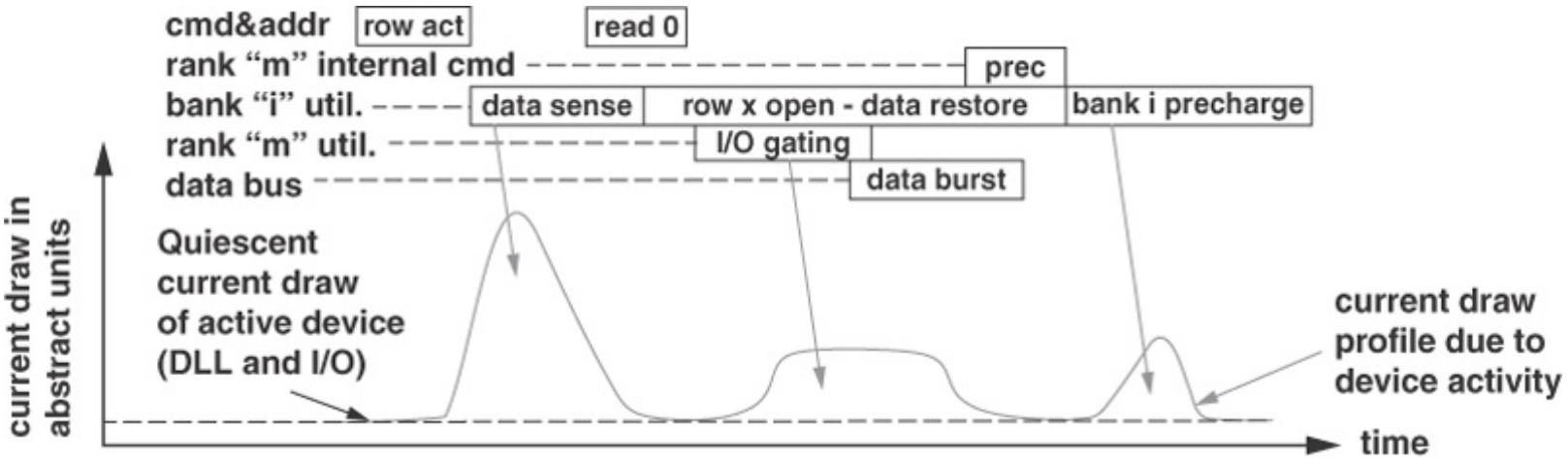
RD following WR: Same Rank, Open Banks



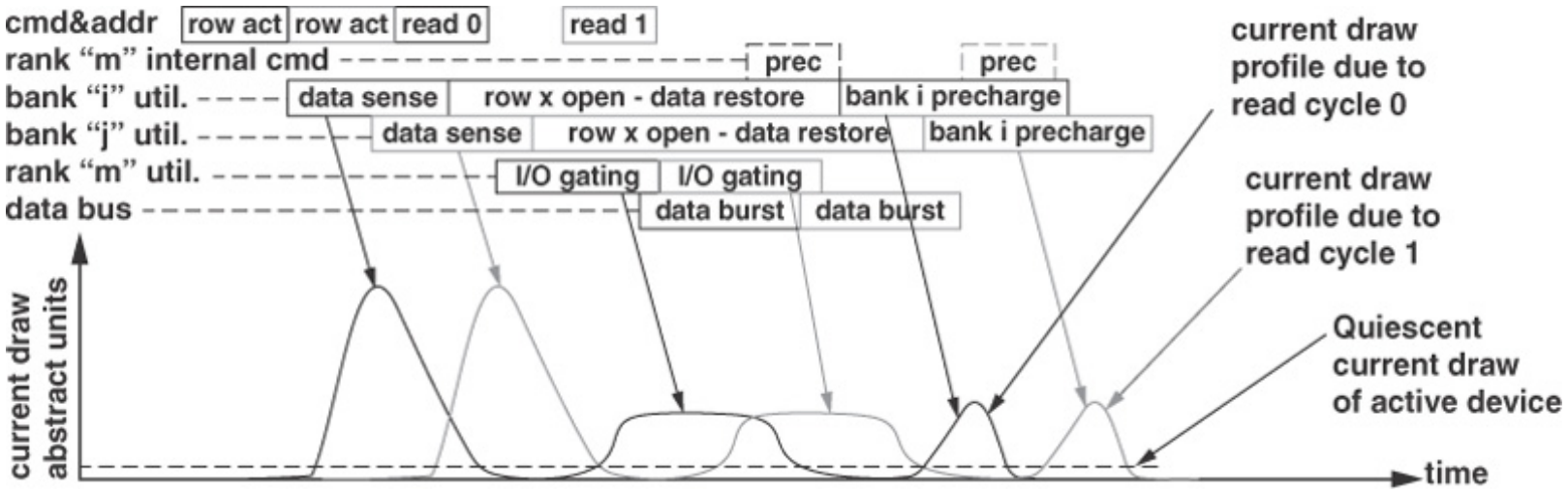
RD Following WR to Diff Ranks, Open Banks



