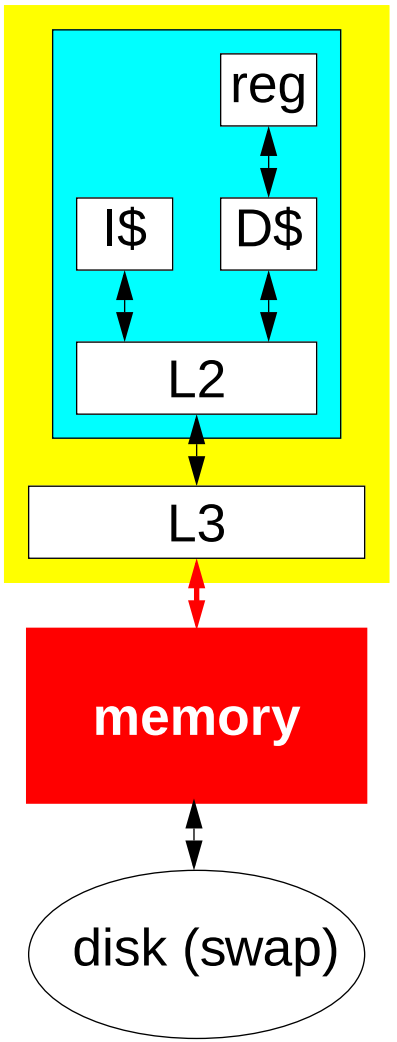
Storage Hierarchy II: Main Memory

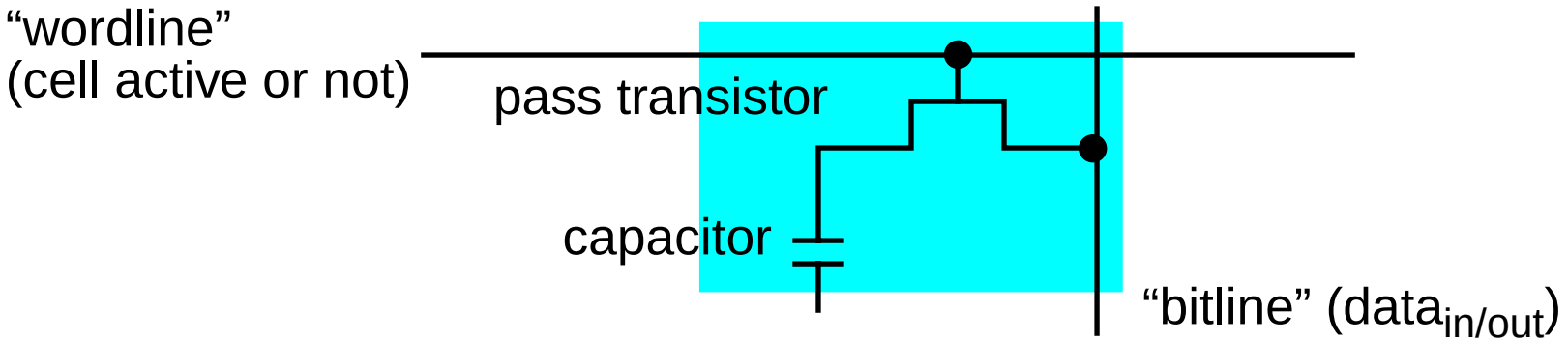


main memory

- memory technology (DRAM)
- interleaving
- special DRAMs
- processor/memory integration

virtual memory and address translation

DRAM (Dynamic Random Access Memory)



