Computer Science 146
Computer Architecture

Fall 2019

Harvard University

Instructor: Prof. David Brooks
dbrooks@eecs.harvard.edu

Lecture 17: Main Memory

Course Outline Revisited

W1	Feb 4		Introduction	Ch. 1
W2	Feb 9/11	Measuring Perf.	ISA Design	Ch. 2, A.1-3
W3	Feb 16/18	Holiday/No Class	Basic Pipelining	A.4-11, Ch3
W4	Feb 23/25	Multicycle/Scoreboard	Tomasulo's Algorithm	Ch 3
W5	Mar 1/3	Branch Pred./Fetch	Mult. Issue/Speculation	Ch 3
W6	Mar 8/10	Processor Case Studies	Static Issue	Ch 3/4
W7	Mar 15/17	Static ILP vs. HW ILP	IA64 Study/Review	Ch 4
W8	Mar 22/24	IA64 Study/Review	Midterm	
W9	Mar 29/31	Spring Break		
W10	Apr 5/7	Caches	Caches	Ch 5
W11	Apr 12/14	Caches	Main Memory	Ch 5
W12	Apr 19/21	Virtual Memory	Shared Memory MPs	Papers
W13	Apr 26/28	Multithreading (SMT/MP)	Storage, I/O, Clusters	Ch 6/7
W14	May 3/5	Security Processors	Network/GPU Processors	Papers

Lecture Outline

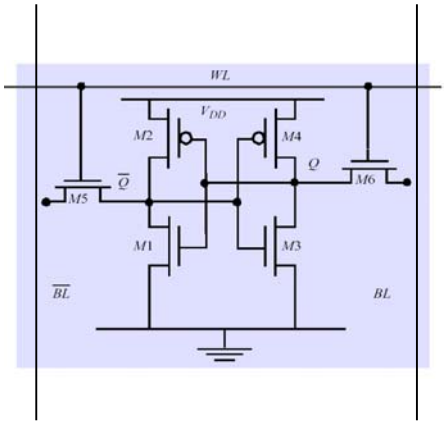
- Main Memory

Main Memory Background

- Random Access Memory
- Different flavors at different levels
  - Physical Makeup (CMOS, DRAM)
  - Low Level Architectures (FPM,EDO,SDRAM,RAMBUS)
- Cache uses *SRAM*: Static Random Access Memory
  - No refresh (6 transistors/bit vs. 1 transistor
    Size: DRAM/SRAM - *4-8x*,
    Cost and Cycle time: SRAM/DRAM - *8-16x*
- Main Memory is *DRAM*: Dynamic Random Access Memory
  - Dynamic since needs to be refreshed periodically (8 ms, 1% time)
  - Addresses divided into 2 halves (Memory as a 2D matrix):
    - *RAS* or *Row Access Strobe*
    - *CAS* or *Column Access Strobe*

Static RAM (SRAM)

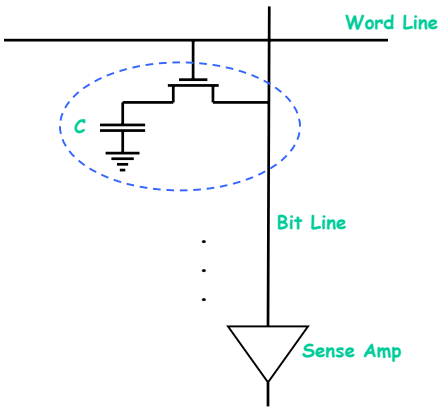
- Six transistors in cross connected fashion
  - Provides regular AND inverted outputs
  - Implemented in CMOS process



Single Port 6-T SRAM Cell

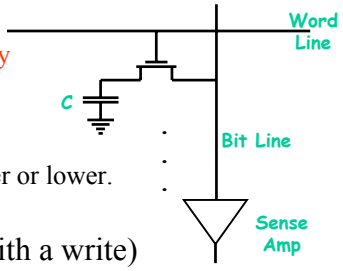
Dynamic RAM

- SRAM cells exhibit high speed/poor density
- DRAM: simple transistor/capacitor pairs in high density form



