ECE 250 / CS250 Introduction to Computer Architecture

Main Memory

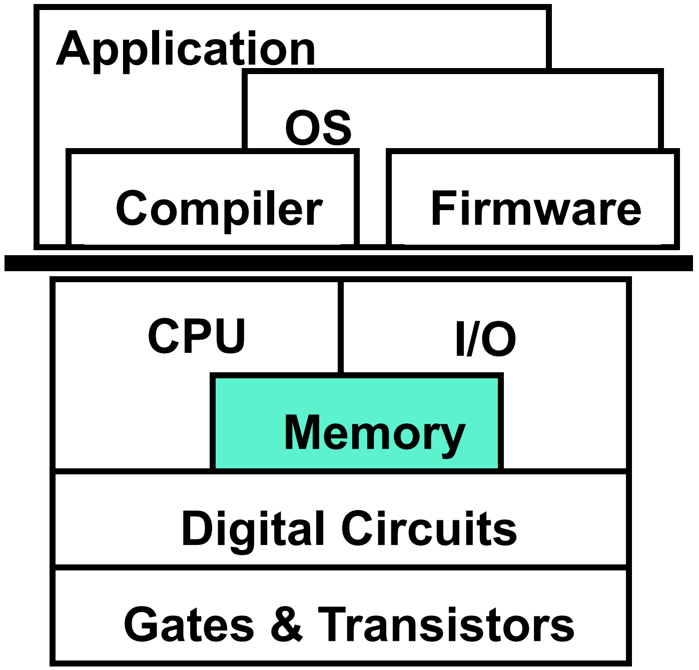
Benjamin C. Lee
Duke University

Slides from Daniel Sorin (Duke) and are derived from work by Amir Roth (Penn) and Alvy Lebeck (Duke)

Where We Are in This Course Right Now

- So far:
  - We know how to design a processor that can fetch, decode, and execute the instructions in an ISA
  - We can pipeline this processor
  - We understand how to design caches
- Now:
  - We learn how to implement main memory in DRAM
  - We learn about virtual memory
- Next:
  - We learn about the lowest level of storage (disks) and I/O

This Unit: Main Memory



- Memory hierarchy review
- DRAM technology
  - A few more transistors
  - Organization: two-level addressing
- Building a memory system
  - Bandwidth matching
  - Error correction
- Organizing a memory system
- Virtual memory
  - Address translation and page tables
  - A virtual memory hierarchy

