CS152: Computer Architecture and Engineering
Locality and Memory Technology

October 29, 1997

Dave Patterson (http.cs.berkeley.edu/~patterson)

lecture slides: http://www-inst.eecs.berkeley.edu/~cs152/

DAP Fa97, © U.CB

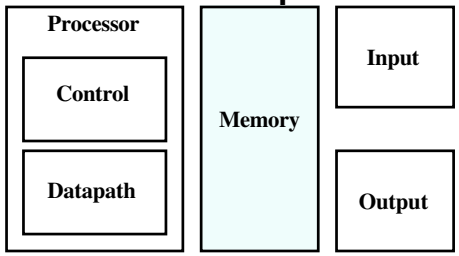
Recap

- MIPS I instruction set architecture made pipeline visible (delayed branch, delayed load)
- More performance from deeper pipelines, parallelism
- Increasing length of pipe increases impact of hazards; pipelining helps instruction bandwidth, not latency
- SW Pipelining
  - Symbolic Loop Unrolling to get most from pipeline with little code expansion, little overhead
- Dynamic Branch Prediction + early branch address for speculative execution
- Superscalar and VLIW
  - CPI < 1
  - Dynamic issue vs. Static issue
  - More instructions issue at same time, larger the penalty of hazards
  - Intel EPIC in IA-64 a hybrid: compact LIW + data hazard check

DAP Fa97, © U.CB

The Big Picture: Where are We Now?

- The Five Classic Components of a Computer



- Today's Topics:
  - Recap last lecture
  - Locality and Memory Hierarchy
  - Administrivia
  - SRAM Memory Technology
  - DRAM Memory Technology
  - Memory Organization

DAP Fa97, © U.CB

Technology Trends (from 1st lecture)

	Capacity	Speed (latency)
Logic:	2x in 3 years	2x in 3 years
DRAM:	4x in 3 years	2x in 10 years
Disk:	4x in 3 years	2x in 10 years

DRAM

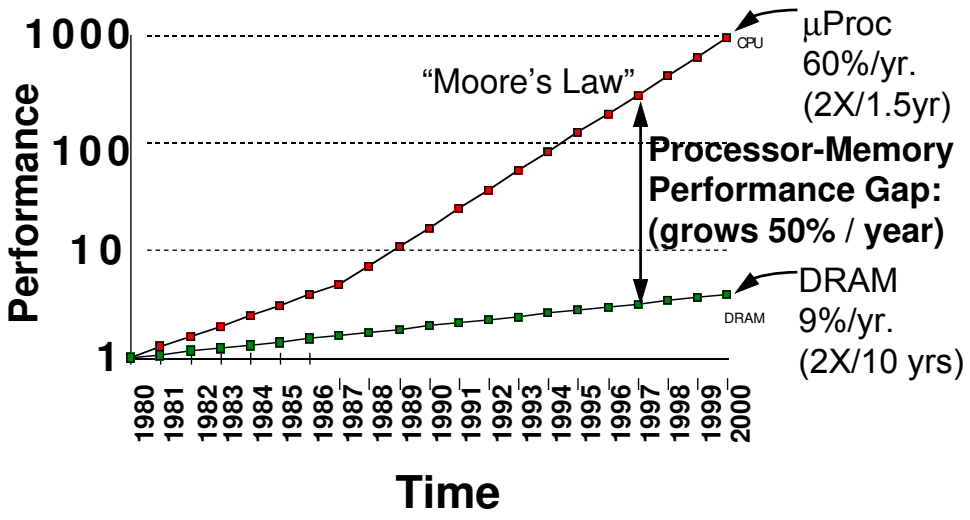
Year	Size	Cycle Time
1980	64 Kb	250 ns
1983	256 Kb	220 ns
1986	1 Mb	190 ns
1989	4 Mb	165 ns
1992	16 Mb	145 ns
1995	64 Mb	120 ns

1000:1! 2:1!