DRAM Operations

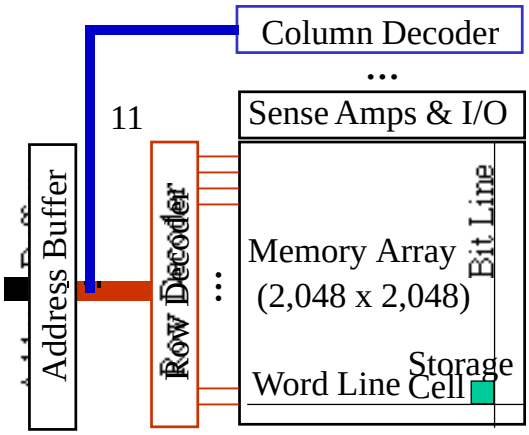
- Write
  - Charge bitline HIGH or LOW and set wordline HIGH
- Read
  - Bit line is precharged to a voltage halfway between HIGH and LOW, and then the word line is set HIGH.
  - Depending on the charge in the cap, the precharged bitline is pulled slightly higher or lower.
  - Sense Amp Detects change



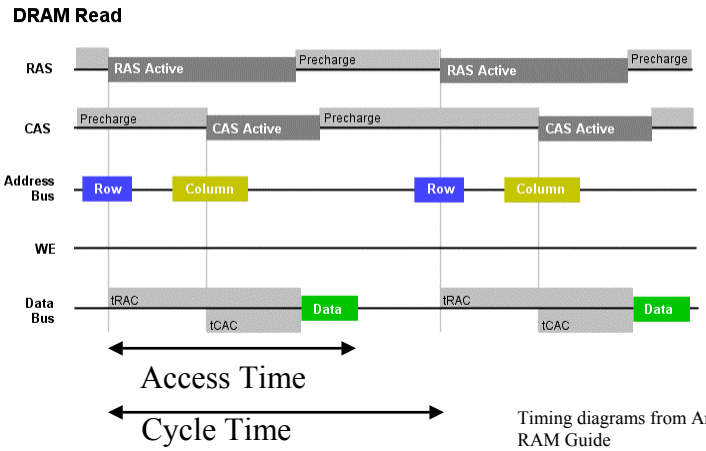
- Reads are destructive (Must follow with a write)
- Must refresh capacitor every so often
- Access Time = Time to Read
- Cycle Time = Time between Reads

DRAM logical organization

- Square Row/Column Matrix
- Multiplex Address Lines to save pins
- Internal Row Buffer
- Put Row Address on Lines
- Set RAS
- Read row into row buffer
- Put Column Adddress on Lines
- Set CAS
- Read Column bits out of row buffer

