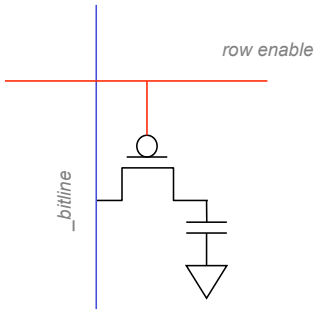
Lecture 9
DRAM Memory



Fall 2016

Pejman Lotfi-Kamran

Slides by Michael Ferdman @ Stony Brook University

Where Are We?

Fr	Sa	Su	Mo	Tu
	27-Shahrivar		29-Shahrivar	
	3-Mehr		5-Mehr	
	10-Mehr		12-Mehr	
	17-Mehr		19-Mehr	
	24-Mehr		26-Mehr	
	1-Aban		3-Aban	
	8-Aban		10-Aban	
	15-Aban		17-Aban	
	22-Aban		24-Aban	
	29-Aban		1-Azar	
	6-Azar		8-Azar	
	13-Azar		15-Azar	
	20-Azar		22-Azar	
	27-Azar		29-Azar	
	4-Dey		6-Dey	

This Lecture
- DRAM

Next Lecture:
- Piplining

SRAM vs. DRAM

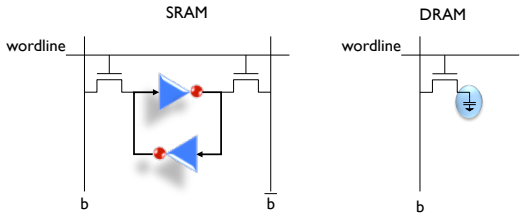
SRAM = Static RAM
- As long as power is present, data is retained

DRAM = Dynamic RAM
- If you don't do anything, you lose the data

SRAM: 6T per bit
- built with normal high-speed CMOS technology

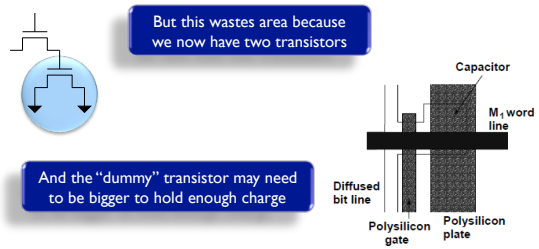
DRAM: 1T per bit (+1 capacitor)
- built with special DRAM process optimized for density

Hardware Structures



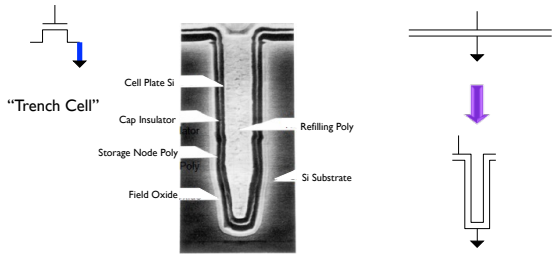
Implementing the Capacitor (1/2)

You can use a "dead" transistor gate:



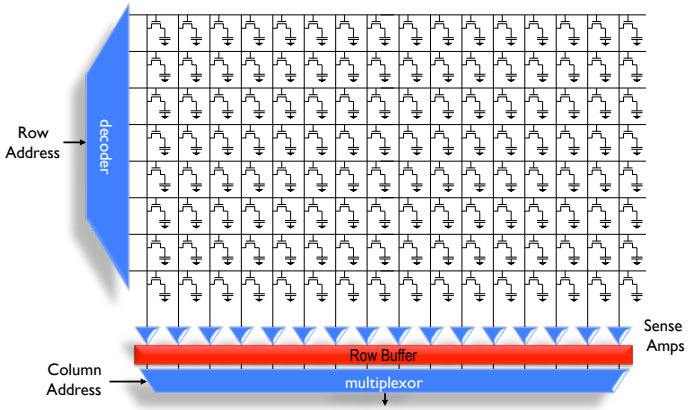
Implementing the Capacitor (2/2)

