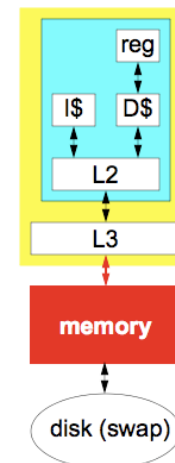


Lecture 27

Memory Technology + Storage + I/O

Storage Hierarchy II: Main Memory



main memory

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

virtual memory and address translation

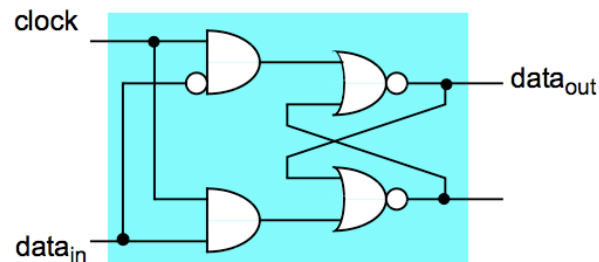
© 2004 by Lebeck, Sorin, Roth,
Hill, Wood, Sohi, Smith,
Vijaykumar, Lipasti

COMPSCI 220 / ECE 252 Lecture Notes
Storage Hierarchy II: Main Memory

1

University of Notre Dame

SRAM (Static Random Access Memory)



- "logic" (CPU process, registers are SRAM)
- store bits in flip-flops (cross-coupled NORs)
- not very dense (six transistors per bit)
- + fast
- + doesn't need to be "refreshed" (data stays as long as power is on)

© 2004 by Lebeck, Sorin, Roth,
Hill, Wood, Sohi, Smith,
Vijaykumar, Lipasti

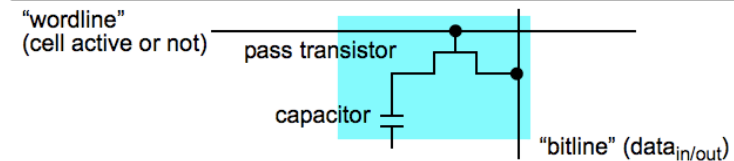
COMPSCI 220 / ECE 252 Lecture Notes
Storage Hierarchy I: Caches

12

University of Notre Dame

University of Notre Dame

DRAM (Dynamic Random Access Memory)



- bit stored as charge in **capacitor**
 - optimized for density (1 transistor for DRAM vs. 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 (not static)
 - 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

© 2004 by Lebeck, Sorin, Roth,
Hill, Wood, Sohi, Smith,
Vijaykumar, Lipasti

COMPSCI 220 / ECE 252 Lecture Notes
Storage Hierarchy II: Main Memory

4

University of Notre Dame

DRAM Chip Specs

Year	#bits	Access Time	Cycle Time
1980	64Kb	150ns	300ns
1990	1Mb	80ns	160ns
1993	4Mb	60ns	120ns
2000	64Mb	50ns	100ns
2004	1Gb	45ns	75ns

- density: +60% annual
 - Moore's law: density doubles every 18 months
- speed: %7 annual

Example: Simple Main Memory

- 32-bit wide 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)
 - B includes time to refresh after a read
- what is the miss penalty for a 4-word block?

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 ← Think about in context of leakage!
 - + access time = cycle time

Simple Main Memory

cycle	addr	mem
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
- higher bandwidth?

Bandwidth: Wider DRAMs

cycle	addr	mem
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

– 64-bit DRAM is probably too wide

Simple Interleaving

cycle	addr	bank0	bank1	bank2	bank3
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	B
6					T

4-word access = 6 cycles

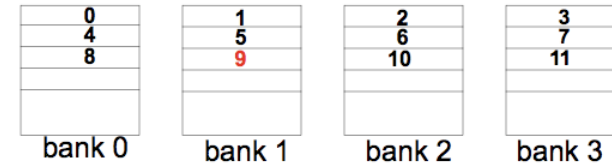
4-word cycle = 4 cycles

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

Bandwidth: 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
- **simple interleaving**: banks share address lines
- word A in bank $(A \% M)$ at $(A \text{ div } M)$
 - e.g., M=4, A=9: bank 1, location 2



Processor/Memory Integration

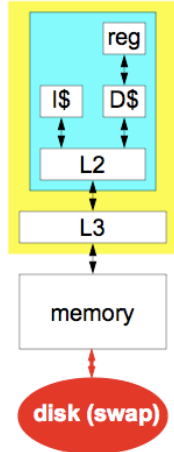
the next logical step: processor and memory on same chip

- 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
- research projects: PIM, IRAM

Storage Hierarchy III: I/O System



- often boring, but still quite important
 - ostensibly about general I/O, mainly about disks
- performance: latency & throughput
- disks
 - parameters
 - extensions
- buses

I/O Device Characteristics

- type
 - input: read only
 - output: write only
 - storage: both
- partner
 - human
 - machine
- data rate
 - peak transfer rate

device	type	partner	data rate KB/s
mouse	I	human	0.01
CRT	O	human	60,000
modem	I/O	machine	2-8
LAN	I/O	machine	500-6000
tape	storage	machine	2000
disk	storage	machine	2000-10,000

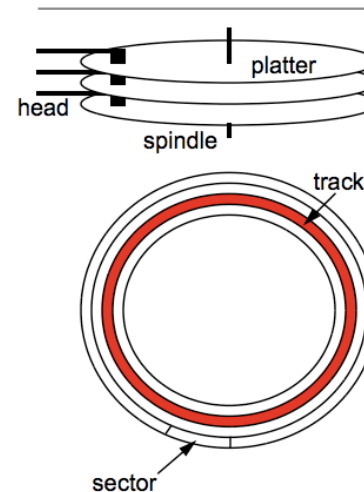
Of interest to this discussion

Both input & output

I/O (Disk) Performance

- who cares? you do
 - remember Amdahl's Law
 - want fast disk access (fast swap, fast file reads)
- I/O performance metrics
 - bandwidth of requests: *I/Os per second (IOPS)* → • Requests processed
 - raw data bandwidth: bytes per second
 - latency: response time
- is I/O (disk) latency important? why not just context-switch?