DAP Fa97, © U.CB

Who Cares About the Memory Hierarchy?

Processor-DRAM Memory Gap (latency)



DAP Fa97, © U.CB

Today's Situation: Microprocessor

- Rely on caches to bridge gap
- Microprocessor-DRAM performance gap
  - time of a full cache miss in instructions executed

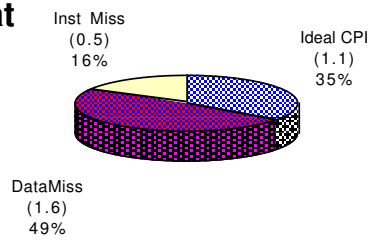
1st Alpha (7000):	340 ns/5.0 ns = 68 clks x 2 or	136 instructions
2nd Alpha (8400):	266 ns/3.3 ns = 80 clks x 4 or	320 instructions
3rd Alpha (t.b.d.):	180 ns/1.7 ns =108 clks x 6 or	648 instructions

  - 1/2X latency x 3X clock rate x 3X Instr/clock ⇒ ≈5X

DAP Fa97, © U.CB

Impact on Performance

- Suppose a processor executes at
  - Clock Rate = 200 MHz (5 ns per cycle)
  - CPI = 1.1
  - 50% arith/logic, 30% ld/st, 20% control



- Suppose that 10% of memory operations get 50 cycle miss penalty
- CPI = ideal CPI + average stalls per instruction
  = 1.1(cyc) +(0.30 (datamops/ins)
  x 0.10 (miss/datamop) x 50 (cycle/miss))
  = 1.1 cycle + 1.5 cycle
  = 2. 6
- 58 % of the time the processor is stalled waiting for memory!
- a 1% instruction miss rate would add an additional 0.5 cycles to the CPI!

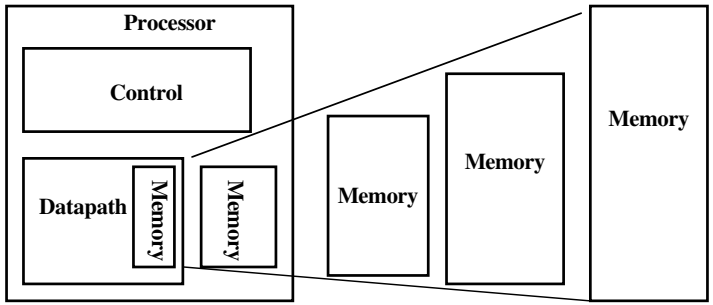
DAP Fa97, © U.CB

The Goal: illusion of large, fast, cheap memory

- Fact: Large memories are slow, fast memories are small
- How do we create a memory that is large, cheap and fast (most of the time)?
  - Hierarchy
  - Parallelism

DAP Fa97, © U.CB

An Expanded View of the Memory System



DAP Fa97, © U.CB

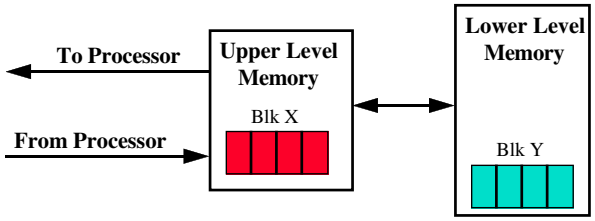
Why hierarchy works

- The Principle of Locality:
  - Program access a relatively small portion of the address space at any instant of time.

DAP Fa97, © U.CB

Memory Hierarchy: How Does it Work?