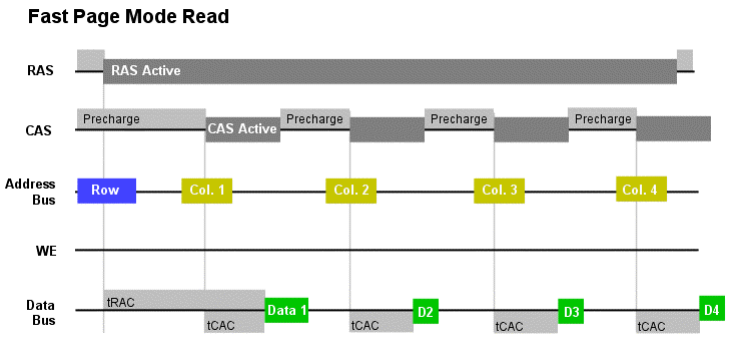


Vanilla DRAM Read

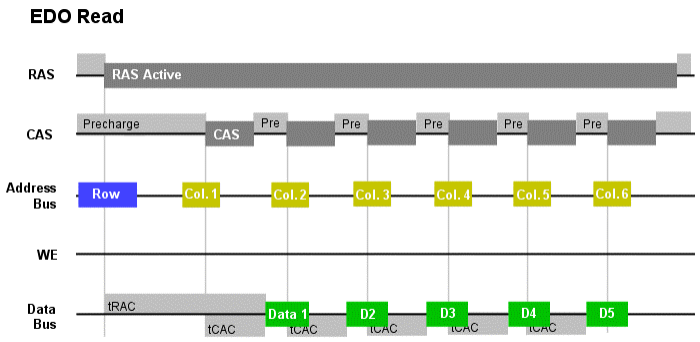


Timing diagrams from Ars Technica RAM Guide

Fast Page DRAM

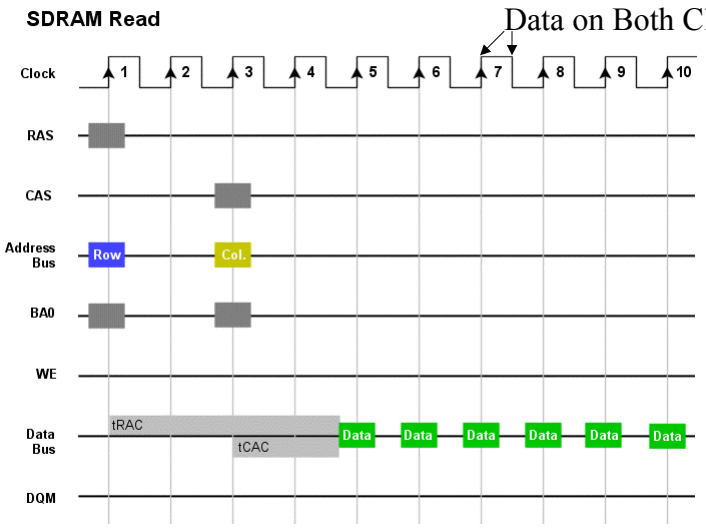


Extended Data Out (EDO) DRAM



Synchronous DRAM

DDR SDRAM: Transmit Data on Both Clock Edges

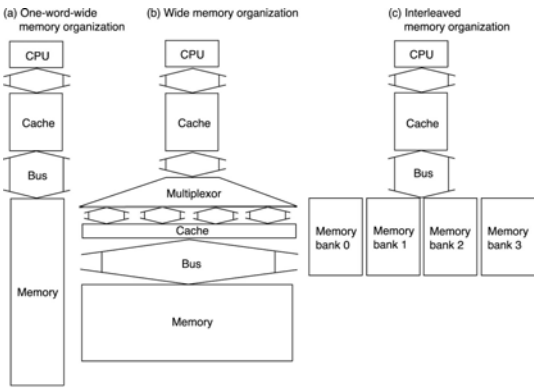


Comparison with SRAM

- By it's nature, DRAM isn't built for speed
  - Response times dependent on capacitive circuit properties which get worse as density increases
- DRAM process isn't easy to integrate into standard CMOS process
- SRAM:
  - Optimized for speed (8x - 16x DRAM), not density
  - Bits not erased on read
    - No refresh, access time = cycle time

Main Memory Organizations

- *Simple*:
  - CPU, Cache, Bus, Memory same width (32 or 64 bits)
- *Wide*:
  - CPU/Mux 1 word; Mux/Cache, Bus, Memory N words (Alpha: 64 bits & 256 bits; UtraSPARC 512)
- *Interleaved*:
  - CPU, Cache, Bus 1 word: Memory N Modules (4 Modules); example is *word interleaved*



Main Memory Configurations

- Simple Main Memory
  - 32-bit DRAM (1 word of data at a time)
  - Access time: 2 cycles (A)
  - Transfer time: 1 cycle (T)
  - Cycle Time: 4 cycles (B = cycle time – access time)
  - Miss penalty for a 4-word block?

Simple Main Memory

Cycle	Addr	Mem	steady
1	12	A	*
2		A	*
3		T/B	*
4		B	*
5	13	A	*
6		A	*
7		T/B	*
8		B	*
9	14	A	*
10		A	*
11		T/B	*
12		B	*
13	15	A	*
14		A	*
15		T/B	*
16		B	*

- 4 word access = 15 cycles
- 4-word cycle = 16 cycles

- How to improve?
  - Lower latency?
    - A,B,T are fixed
  - Higher bandwidth?

Bandwidth: Wider DRAMs

Cycle	Addr	Mem	steady
1	12	A	*
2		A	*
3		T/B	*
4		B	*
5	14	A	*
6		A	*
7		T/B	*
8		B	*

- 64-bit DRAM instead
- 4 word access = 7 cycles
- 4-word cycle = 8 cycles
- 64-bit buses are more expensive (Pentium vs. 486)

Bandwidth: Interleaving/Banking

- Use Multiple DRAMs, exploit their aggregate bandwidth
  - Each DRAM is called a bank
  - M 32-bit banks
  - Word A in bank (A % M) at (A div M)
  - Simple interleaving: banks share address lines