Disk Parameters



- 1–20 *platters* (data on both sides)
 - magnetic iron-oxide coating
 - 1 read/write head per side
- 500–2500 *tracks* per platter
 - sometimes fewer on inside tracks
- 32–128 *sectors* per track
 - usually fixed number of bytes/sector
 - data + ECC (parity) + gap
- 4–24GB total
- 3000–10000 RPM

Disk Performance Example

- parameters
 - 3600 RPM \Rightarrow 60 RPS (may help to think in units of tracks/sec)
 - avg seek time: 9ms
 - 100 sectors per track, 512 bytes per sector
 - controller + queuing delays: 1ms
- Q: average time to read 1 sector (512 bytes)?
 - $\text{rate}_{\text{transfer}} = 100 \text{ sectors/track} * 512 \text{ B/sector} * 60 \text{ RPS} = 2.4 \text{ MB/s}$
 - $t_{\text{transfer}} = 512 \text{ B} / 2.4 \text{ MB/s} = 0.2\text{ms}$
 - $t_{\text{rotation}} = .5 / 60 \text{ RPS} = 8.3\text{ms}$
 - $t_{\text{disk}} = 9\text{ms (seek)} + 8.3\text{ms (rotation)} + 0.2\text{ms (xfer)} + 1\text{ms} = 18.5\text{ms}$
 - t_{transfer} is only a small component! counter-intuitive?
 - end of story? no! t_{queuing} not fixed (gets longer with more requests)

Disk Alternatives

- solid state disk (SSD)
 - DRAM + battery backup with standard disk interface
 - + fast: no seek time, no rotation time, fast transfer rate
 - expensive
- FLASH memory
 - + fast: no seek time, no rotation time, fast transfer rate
 - + non-volatile
 - slow
 - "wears" out over time
- optical disks (CDs, DVDs)
 - cheap if write-once, expensive if write-multiple
 - slow

Actually, reads are
proportional to normal
DRAM, but writes take
longer

Disk Usage Models

- data mining + supercomputing
 - large files, sequential reads
 - raw data transfer rate ($\text{rate}_{\text{transfer}}$) is most important
 - transaction processing
 - large files, but random access, many small requests
 - IOPS is most important
 - time sharing filesystems
 - small files, sequential accesses, potential for file caching
 - IOPS is most important
- What metrics are important for what applications?
- must design disk (I/O) system based on target workload
- use disk benchmarks (they exist)

Extensions to Conventional Disks

- increasing density: more sensitive heads, finer control
 - increases cost
- fixed head: head per track
 - + seek time eliminated
 - low track density
- parallel transfer: simultaneous read from multiple platters
 - difficulty in looking onto different tracks on multiple surfaces
 - lower cost alternatives possible (disk arrays)

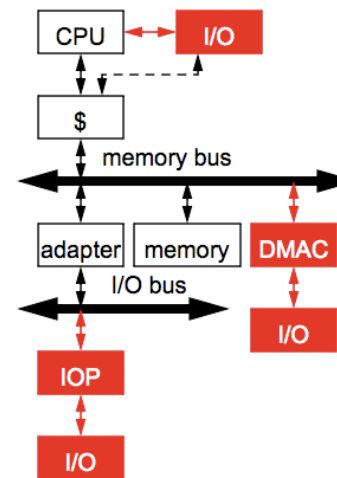
More Extensions to Conventional Disks

- **disk caches:** disk-controller RAM buffers data
 - + fast writes: RAM acts as a write buffer
 - + better utilization of host-to-device path
 - high miss rate increases request latency
- **disk scheduling:** schedule requests to reduce latency
 - e.g., schedule request with shortest seek time
 - e.g., “elevator” algorithm for seeks (head sweeps back and forth)
 - works best for unlikely cases (long queues)

Bus Issues (Memory & I/O Buses)

- **clocking:** is bus clocked?
 - synchronous: clocked, short bus \Rightarrow fast
 - asynchronous: no clock, use “handshaking” instead \Rightarrow slow
- **switching:** when is control of bus acquired and released?
 - atomic: bus held until request complete \Rightarrow slow
 - split-transaction (pipelined): bus free btwn request & reply \Rightarrow fast
- **arbitration:** how do we decide who gets the bus next?
 - overlap arbitration for next master with current transfer
 - daisy chain: closer devices have priority \Rightarrow slow
 - distributed: wired-OR, low-priority back-off \Rightarrow medium
- some other issues
 - split data/address lines, width, burst transfer

I/O System Architecture



- buses
 - memory bus
 - I/O bus
- I/O processing
 - program controlled
 - DMA
 - I/O processors (IOPs)

I/O and Memory Buses

		bits	MHz	peak MB/s	special features
memory buses	Summit	128	60	960	
	Challenge	256	48	1200	
	XDBus	144	66	1056	
I/O buses	ISA	16	8	16	original PC bus
	IDE	16	8	16	tape, CD-ROM
	PCI	32(64)	33(66)	133(266)	“plug+play”
	SCSI/2	8/16	5/10	10/20	high-level interface
	PCMCIA	8/16	8	16	modem, “hot-swap”
	USB	serial	isoch.	1.5	power line, packetized
	FireWire	serial	isoch.	100	fast USB

- memory buses: speed (usually custom design)
- I/O buses: compatibility (usually industry standard) + cost