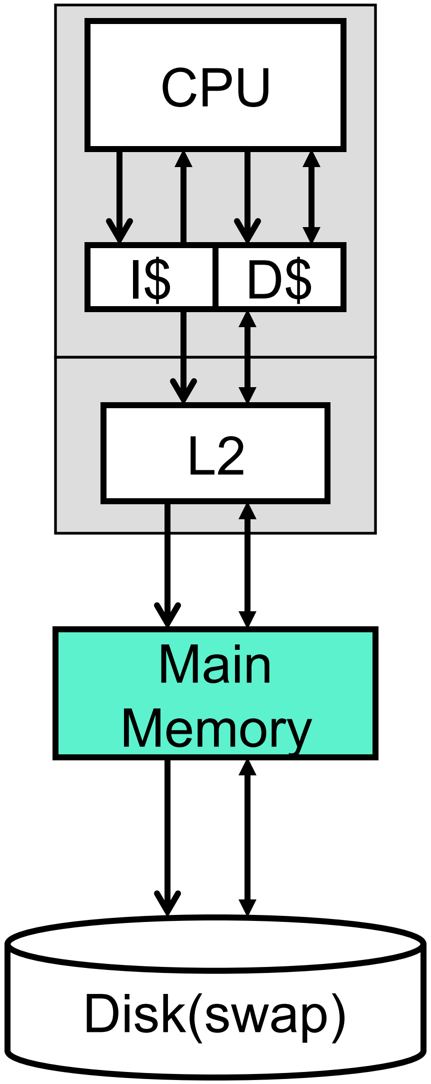
Readings

- Patterson and Hennessy
    - Still in Chapter 5

Memory Hierarchy Review

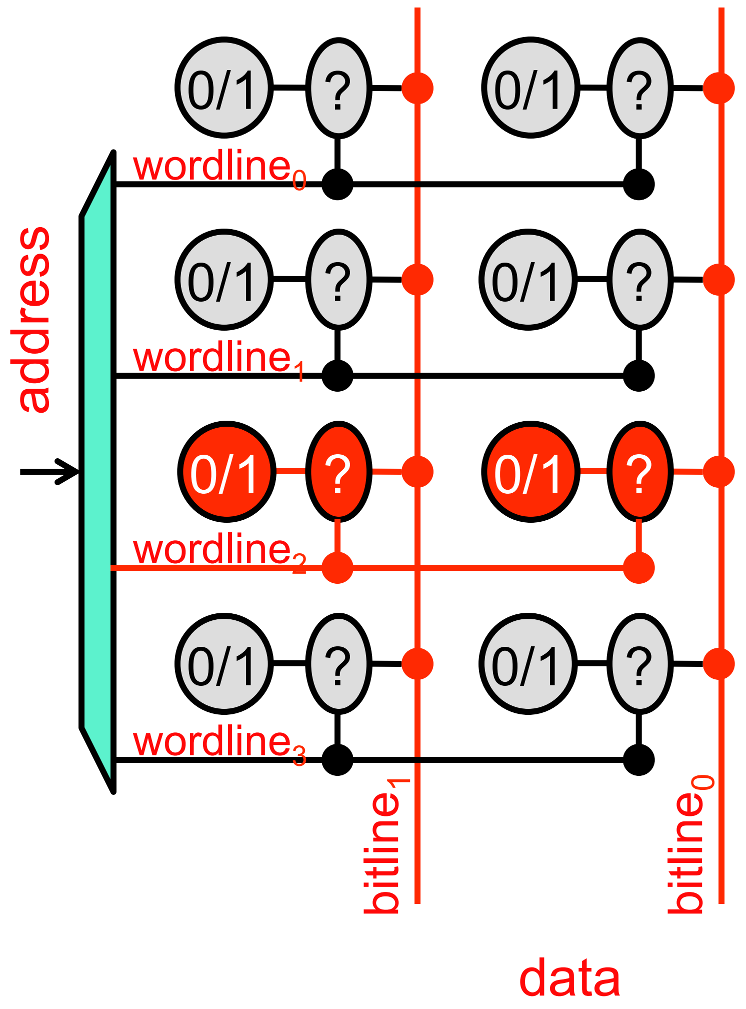
- Storage: registers, **memory**, disk
  - Memory is fundamental element (unlike caches or disk)
- Memory component performance
  - $\mathbf{t_{avg} = t_{hit} + \%_{miss} * t_{miss}}$
  - Can't get both low t_{hit} and $\%_{miss}$ in a single structure
- Memory hierarchy
  - Upper components: small, fast, expensive
  - Lower components: big, slow, cheap
  - t_{avg} of hierarchy is close to t_{hit} of upper (fastest) component
    - 10/90 rule: 90% of stuff found in fastest component
  - **Temporal/spatial locality**: automatic up-down data movement

Concrete Memory Hierarchy



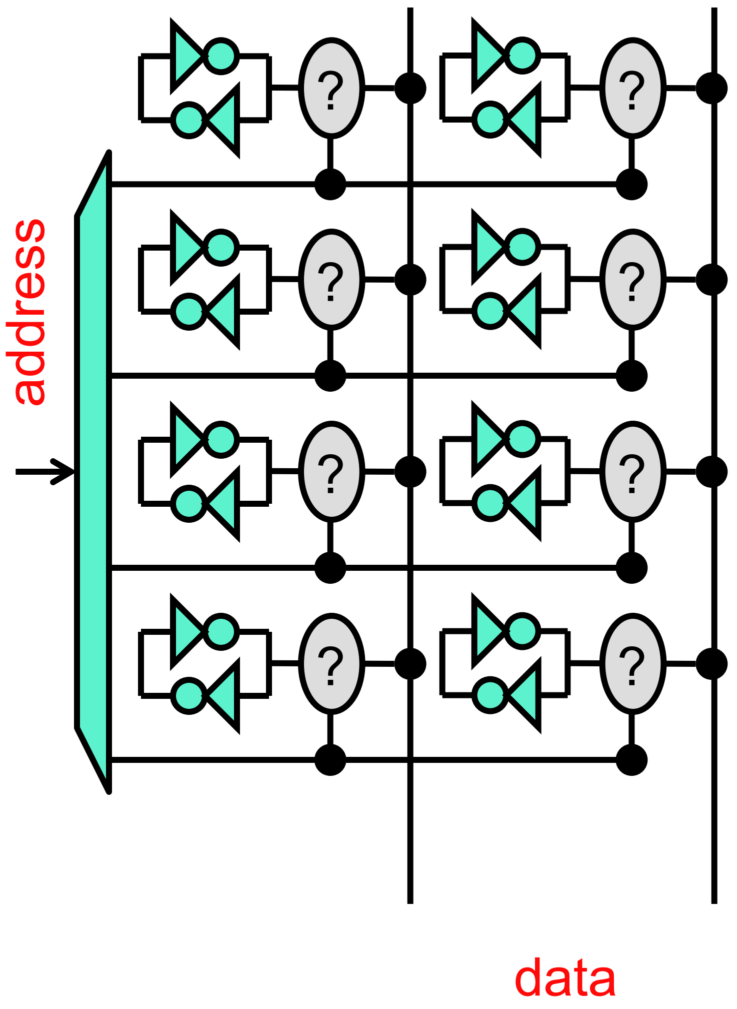
- 1st/2nd(/3rd) levels: caches (L1 I$, L1 D$, L2)
  - Made of SRAM
  - Managed in hardware
  - Previous unit of course
- Below caches level: **main memory**
  - Made of DRAM
  - Managed in software
  - This unit of course
- Below memory: disk (swap space)
  - Made of magnetic iron oxide disks
  - Managed in software
  - Next unit