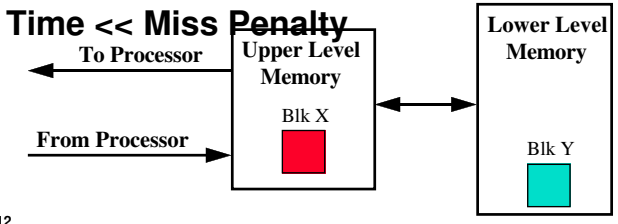
- Temporal Locality (Locality in Time):
  => Keep most recently accessed data items closer to the processor
- Spatial Locality (Locality in Space):
  => Move blocks consists of contiguous words to the upper levels



DAP Fa97, © U.CB

Memory Hierarchy: Terminology

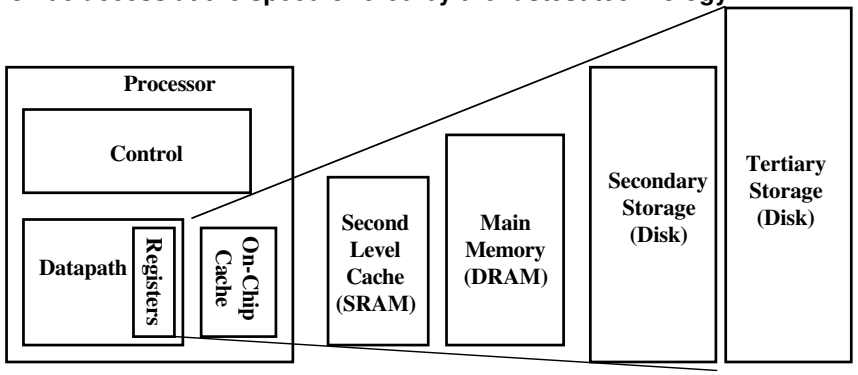
- Hit: data appears in some block in the upper level (example: Block X)
  - Hit Rate: the fraction of memory access found in the upper level
  - Hit Time: Time to access the upper level which consists of RAM access time + Time to determine hit/miss
- Miss: data needs to be retrieve from a block in the lower level (Block Y)
  - Miss Rate = 1 - (Hit Rate)
  - Miss Penalty: Time to replace a block in the upper level + Time to deliver the block the processor
- Hit Time << Miss Penalty



DAP Fa97, © U.CB

Memory Hierarchy of a Modern Computer System

- By taking advantage of the principle of locality:
  - Present the user with as much memory as is available in the cheapest technology.
  - Provide access at the speed offered by the fastest technology.



Speed (ns):	1s	10s	100s	10,000,000s (10s ms)	10,000,000,000s (10s sec)
Size (bytes):	100s	Ks	Ms	Gs	Ts

DAP Fa97, © U.CB

How is the hierarchy managed?

- Registers <-> Memory
  - by compiler (programmer?)
- cache <-> memory
  - by the hardware
- memory <-> disks
  - by the hardware and operating system (virtual memory)
  - by the programmer (files)

DAP Fa97, © U.CB

Memory Hierarchy Technology

- Random Access:
  - "Random" is good: access time is the same for all locations
  - DRAM: Dynamic Random Access Memory
    - High density, low power, cheap, slow
    - Dynamic: need to be "refreshed" regularly
  - SRAM: Static Random Access Memory
    - Low density, high power, expensive, fast
    - Static: content will last "forever"(until lose power)
- "Non-so-random" Access Technology:
  - Access time varies from location to location and from time to time
  - Examples: Disk, CDROM
- Sequential Access Technology: access time linear in location (e.g.,Tape)
- The next two lectures will concentrate on random access technology
  - The Main Memory: DRAMs + Caches: SRAMs

DAP Fa97, © U.CB

Main Memory Background

- Performance of Main Memory:
  - Latency: Cache Miss Penalty
    - *Access Time*: time between request and word arrives
    - *Cycle Time*: time between requests
  - Bandwidth: I/O & Large Block Miss Penalty (L2)
- Main Memory is *DRAM*: Dynamic Random Access Memory
  - Dynamic since needs to be refreshed periodically (8 ms)
  - Addresses divided into 2 halves (Memory as a 2D matrix):
    - *RAS* or *Row Access Strobe*
    - *CAS* or *Column Access Strobe*
- Cache uses *SRAM*: Static Random Access Memory
  - No refresh (6 transistors/bit vs. 1 transistor /bit)
  - Address not divided
- *Size*: DRAM/SRAM ≈ *4-8*, *Cost/Cycle time*: SRAM/DRAM ≈ *8-16*

