

## ECE 152 Introduction to Computer Architecture

Main Memory and Virtual Memory  
Copyright 2008 Daniel J. Sorin  
Duke University

Slides are derived from work by  
Amir Roth (Penn)  
Spring 2008

1

## Where We Are in This Course Right Now

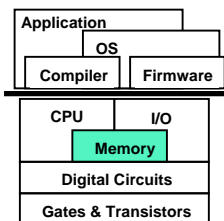
- So far:
  - We know how to design a processor that can fetch, decode, and execute the instructions in an ISA
  - We can pipeline this processor
  - We understand how to design caches
- Now:
  - We learn how to implement main memory in DRAM
  - We learn about virtual memory
- Next:
  - We learn about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

© 2008 Daniel J. Sorin from Roth

ECE 152

2

## 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

© 2008 Daniel J. Sorin from Roth

ECE 152

3

## Readings

- Patterson and Hennessy
  - Chapter 7 (including 7.4 this time)
  - Appendix B.5 (section about DRAM)

© 2008 Daniel J. Sorin from Roth

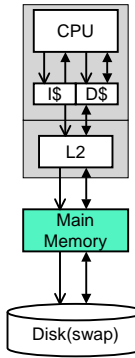
ECE 152

4

## Memory Hierarchy Review

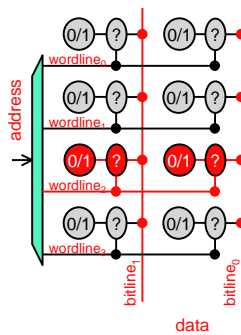
- Storage: registers, **memory**, disk
  - Memory is fundamental element (unlike caches or disk)
- Memory component performance
  - $t_{avg} = t_{hit} + \%_{miss} * t_{miss}$
  - Can't get both low  $t_{hit}$  and  $\%_{miss}$  in a single structure
- Memory hierarchy
  - Upper components: small, fast, 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



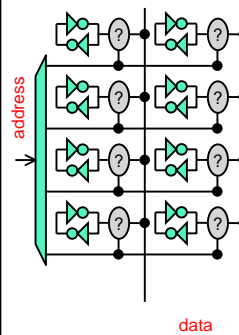
- 1<sup>st</sup>/2<sup>nd</sup>/(3<sup>rd</sup>) levels: caches (L1 I\$, L1 D\$, L2)
  - Made of SRAM
  - Managed in hardware
  - Previous unit of course
- Below caches level: **main memory**
  - Made of DRAM
  - Managed in software
  - This unit of course
- Below memory: disk (swap space)
  - Made of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

## RAM in General (SRAM and DRAM)



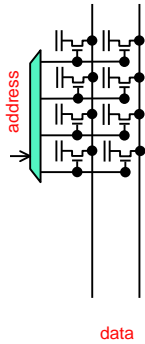
- RAM: large storage arrays
- Basic structure
  - MxN array of bits (M N-bit words)
    - This one is 4x2
  - Bits in word connected by **wordline**
  - Bits in position connected by **bitline**
- Operation
  - Address decodes into M wordlines
  - Assert wordline → word on bitlines
  - Bit/bitline connection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



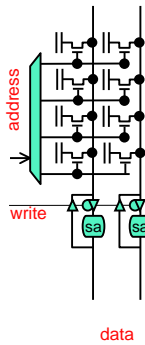
- **DRAM:** dynamic RAM
  - Bits as capacitors (if charge, bit=1)
  - Pass transistors as ports
  - One transistor per bit/port
- **"Dynamic"** means
  - Capacitors not connected to pwr/gnd
  - Stored charge decays over time
  - Must be explicitly refreshed
- Designed for density
  - Moore's Law ...

## 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

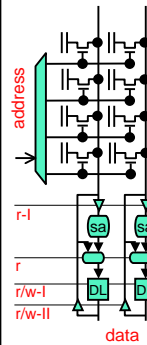
- Commodity DRAM parameters
  - 16X increase in capacity every 8 years = 2X every 2 years
  - Not quite 2X every 18 months (Moore's Law) but still close

## DRAM Operation I



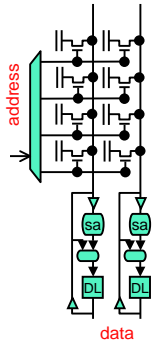
- Read: similar to SRAM read
    - Phase I: pre-charge bitlines to 0.5V
    - Phase II: decode address, enable wordline
      - Capacitor swings bitline voltage up (down)
      - Sense-amplifier interprets swing as 1 (0)
    - **Destructive read:** word bits now discharged
    - Unlike SRAM
  - Write: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



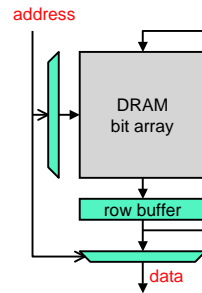
- Solution: add set of D-latches (**row buffer**)
  - Read: two steps
    - Step I: read selected word into row buffer
    - Step IIA: read row buffer out to pins
    - Step IIB: write row buffer back to selected word + Solves "destructive read" problem
  - Write: two steps
    - Step IA: read selected 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



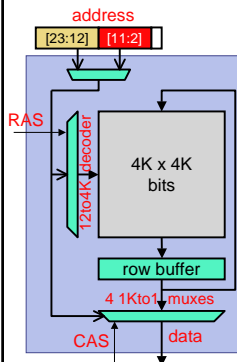
- DRAM periodically refreshes all contents
  - Loops through all words
    - Reads word into row buffer
    - Writes row buffer back into DRAM array
- 1–2% of DRAM time occupied by refresh

## 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?

## Two-Level Addressing



- **Two-level addressing**
  - Row decoder & column muxes share address lines
  - Two strobes (RAS, CAS) signal which part of address currently on bus
- Asynchronous access (no clock)
  - Level 1: RAS high
    - Upper address bits on address bus
    - Read row into row buffer
  - Level 2: CAS high
    - Lower address bits on address bus
    - Mux row buffer onto data bus

## Access Time and Cycle Time

- DRAM access much 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 DRAMmi

## 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, 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