RAM in General (SRAM and DRAM)



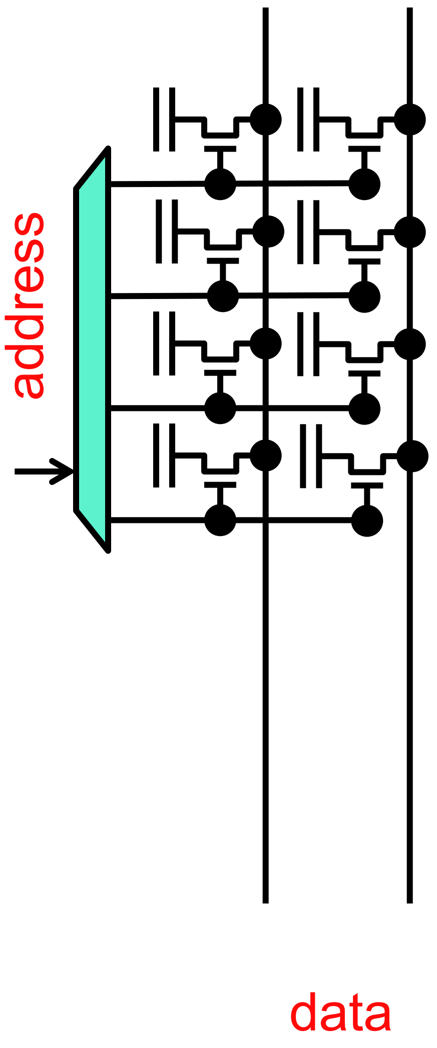
- RAM: large storage arrays
- Basic structure
  - MxN array of bits (M N-bit words)
    - This one is 4x2
  - Bits in word connected by **wordline**
  - Bits in position connected by **bitline**
- Operation
  - Address decodes into M wordlines
  - Assert wordline → word on bitlines
  - Bit/bitline connection → read/write
- Access latency
  - #ports * √#bits

SRAM



- **SRAM**: static RAM
  - Bits as cross-coupled inverters
  - Four transistors per bit
  - More transistors for ports
- **"Static"** means
  - Inverters connected to pwr/gnd
  - Bits naturally/continuously "refreshed"
  - Bit values never decay
- Designed for speed

DRAM



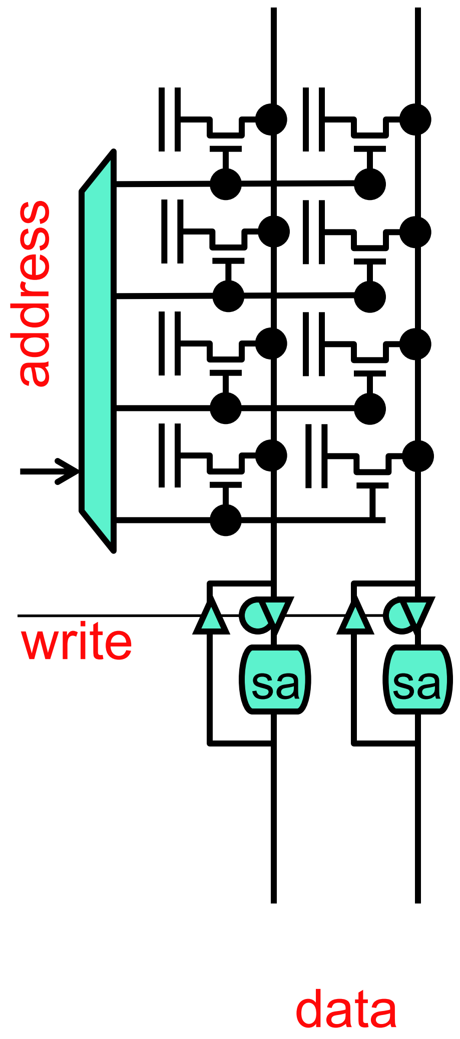
- **DRAM**: dynamic RAM
  - Bits as capacitors (if charge, bit=1)
  - Pass transistors as ports
  - One transistor per bit/port
- **"Dynamic"** means
  - Capacitors not connected to pwr/gnd
  - Stored charge decays over time
  - Must be explicitly refreshed
- Designed for density
  - Moore's Law …

Moore's Law (DRAM capacity)

Year	Capacity	$/MB	Access time
1980	64Kb	$1500	250ns
1988	4Mb	$50	120ns
1996	64Mb	$10	60ns
2004	1Gb	$0.5	35ns
2008	4Gb	~$0.15	20ns

- Commodity DRAM parameters
  - 16X increase in capacity every 8 years = 2X every 2 years
    - Not quite 2X every 18 months (Moore's Law) but still close