There are other advanced structures



DRAM figures from this slide and the previous one were taken from Prof. Nikolic's EECS141/2003 Lecture notes from UC-Berkeley

DRAM Chip Organization (1/2)



DRAM Chip Organization (2/2)

Low-Level organization is very similar to SRAM

Cells are only single-ended

- Reads *destructive*: contents are erased by reading

Row buffer holds read data

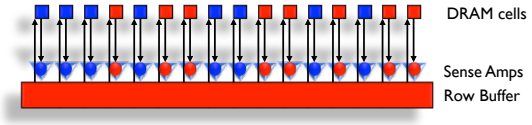
- Data in row buffer is called a *DRAM row*
  - Often called "page" - not necessarily same as OS page
- Read gets entire row into the buffer
- Block reads always performed out of the row buffer
  - Reading a whole row, but accessing one block
  - Similar to reading a cache line, but accessing one word

DRAM Read

After a read, the contents of the DRAM cell are gone
- But still "safe" in the row buffer

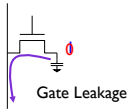
Write bits back before doing another read

Reading into buffer is slow, but reading buffer is fast
- Try reading multiple lines from buffer *(row-buffer hit)*

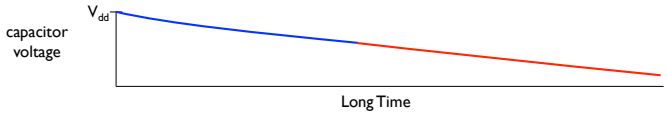


DRAM Refresh (1/2)

Gradually, DRAM cell loses contents
- Even if it's not accessed
- This is why it's called "dynamic"



DRAM must be regularly read and re-written
- What to do if no read/write to row for long time?



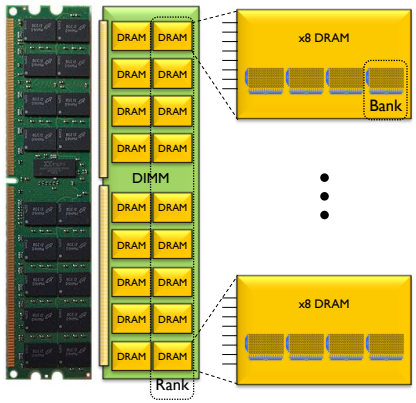
DRAM Refresh (2/2)

Burst Refresh
- Stop the world, refresh all memory

Distributed refresh
- Space out refresh one row at a time
- Avoids blocking memory for a long time

Self-refresh (low-power mode)
- Tell DRAM to refresh itself
- Turn off memory controller
- Takes some time to exit self-refresh

DRAM Organization



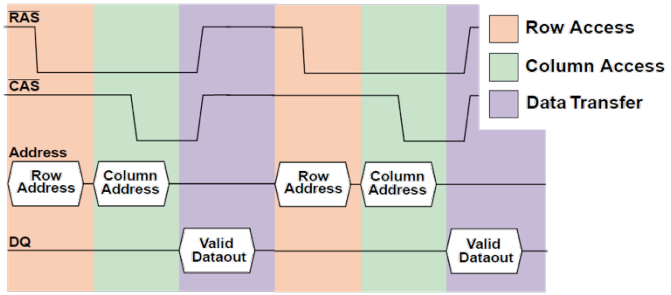
All banks within the rank share all address and control pins

All banks are independent, but can only talk to one bank at a time

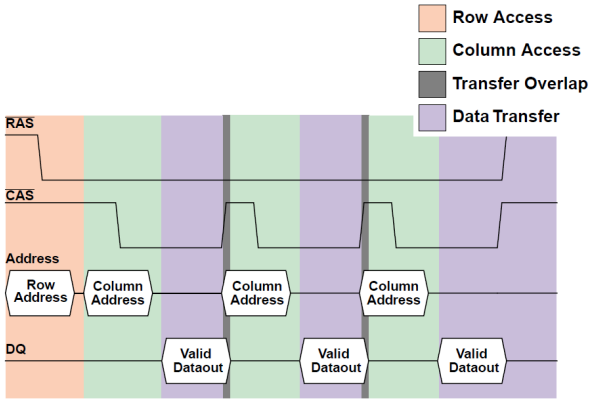
x8 means each DRAM outputs 8 bits, need 8 chips for DDRx (64-bit)