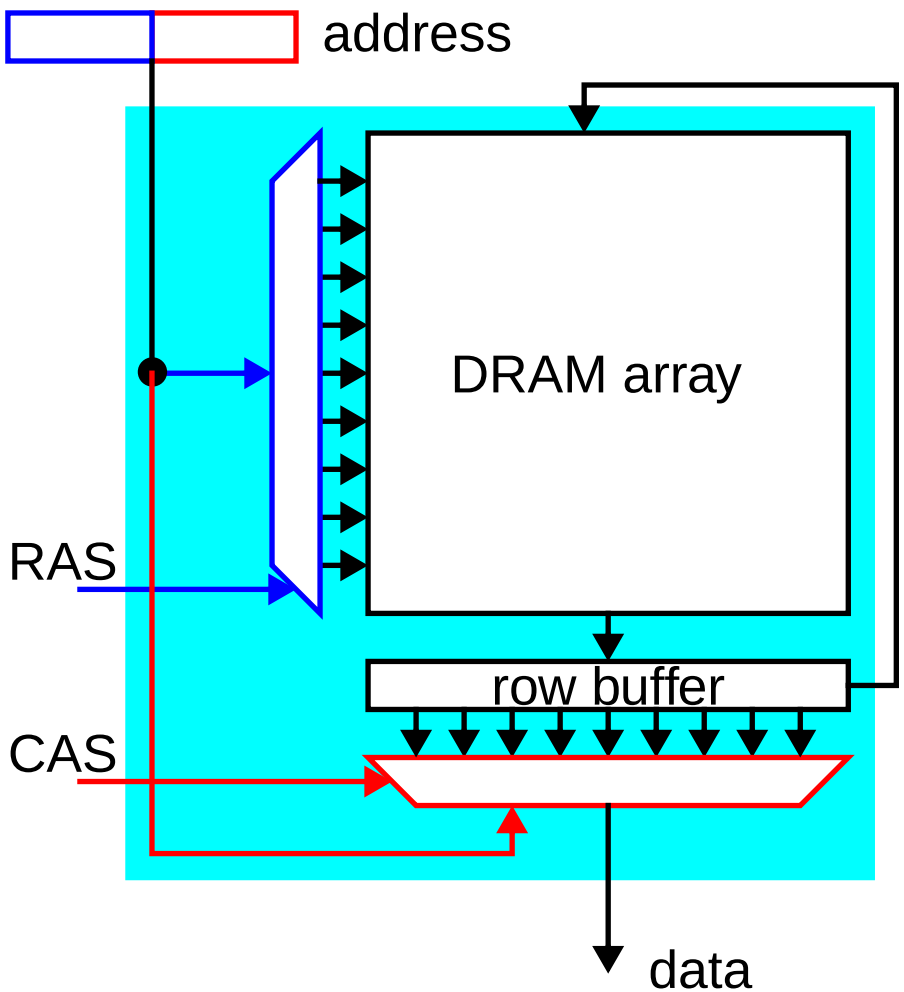
- bit stored as charge in capacitor
  - optimized for density (1 transistor, 6 for SRAM)

– capacitor discharges on a read (destructive read)
  - read is automatically followed by a write (to restore bit)

– charge leaks away over time
  - refresh by reading/writing every bit once every 2ms (row at a time)

- access time (time to read)
- cycle time (time between reads) > access time

DRAM Organization



- square row/column matrix
- multiplexed address lines
- internal row buffer
- operation
  - put row address on lines
  - set row address strobe (RAS)
  - read row into row buffer
  - put column address on lines
  - set column address strobe (CAS)
  - read column bits out of row buffer
  - write row buffer contents to row
- usually narrow interface (data)

Comparison with SRAM

SRAM

- optimized for speed, then density
  - + 1/4–1/8 access time of DRAM
  - – 1/4 density of DRAM
- bits stored as flip-flops (4-6 transistors per bit)
- static: bit not erased on a read
  - + no need to refresh
  - – greater power dissipated than DRAM
  - + access time = cycle time
- non-multiplexed address/data lines

DRAM Specs

Year	#bits	Access Time	Cycle Time
1980	64Kb	150ns	300ns
1990	1Mb	80ns	160ns
1993	4Mb	60ns	120ns
2000	64Mb	50ns	100ns

- density: +60% annual
  - Moore’s law: doubles every 18 months
- speed: %7 annual
  - much flatter improvement

Simple Main Memory

- 32-bit DRAM (1 word of data at a time)
  - pretty wide for an actual DRAM
- access time: 2 cycles (A)
- transfer time: 1 cycle (T)
  - time on the bus
- cycle time: 4 cycles (B = cycle time - access time)
- what is the miss penalty for 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

can we speed this up?

- lower latency?
  - no
    - A,B & T are fixed
    - “9 women...”
- 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	*

new parameter

- 64-bit DRAMs

4-word access = 7 cycles

4-word cycle = 8 cycles

– 64-bit bus
  - wide buses (especially off-chip) are hard
  - electrical problems

– larger expansion size

Bandwidth: Simple Interleaving/Banking

use multiple DRAMs, exploit their aggregate bandwidth

- each DRAM called a bank
  - not true: sometimes collection of DRAMs together 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