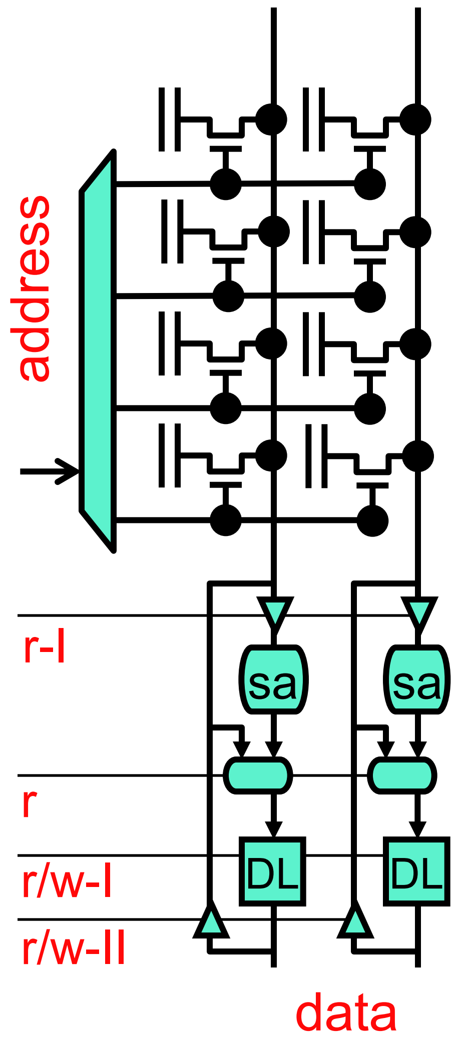
DRAM Operation I



- Read: similar to SRAM read
  - Phase I: pre-charge bitlines (to ~0.5V)
  - Phase II: decode address, enable wordline
    - Capacitor swings bitline voltage up (down)
    - Sense-amplifier interprets swing as 1 (0)
  - **Destructive read**: word bits now discharged
    - Unlike SRAM
- Write: similar to SRAM write
  - Phase I: decode address, enable wordline
  - Phase II: enable bitlines
    - High bitlines charge corresponding capacitors

– What about **leakage over time**?

DRAM Operation II

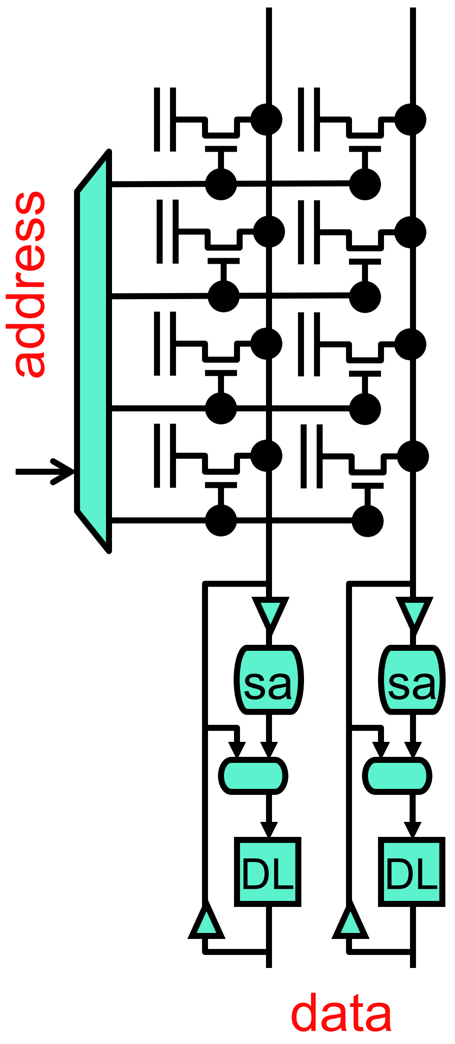


- Solution: add set of D-latches (**row buffer**)

- Read: two steps
  - Step I: read selected word into row buffer
  - Step IIA: read row buffer out to pins
  - Step IIB: write row buffer back to selected word
  - + Solves "destructive read" problem
- Write: two steps
  - Step IA: read selected word into row buffer
    - Deletes what was in that word before
  - Step IB: write data into row buffer
  - Step II: write row buffer back to selected word