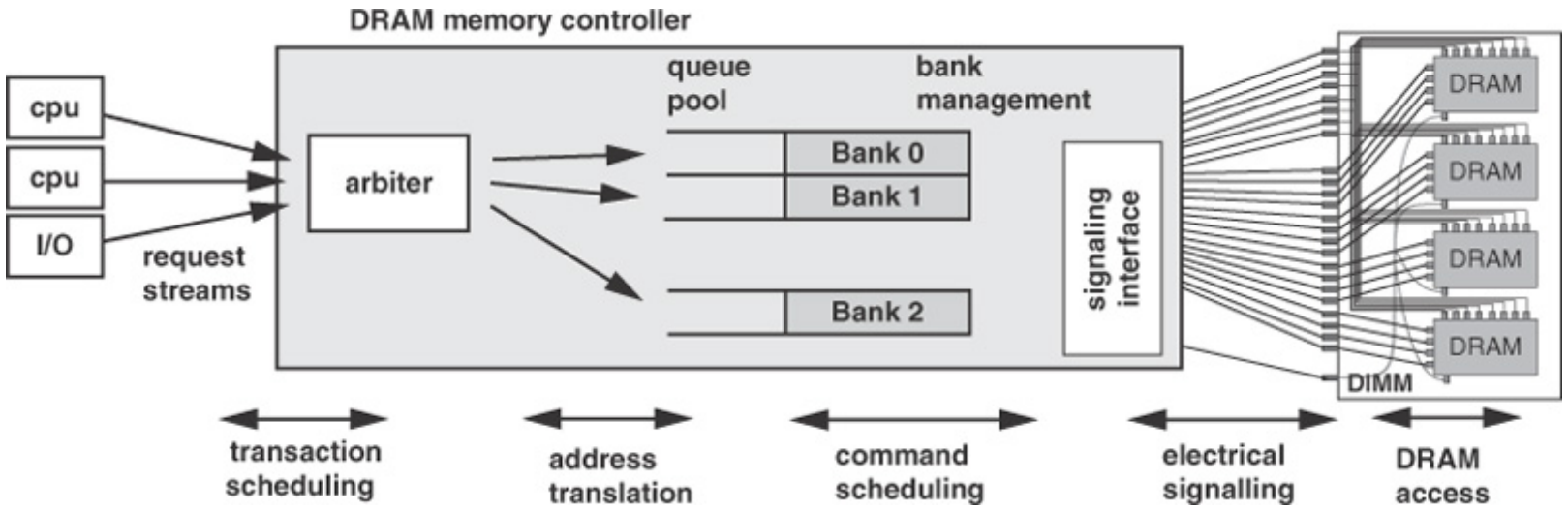
Current Profile of DRAM Read Cycle



Current Profile of 2 DRAM RD Cycles



Putting it all together: System View



In Modern Systems