Why 9 chips per rank? 64 bits data, 8 bits ECC

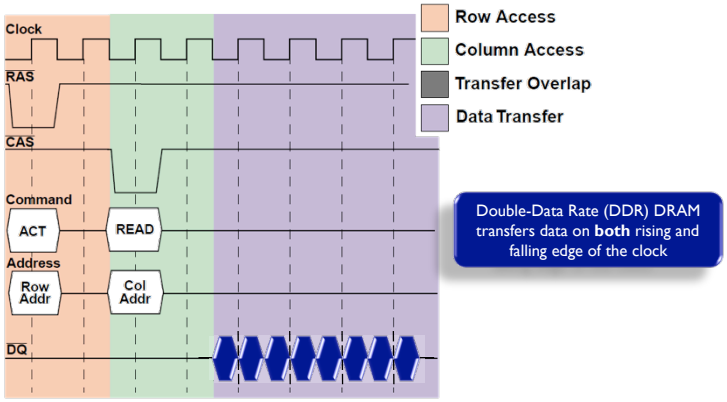
DRAM Read Timing



DRAM Read Timing with Fast-Page Mode

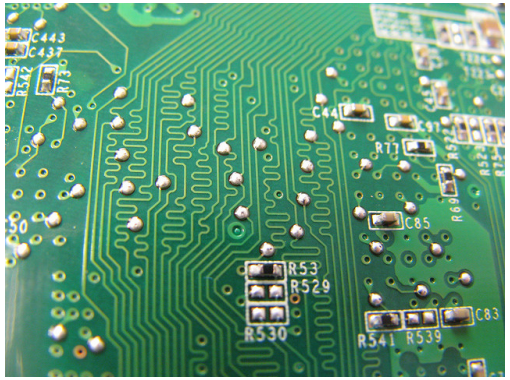


SDRAM Read Timing



Double-Data Rate (DDR) DRAM transfers data on **both** rising and falling edge of the clock

DRAM Signal Timing



CPU-to-Memory Interconnect (1/3)

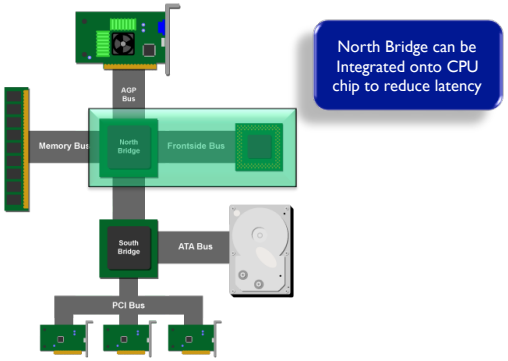
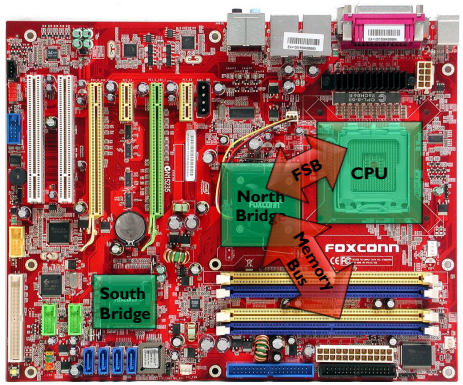
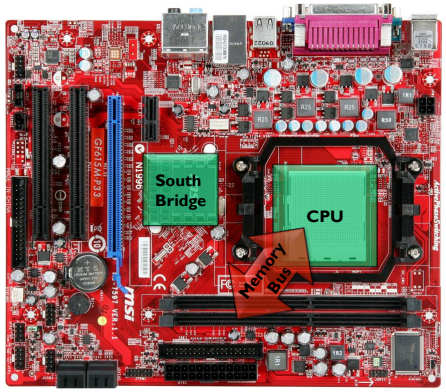