+ Also helps to solve leakage problem …

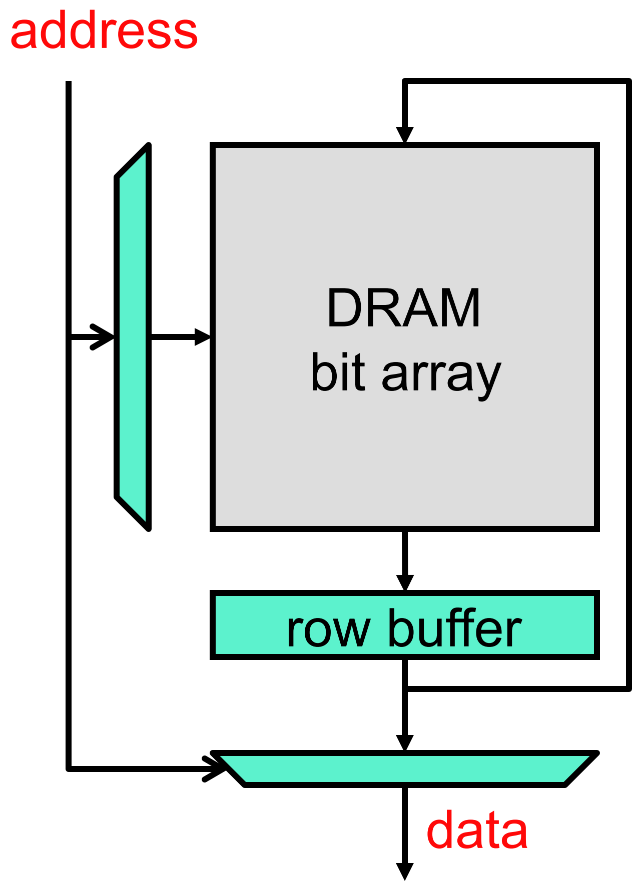
DRAM Refresh



- DRAM periodically refreshes all contents
  - Loops through all words
    - Reads word into row buffer
    - Writes row buffer back into DRAM array

  - 1–2% of DRAM time occupied by refresh

DRAM Parameters



- DRAM parameters
  - Large capacity: e.g., 1-4Gb
    - Arranged as square
    + Minimizes wire length
    + Maximizes refresh efficiency
  - Narrow data interface: 1–16 bit
    - Cheap packages → few bus pins
    - Pins are expensive
  - Narrow address interface: N/2 bits
    - 16Mb DRAM had a 12-bit address bus
    - How does that work?

Access Time and Cycle Time

- DRAM access slower than SRAM
  - More bits → longer wires
  - Buffered access with two-level addressing
  - SRAM access latency: 2–3ns
  - DRAM access latency: 20-35ns

- DRAM cycle time also longer than access time
  - **Cycle time**: time between start of consecutive accesses
  - SRAM: cycle time = access time
    - Begin second access as soon as first access finishes
  - DRAM: cycle time = 2 * access time
    - Why? Can't begin new access while DRAM is refreshing row

Brief History of DRAM

- DRAM (memory): a major force behind computer industry
  - Modern DRAM came with introduction of IC (1970)
  - Preceded by magnetic "core" memory (1950s)
    - Core more closely resembles today's disks than memory
    - "Core dump" is legacy terminology
  - And by mercury delay lines before that (ENIAC)
    - Re-circulating vibrations in mercury tubes

"the one single development that put computers on their feet was the invention of a reliable form of memory, namely the core memory… It's cost was reasonable, it was reliable, and because it was reliable it could in due course be made large"

Maurice Wilkes

Memoirs of a Computer Programmer, 1985

A Few Flavors of DRAM

- DRAM comes in several different varieties
  - Go to Dell.com and see what kinds you can get for your laptop
- SDRAM = synchronous DRAM
  - Fast, clocked DRAM technology
  - Very common now
  - Several flavors: DDR, DDR2, DDR3
- RDRAM = Rambus DRAM
  - Very fast, expensive DRAM
- GDRAM = graphics DRAM
  - GDDR

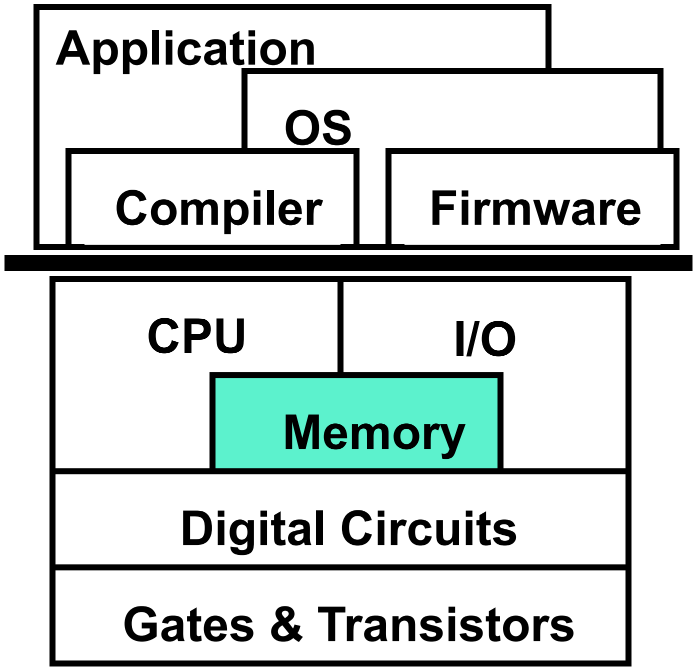
DRAM Packaging

- DIMM = dual inline memory module
  - E.g., 8 DRAM chips, each chip is 4 or 8 bits wide