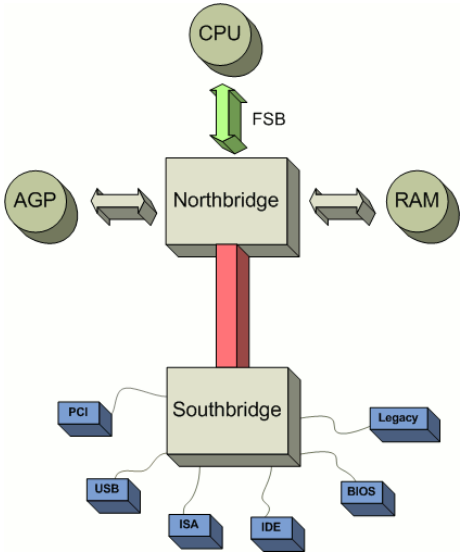
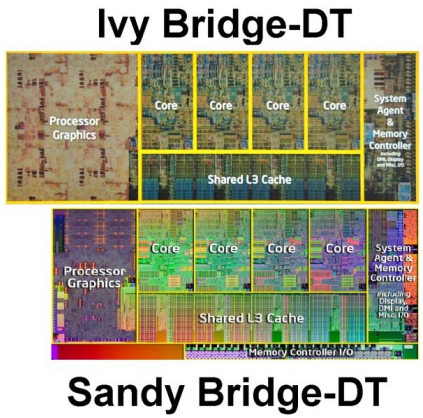
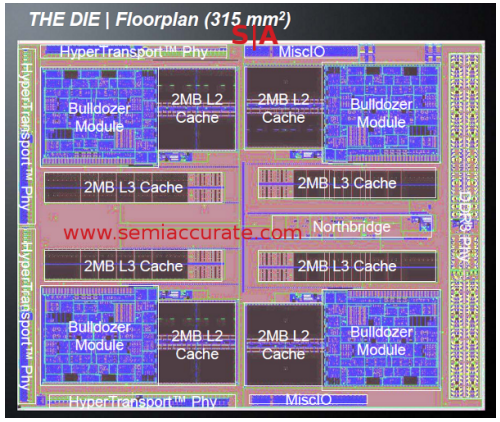
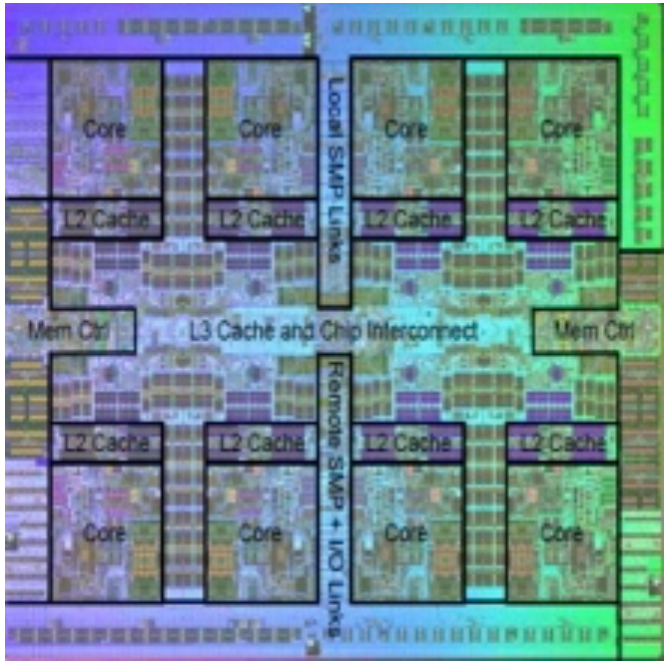