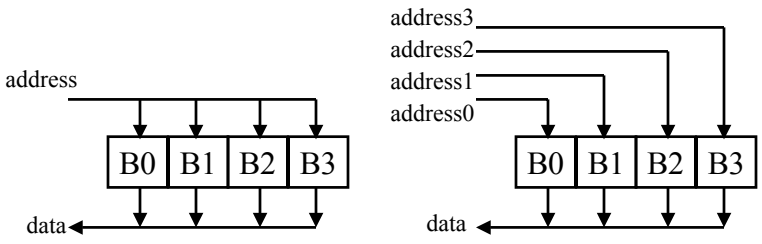
Simple Interleaving

Cycle	Addr	Bank0	Bank1	Bank2	Bank3	steady
1	12	A	A	A	A	
2		A	A	A	A	
3		T/B	B	B	B	*
4		B	T/B	B	B	*
5				T		*
6					T	*

- 4-word access = 6-cycles
- 4-word cycle = 4-cycles
  - Can start a new access in cycle 5
  - Overlap access with transfer (and still use a 32-bit bus!)

Complex Interleaving

- Simple interleaving: banks share address lines
- Complex interleaving: banks are independent
  - More expensive (separate address lines for each bank)



Complex Interleaving

Cycle	Addr	Bank0	Bank1	Bank2	Bank3	steady
1	12	A				
2	13	A	A			
3	14	T/B	A	A		*
4	15	B	T/B	A	A	*
5			B	T/B	A	*
6				B	T/B	*
7					B	

- 4-word access = 6-cycles
- 4-word cycle = 4-cycles
  - Same as simple interleaving

Simple Interleaving (Non-Sequential)

Cycle	Addr	Bank0	Bank1	Bank2	Bank3	steady
1	12(15)	A	A	A	A	*
2		A	A	A	A	*
3		T/B	B	B	B	*
4		B	B	B	T/B	*
5	18	A	A	A	A	*
6		A	A	A	A	*
7		B	B	T/B	B	*
8		B	B	B	B	*
9	21	A	A	A	A	*
10		A	A	A	A	*
11		B	T/B	B	B	*
12		B	B	B	B	*

- Non-sequential access, e.g. stride = 3
- 4-word access = 4-word cycle = 12-cycles

Complex Interleaving (Non-Sequential)

Cycle	Addr	Bank0	Bank1	Bank2	Bank3	steady
1	12	A				*
2	15	A			A	*
3	18	T/B		A	A	*
4	21	B	A	A	T/B	*
5			A	T/B	B	
6			T/B	B		

- 4-word access = 6-cycles
- 4-word cycle = 4-cycles
- DMA (I/O), Multiprocessors are non-sequential
- Want more banks than words in a cache line
  - Multiple cache misses in parallel (non-blocking caches)

Interleaving Problem

Cycle	Addr	Bank0	Bank1	Bank2	Bank3	steady
1	12	A				*
2		A				*
3		T/B				*
4		B				*
5	20	A				*
6		A				*
7		T/B				*
8		B				*

- Powers of 2 strides are a problem – all addresses, same bank
- 4-word access = 15 cycles, 4-word cycle = 16 cycle
- Solution: Use prime number of banks (e.g. 17)

Avoiding Bank Conflicts

- Lots of banks

```
int x[256][512];
    for (j = 0; j < 512; j = j+1)
         for (i = 0; i < 256; i = i+1)
              x[i][j] = 2 * x[i][j];
```

- Even with 128 banks, since 512 is multiple of 128, conflict on word accesses
- SW: loop interchange or declaring array not power of 2 ("array padding")
- HW: Add more Banks, Add Prime number of banks
  - bank number = address mod number of banks
  - address within bank = address / number of words in bank
  - modulo & divide per memory access with prime no. banks?
  - address within bank = address mod number words in bank
  - bank number? easy if 2^N words per bank

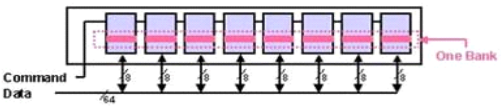
Independent Memory Banks

- How many banks?