DRAM: A Vast Topic

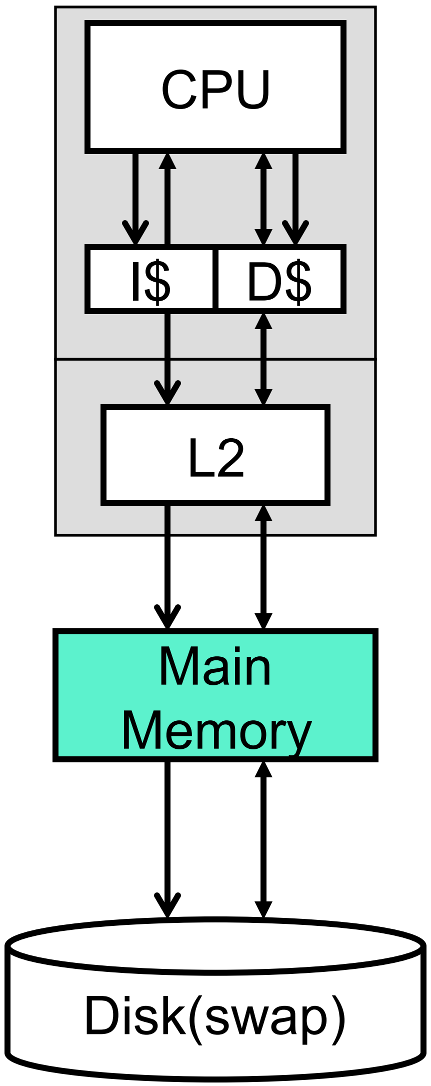
- Many flavors of DRAMs
  - DDR3 SDRAM, RDRAM, GDRAM, etc.
- Many ways to package them
  - SIMM, DIMM, FB-DIMM, etc.
- Many different parameters to characterize their timing
  - t_{RC}, t_{RAC}, t_{RCD}, t_{RAS}, etc.
- Many ways of using row buffer for "caching"
- Etc.
- There's at least one whole textbook on this topic!
  - And it has ~1K pages
- We could, but won't, spend rest of semester on DRAM

This Unit: Main Memory



- Memory hierarchy review
- DRAM technology
  - A few more transistors
  - Organization: two level addressing
- Building a memory system
  - Bandwidth matching
  - Error correction
- Organizing a memory system
- Virtual memory
  - Address translation and page tables
  - A virtual memory hierarchy

Building a Memory System



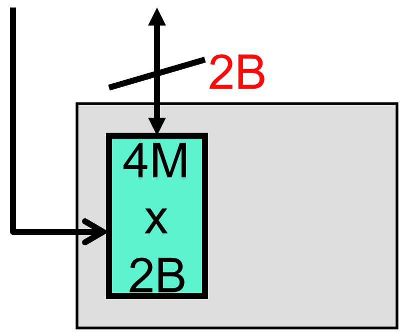
- How do we build an efficient main memory out of standard DRAM chips?
  - How many DRAM chips?
  - What width/speed (data) bus to use?
    - Assume separate address bus

An Example Memory System

- Parameters
  - 32-bit machine
  - L2 with 32B blocks (must pull 32B out of memory at a time)
  - 4Mx16b DRAMs, 20ns access time, 20ns refresh time
    - Each chip is 4Mx2B = 8 MB
  - 100MHz (10ns period) data bus
  - 100MHz, 32-bit address bus

- How many DRAM chips?
- How wide to make the data bus?

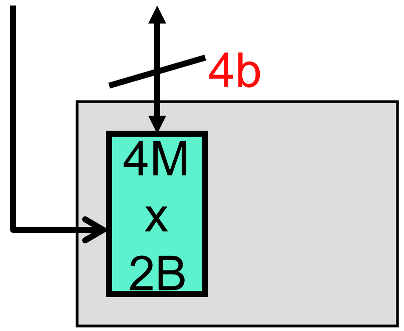
First Memory System Design



- 1 DRAM + 16b (=2B) bus
  - Access time: 630ns
    - Not including address
  - Cycle time: 640ns
    - DRAM ready to handle another miss
  - Observation: data bus idle 75% of time!
    - We have over-designed bus
    - Can we use a cheaper bus?

T (ns)	DRAM	Data Bus
10	**[31:30]**	
20	**[31:30]**	
30	**refresh**	**[31:30]**
40	**refresh**	
50	[29:28]	
60	[29:28]	
70	refresh	[29:28]
80	refresh	
…	…	…
600	refresh	
610	[1:0]	
620	[1:0]	
630	refresh	**[1:0]**
640	refresh	

Second Memory System Design



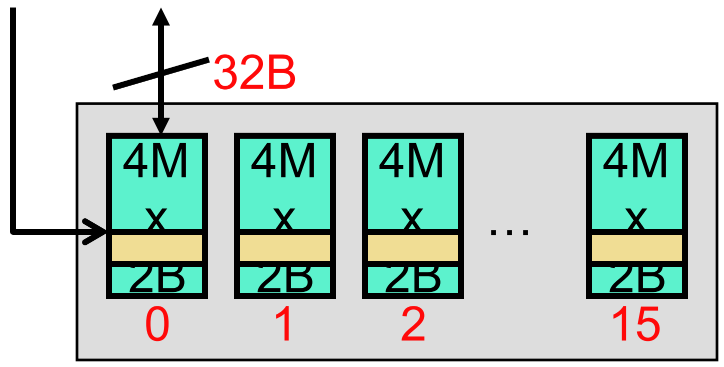
- 1 DRAM + **4b** bus
  - One DRAM chip, don't need 16b bus
  - DRAM: 2B / 40ns → 4b / 10ns
  - Balanced system → match bandwidths

  - Access time: 660ns (30ns longer=+4%)
  - Cycle time: 640ns (same as before)

  + Much cheaper!