Image SRC wikipedia.org





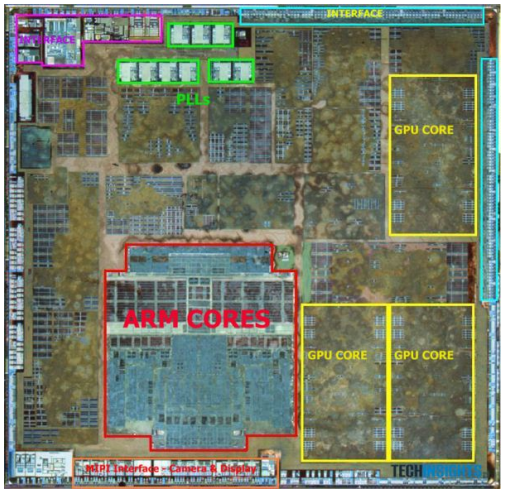


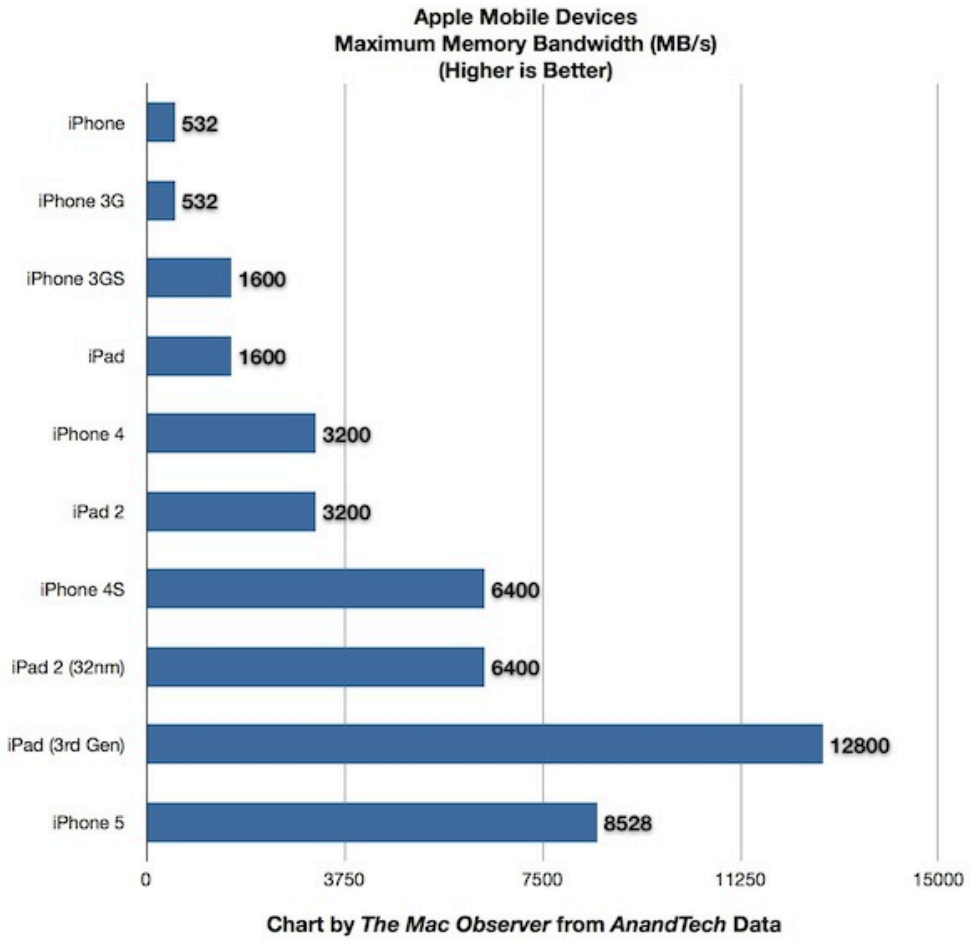
Bandwidth and Capacity

- Total System Capacity
  - # memory controllers *
  - # memory channels per memory controller *
  - # dimms per channel *
  - # ranks on dimm *
  - # drams per rank holding data

- Total System Bandwidth
  - # memory controllers *
  - # memory channels per memory controller *
  - Bit rate of dimm *
  - Width of data from dimm

A6 Die Photo


INTERFACE
PLLs
GPU CORE
ARM CORES
GPU CORE
GPU CORE


Apple Mobile Devices
Maximum Memory Bandwidth (MB/s)
(Higher is Better)
iPhone
532
iPhone 3G
532
iPhone 3GS
1600
iPad
1600
iPhone 4
3200
iPad 2
3200
iPhone 4S
6400
iPad 2 (32nm)
6400
iPad (3rd Gen)
12800
iPhone 5
8528
0
3750
7500
11250
15000
Chart by The Mac Observer from AnandTech Data

In class Design Exercise

- **Design the Instruction Issue Logic for a Out-of-order Processor.**

Summary

- Unit 1: Scaling, Design Process, Economics
- Unit 2: System Verilog for Design
- Unit 3: Design Building Blocks
  - Logic (Basic Units)
  - Memory (SRAM, DRAM)
  - Communication (Busses, Network-on-Chips)

- Next Unit
  - Design Validation