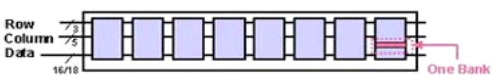
  number banks ≥ number clocks to access word in bank

  - For sequential accesses, otherwise will return to original bank before it has next word ready
- Increasing DRAM => fewer chips => less banks

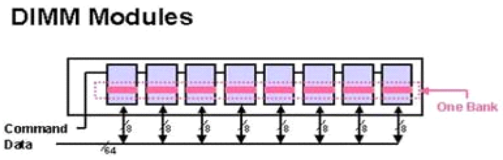
DIMM Modules 4 banks/chip * 1 rank = 4 total banks

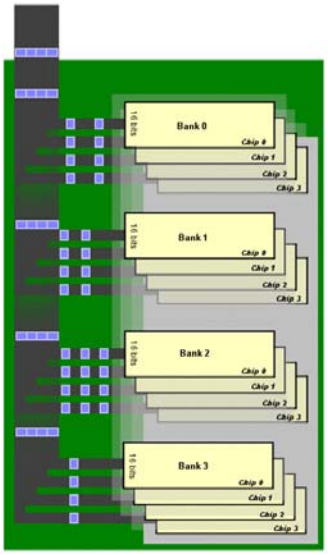


RIMM Modules 16 banks/chip * 8 chips = 128 banks



Independent Memory Banks

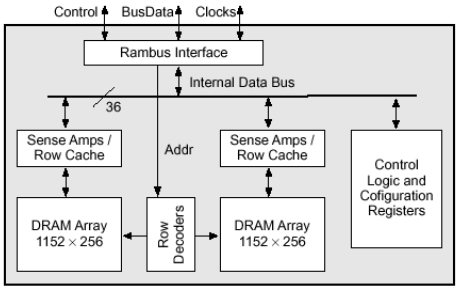




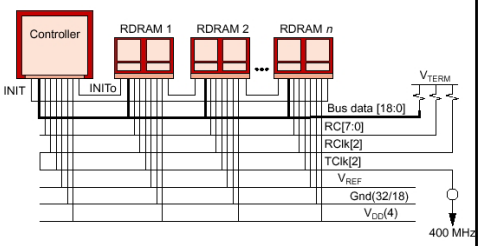
- DIMM (Dual-Inline Memory Module) Configuration
- Banking occurs at the chip, module, and system levels
- 1 Rank of devices responds to each access
  - All devices respond similarly
- Single-Sided DIMM
  - 4 banks per device => DIMM has 4 banks
- 512MB DIMM = 8x64Mx8, 4 Banks

RAMBUS (RDRAM)

- Protocol based RAM w/ narrow (16-bit) bus
  - High clock rate (400 Mhz), but long latency
  - Pipelined operation
- Multiple arrays w/ data transferred on both edges

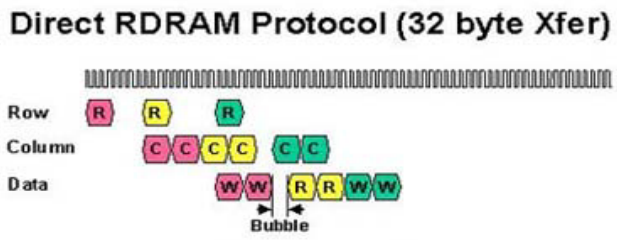


RAMBUS Bank



RDRAM Memory System

RDRAM Timing



- Separate Row and Column control
  - Enables pipelining, enhances performance
- Smaller Write-Read bubble
  - Increases bandwidth
- Bank conflicts still possible
  - High bank counts reduce probability of conflicts

Independent Memory Banks

- Standard PC Upgrade Path
  - Traditional DIMMS => 8 devices at a time with 8-bit chips
  - Rambus RIMMs => One at a time
- Successful Markets: PlayStation 2 (High Bandwidth, Small Memory)
- Rambus: 400MHz, 16-bits per channel, 2-bits per clock
  - 1.6GB/sec per channel (only 1 chip needed)
  - 2 Rambus Channels in Parallel, 3.2GB/sec memory bandwidth
- Traditional:PC100 SDRAM: 100MHz, 1-bit per clock
  - Would need 32 chips to achieve 3.2GB/sec bandwidth

Interleaving Summary

- Banks
  - Method to get high bandwidth with cheap (narrow) bus
- Bandwidth determines memory capacity
  - Hard to make many banks from narrow DIMMs
  - 32, 64-bit banks from 1x64MB DRAMS => 2048 DIMMS => 4GB
  - Can't force customers to buy so much memory to get good bandwidth
  - Must use wider DRAMs
  - RAMBUS does better with small memory systems (PS2)
  - Big servers have lots of memory so traditional banking works

Next Time

- Multiprocessors