T (ns)	DRAM	Bus
10	**[31:30]**	
20	**[31:30]**	
30	**refresh**	**[31H]**
40	**refresh**	**[31L]**
50	[29:28]	**[30H]**
60	[29:28]	**[30L]**
70	refresh	[29H]
80	refresh	[29L]
…	…	…
600	[1:0]	[2H]
610	[1:0]	[2L]
620	refresh	[1H]
640	refresh	[1L]
650		[0H]
660		**[0L]**

Third Memory System Design



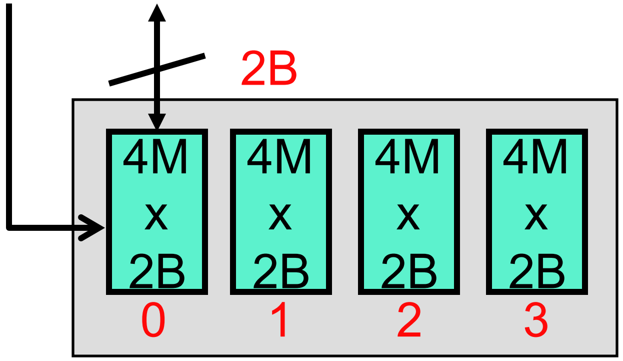
T (ns)	DRAM0	DRAM1	DRAM15	Bus
10	**[31:30]**	**[29:28]**	**[1:0]**	
20	**[31:30]**	**[29:28]**	**[1:0]**	
30	**refresh**	**refresh**	**refresh**	**[31:0]**
40	**refresh**	**refresh**	**refresh**	

- How fast can we go?
- 16 DRAM chips + 32B bus
  - **Stripe data across chips**
  - Byte M in chip (M/2)%16 (e.g., byte 38 is in chip 3)
  - Access time: 30ns
  - Cycle time: 40ns
  - – 32B bus is very expensive

Latency and Bandwidth

- In general, given bus parameters…
  - Find smallest number of chips that minimizes cycle time
  - Approach: match bandwidths between DRAMs and data bus
    - If they don't match, you're paying too much for the one with more bandwidth

Fourth Memory System Design



- 2B bus
  - Bus b/w: 2B/10ns
  - DRAM b/w: 2B/40ns
  - 4 DRAM chips
  - Access time: 180ns
  - Cycle time: 160ns

T (ns)	DRAM0	DRAM1	DRAM2	DRAM3	Bus
10	**[31:30]**	**[29:28]**	**[27:26]**	**[25:24]**	
20	**[31:30]**	**[29:28]**	**[27:26]**	**[25:24]**	
30	**refresh**	**refresh**	**refresh**	**refresh**	**[31:30]**
40	**refresh**	**refresh**	**refresh**	**refresh**	**[29:28]**
50	[23:22]	[21:20]	[19:18]	[17:16]	**[27:26]**
60	[23:22]	[21:20]	[19:18]	[17:16]	**[25:24]**
…	…	…	…	…	…
110	refresh	refresh	refresh	refresh	[15:14]
120	refresh	refresh	refresh	refresh	[13:12]
130	[7:6]	[5:4]	[3:2]	[1:0]	[11:10]
140	[7:6]	[5:4]	[3:2]	[1:0]	[9:8]
150	refresh	refresh	refresh	refresh	[7:6]
160	refresh	refresh	refresh	refresh	[5:4]
170					[3:2]
180					**[1:0]**

Memory Access and Clock Frequency

- Nominal **clock frequency** applies to CPU and caches
  - Memory bus has its own clock, typically much slower
  - SDRAM operates on bus clock