DAP Fa97, © U.CB

Random Access Memory (RAM) Technology

- Why do computer designers need to know about RAM technology?
  - Processor performance is usually limited by memory bandwidth
  - As IC densities increase, lots of memory will fit on processor chip
    - Tailor on-chip memory to specific needs
      - Instruction cache
      - Data cache
      - Write buffer
- What makes RAM different from a bunch of flip-flops?
  - Density: RAM is much more denser

DAP Fa97, © U.CB

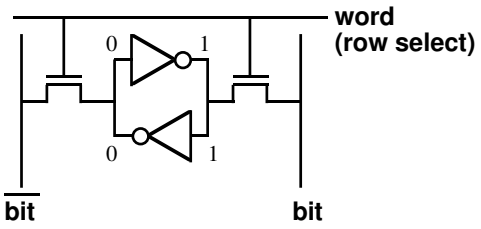
Administrative Issues

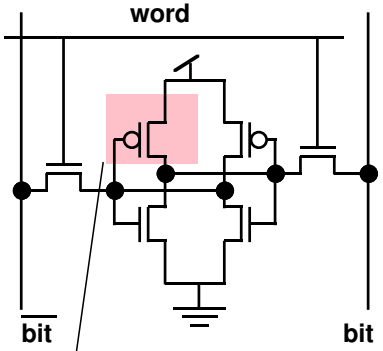
- Office Hours:
  - Gebis: Tuesday, 3:30-4:30
  - Kirby: ?
  - Kozyrakis: Monday 1pm-2pm, Th 11am-noon 415 Soda Hall
  - Patterson: Wednesday 12-1 and Wednesday 3:30-4:30 635 Soda Hall
- Reflector site for handouts and lecture notes (backup):
  - http://HTTP.CS.Berkeley.EDU/~patterson/152F97/index_handouts.html
  - http://HTTP.CS.Berkeley.EDU/~patterson/152F97/index_lectures.html
- Computers in the news
  - Intel buys DEC fab line for $700M + rights to DEC patents; + Intel pays some royalty per chip from 1997-2007
  - DEC has rights to continue fab Alpha in future on Intel owned line
  - Intel offers jobs to 2000 fab/process people; DEC keeps MPU designers
  - DEC will build servers based on IA-64 (+Alpha); customers choose
  - Intel gets rights to DEC UNIX