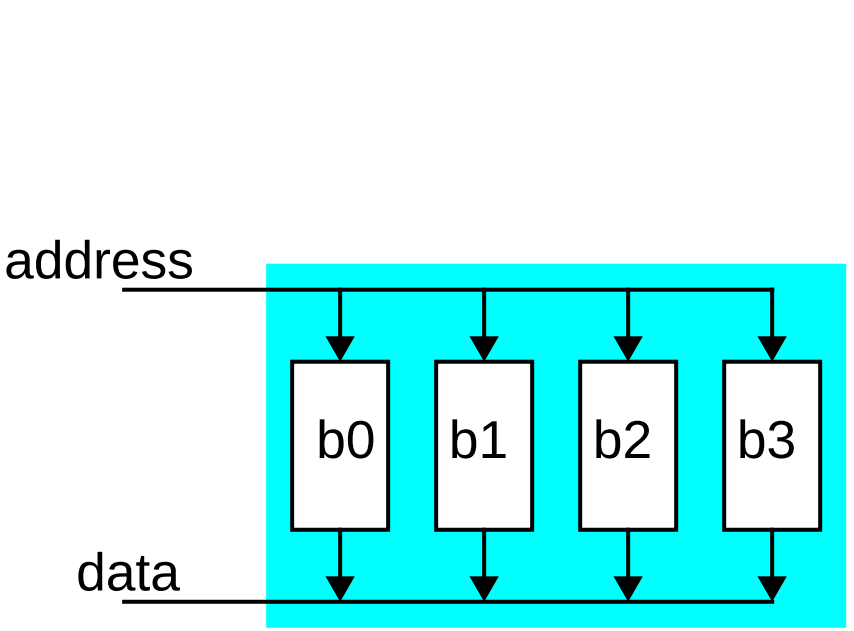
+ can start a new access in cycle 5
+ overlap access with transfer
+ and still use a 32-bit bus!

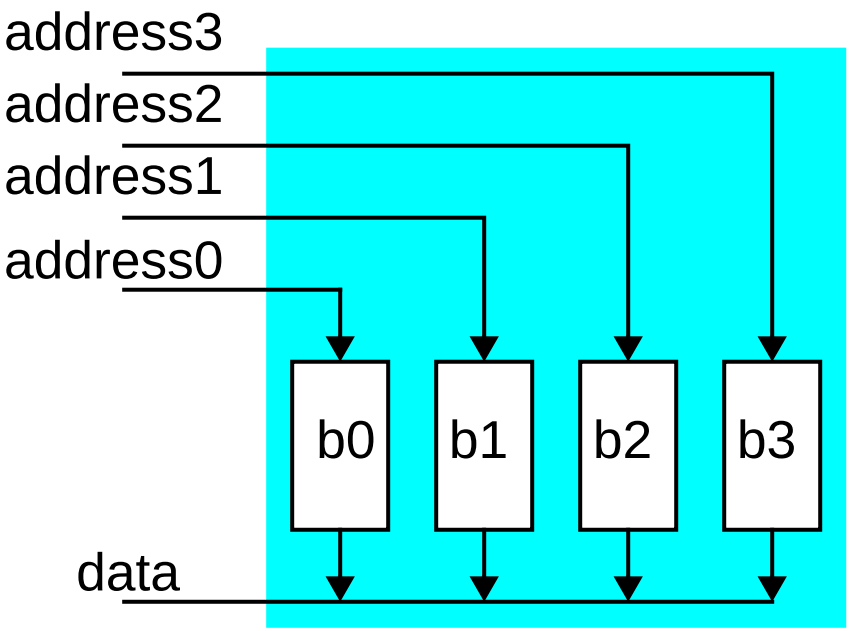
Bandwidth: Complex Interleaving

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
- so why use complex interleaving?

DRAM Optimizations

normal operation: read row into buffer, read column from buffer

observation: why not do multiple accesses from row buffer?

- nibble mode: additional bits per access (narrow DRAMs)
- page mode: change column address
- static column mode (SCRAM): don’t toggle CAS
- cached DRAMs: multiple row buffers

orthogonally: synchronous DRAMs (SDRAM)

- clock replaces RAS/CAS

+ faster

RAMBUS

a completely new memory interface [Horowitz]

- high level behaviors (like a memory controller)
- synchronous, no CAS/RAS
- internal caching (4–16 row buffers)
- split transaction (address queuing)
- 8-bit data
  - narrow (fix w/ multiple RAMBUS channels)
- variable length sequential transfers

+ 2ns/byte transfer time
  - 5GB/s: we initially said we couldn’t get this much b/w

– expensive

Processor/Memory Integration

the next logical step: processor and memory on same chip

- move on-chip: FP, L2 caches, graphics. why not memory?
- problem: processor/memory technologies incompatible
  - different number/kinds of metal layers
  - DRAM: capacitance is a good thing, logic: capacitance a bad thing

what needs to be done?

- use some DRAM area for simple processor (10% enough)
- eliminate external memory bus, milk performance from that
- integrate interconnect interfaces (processor/memory unit)
- re-examine tradeoffs: technology, cost, performance