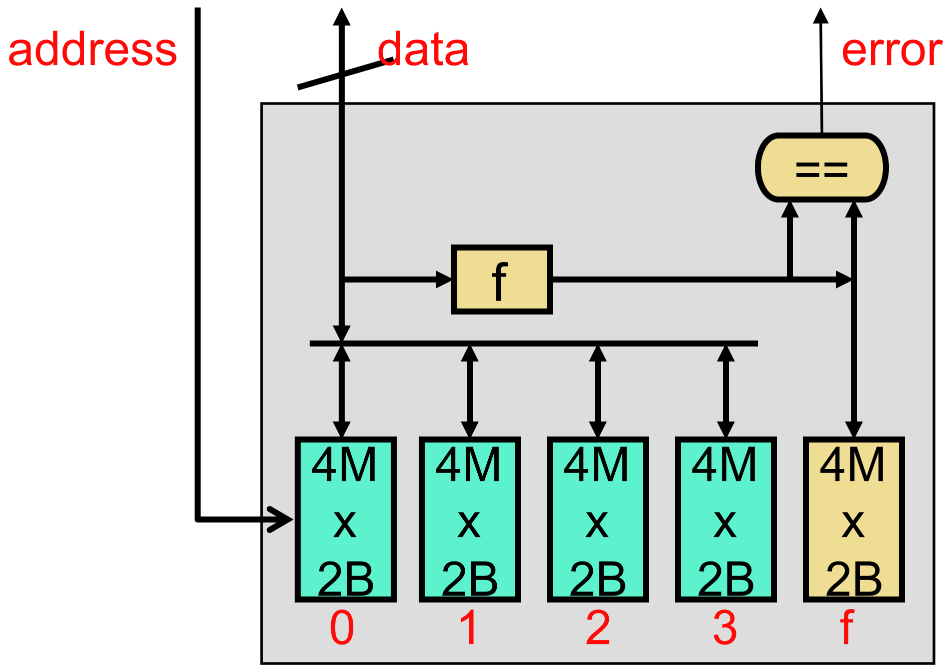
- Another reason why processor clock frequency isn't perfect performance metric
  - Clock frequency increases don't reduce memory or bus latency
  - May make misses come out faster
    - At some point memory bandwidth may become a **bottleneck**
    - Further increases in (core) clock speed won't help at all

Error Detection and Correction

- One last thing about DRAM technology: **errors**
  - DRAM fails at a higher rate than SRAM or CPU logic
    - Capacitor wear
    - Bit flips from energetic α-particle strikes
    - Many more bits
  - Modern DRAM systems: built-in error detection/correction

- **Key idea: checksum-style redundancy**
  - Main DRAM chips store data, additional chips store f(data)
    - $|f(data)| < |data|$
  - On read: re-compute f(data), compare with stored f(data)
    - Different ? Error…
  - Option I (**detect**): kill program
  - Option II (**correct**): enough information to fix error? fix and go on

Error Detection and Correction



- Error detection/correction schemes distinguished by…
  - How many (simultaneous) errors they can detect
  - How many (simultaneous) errors they can correct

Error Detection Example: Parity

- **Parity**: simplest scheme
  - $f(\text{data}_{N-1\ldots0}) = \text{XOR}(\text{data}_{N-1}, \ldots, \text{data}_1, \text{data}_0)$
  - + Single-error detect: detects a single bit flip (common case)
    - Will miss two simultaneous bit flips…
    - But what are the odds of that happening?
  - – Zero-error correct: no way to tell which bit flipped

– Many other schemes exist for detecting/correcting errors

- – Take ECE 254 (Fault Tolerant Computing) for more info

Memory Organization

- So data is striped across DRAM chips
- But how is it organized?
  - Block size?
  - Associativity?
  - Replacement policy?
  - Write-back vs. write-thru?
  - Write-allocate vs. write-non-allocate?
  - Write buffer?
  - Optimizations: victim buffer, prefetching, anything else?

Low $\%_{miss}$ At All Costs

- For a memory component: t_{hit} vs. $\%_{miss}$ tradeoff

- Upper components (I\$, D\$) emphasize low t_{hit}
  - Frequent access → minimal t_{hit} important
  - t_{miss} is not bad → minimal $\%_{miss}$ less important
  - Low capacity/associativity/block-size, write-back or write-through

- Moving down (L2) emphasis turns to $\%_{miss}$
  - Infrequent access → minimal t_{hit} less important
  - t_{miss} is bad → minimal $\%_{miss}$ important
  - High capacity/associativity/block size, write-back

- For memory, emphasis entirely on $\%_{miss}$
  - t_{miss} is disk access time (measured in ms, not ns)

Typical Memory Organization Parameters

Parameter	**I$/D$**	**L2**	**Main Memory**
t_{hit}	1-2ns	10ns	**30ns**
t_{miss}	**10ns**	**30ns**	**10ms (10M ns)**
Capacity	8–64KB	128KB–2MB	**512MB–8GB**
Block size	16–32B	32–256B	**8–64KB pages**
Associativity	1–4	4–16	**Full**
Replacement Policy	NMRU	NMRU	**working set**
Write-through?	Sometimes	No	**No**
Write-allocate?	Sometimes	Yes	**Yes**
Write buffer?	Yes	Yes	**No**
Victim buffer?	Yes	No	**No**
Prefetching?	Sometimes	Yes	**Sometimes**

One Last Gotcha

- On a 32-bit architecture, there are 2^{32} byte addresses
  - Requires 4 GB of memory
  - But not everyone buys machines with 4 GB of memory
  - And what about 64-bit architectures?
- Let’s take a step back…
  - Interface between computer architecture and operating system?