DAP Fa97, © U.CB

Static RAM Cell

6-Transistor SRAM Cell



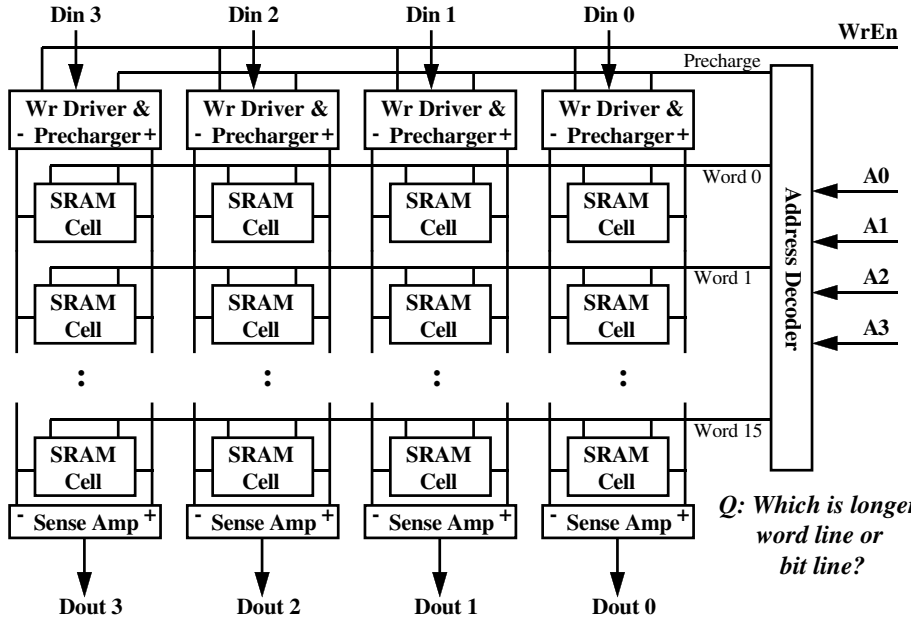


replaced with pullup to save area

- Write:
  1. Drive bit lines (bit=1, $\overline{bit}$=0)
  2.. Select row
- Read:
  1. Precharge bit and bit to Vdd
  2.. Select row
  3. Cell pulls one line low
  4. Sense amp on column detects difference between bit and $\overline{bit}$

DAP Fa97, © U.CB

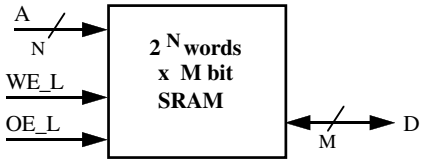
Typical SRAM Organization: 16-word x 4-bit



Q: Which is longer: word line or bit line?

DAP Fa97, © U.CB

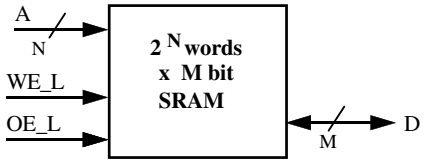
Logic Diagram of a Typical SRAM

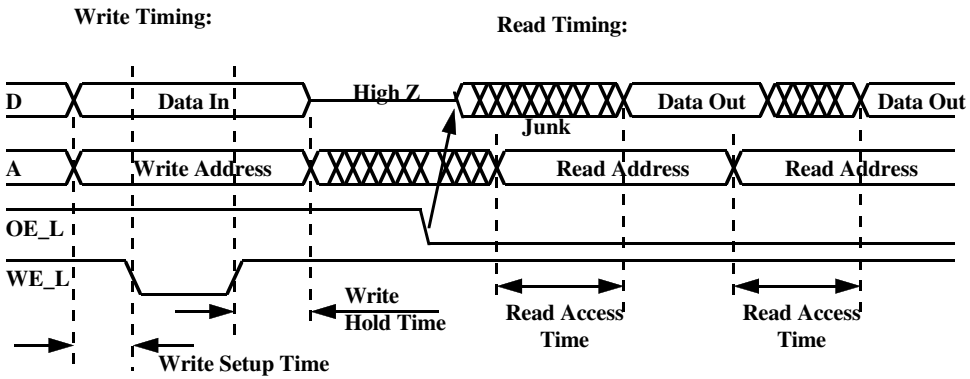


- Write Enable is usually active low (WE_L)
- Din and Dout are combined to save pins:
  - A new control signal, output enable (OE_L) is needed
  - WE_L is asserted (Low), OE_L is disasserted (High)
    - D serves as the data input pin
  - WE_L is disasserted (High), OE_L is asserted (Low)
    - D is the data output pin
  - Both WE_L and OE_L are asserted:
    - Result is unknown. Don't do that!!!
- Although could change VHDL to do what desire, must do the best with what you've got (vs. what you need)