Figure from ArsTechnica

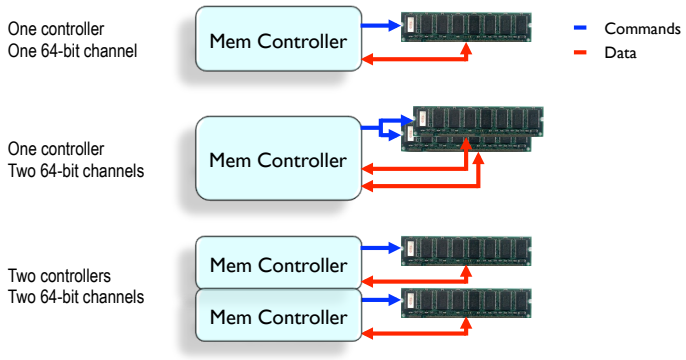
CPU-to-Memory Interconnect (2/3)



CPU-to-Memory Interconnect (3/3)



Memory Channels



Memory-Level Parallelism (MLP)

What if memory latency is 10000 cycles?

- Runtime dominated by waiting for memory
- What matters is ***overlapping memory accesses***

Memory-Level Parallelism (MLP):

- "Average number of outstanding memory accesses when at least one memory access is outstanding."

MLP is a metric

- ***Not*** a fundamental property of workload
- Dependent on the microarchitecture

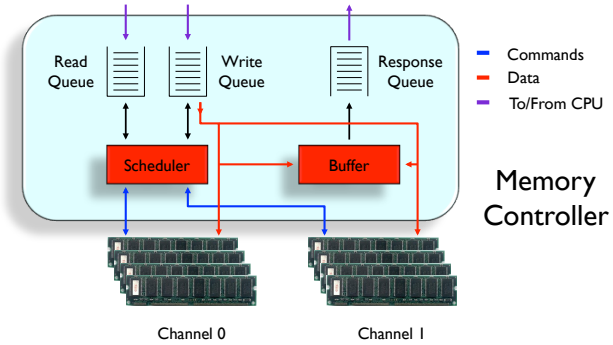
AMAT with MLP

If …
cache hit is 10 cycles (core to L1 and back)
memory access is 100 cycles (core to mem and back)

Then …
at 50% miss ratio, avg. access: 0.5×10+0.5×100 = 55

Unless MLP is >1.0, then…
at 50% mr,1.5 MLP,avg. access:(0.5×10+0.5×100)/1.5 = 37
at 50% mr,4.0 MLP,avg. access:(0.5×10+0.5×100)/4.0 = 14

Memory Controller (1/2)



Memory Controller (2/2)

Memory controller connects CPU and DRAM

Receives requests after cache misses in LLC

- Possibly originating from multiple cores

Complicated piece of hardware, handles:

- DRAM Refresh
- Row-Buffer Management Policies
- Address Mapping Schemes
- Request Scheduling

Row-Buffer Management Policies

Open-page

- After access, keep page in DRAM row buffer
- Next access to same page → lower latency
- If access to different page, must close old one first
  - Good if lots of locality

Close-page

- After access, immediately close page in DRAM row buffer
- Next access to different page → lower latency
- If access to different page, old one already closed
  - Good is no locality (random access)

Request Scheduling (1/3)

Write buffering

- Writes can wait until reads are done

Queue DRAM commands

- Usually into per-bank queues
- Allows easily reordering ops. meant for same bank

Common policies:

- *First-Come-First-Served (FCFS)*
- *First-Ready—First-Come-First-Served (FR-FCFS)*

Request Scheduling (2/3)

First-Come-First-Served

- Oldest request first

First-Ready—First-Come-First-Served

- *Prioritize column changes over row changes*
- *Skip over older conflicting requests*
- Find row hits (on queued reqs., even if close-page policy)
- Find oldest
  - If no conflicts with in-progress request → good
  - Otherwise (if conflicts), try next oldest

Overcoming Memory Latency

Caching

- Reduce average latency by avoiding DRAM altogether
- Limitations
  - Capacity (programs keep increasing in size)
  - Compulsory misses

Prefetching

- Guess what will be accessed next
  - Put in into the cache