DAP Fa97, © U.CB

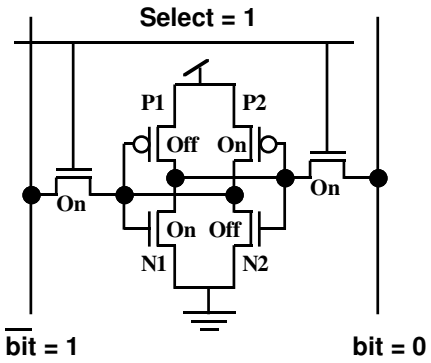
Typical SRAM Timing





DAP Fa97, © U.CB

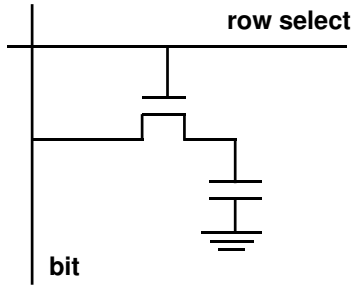
Problems with SRAM



- Six transistors use up a lot of area
- Consider a "Zero" is stored in the cell:
  - Transistor N1 will try to pull "bit" to 0
  - Transistor P2 will try to pull "bit bar" to 1
- But bit lines are precharged to high: Are P1 and P2 necessary?

DAP Fa97, © U.CB

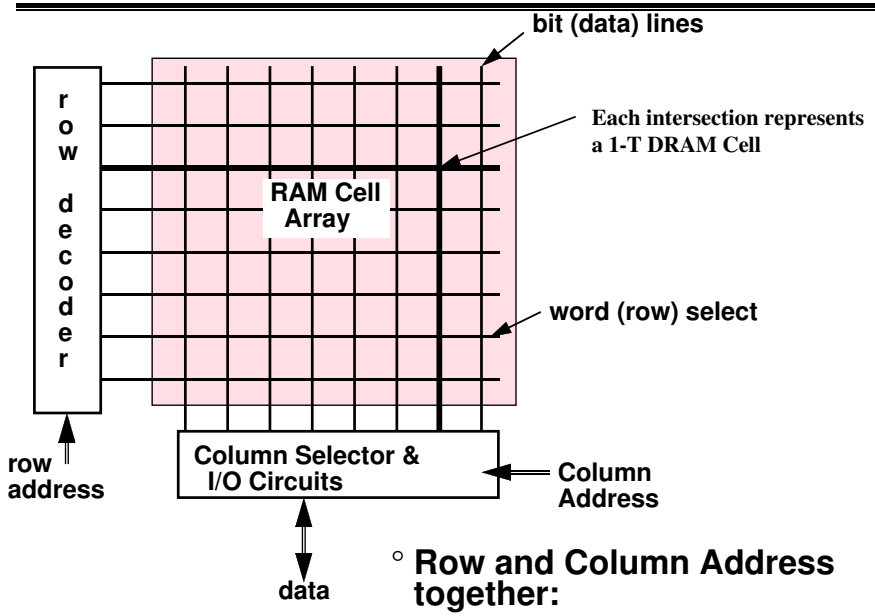
1-Transistor Memory Cell (DRAM)



- Write:
  - 1. Drive bit line
  - 2.. Select row
- Read:
  - 1. Precharge bit line to Vdd
  - 2.. Select row
  - 3. Cell and bit line share charges
    - Very small voltage changes on the bit line
  - 4. Sense (fancy sense amp)
    - Can detect changes of ~1 million electrons
  - 5. Write: restore the value
- Refresh
  - 1. Just do a dummy read to every cell.

DAP Fa97, © U.CB

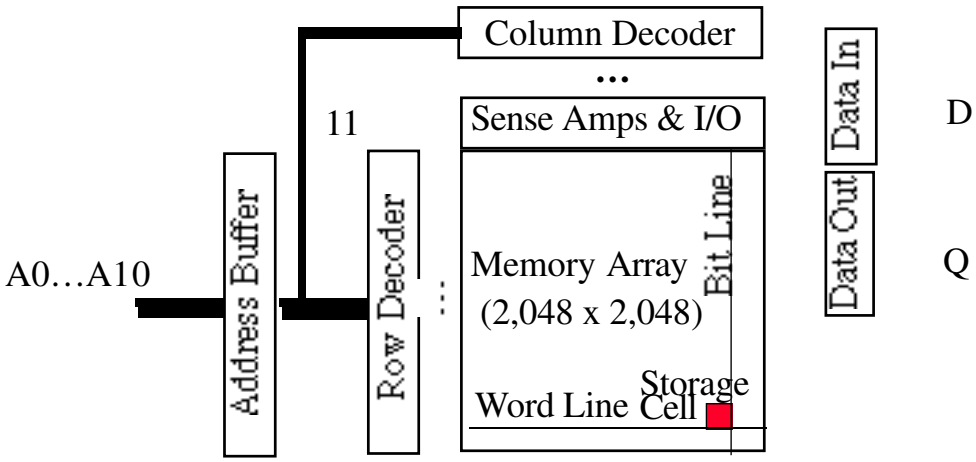
Classical DRAM Organization (square)



- Row and Column Address together:
  - Select 1 bit a time

DAP Fa97, © U.CB

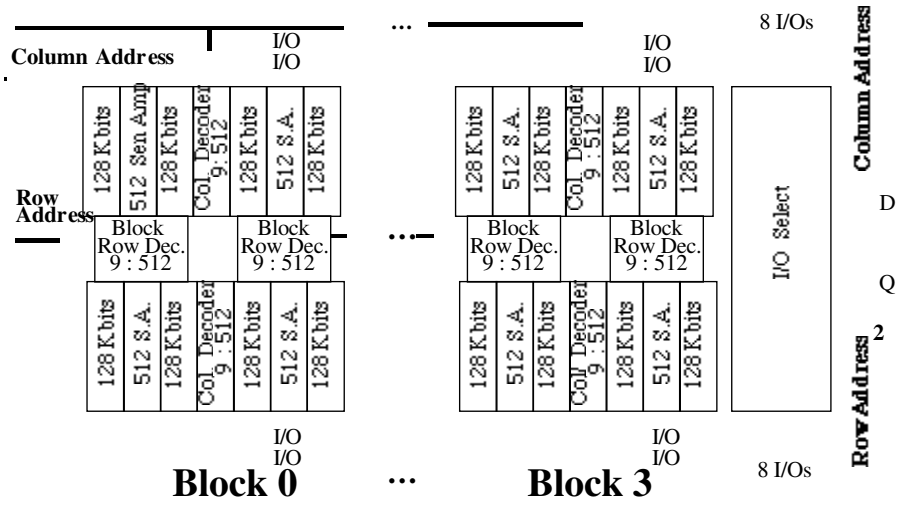
DRAM logical organization (4 Mbit)



- Square root of bits per RAS/CAS

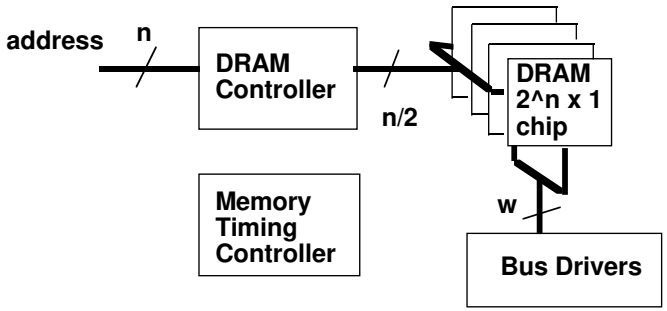
DAP Fa97, © U.CB

DRAM physical organization (4 Mbit)



DAP Fa97, © U.CB

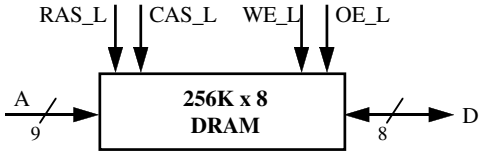
Memory Systems



Tc = Tcycle + Tcontroller + Tdriver

DAP Fa97, © U.CB

Logic Diagram of a Typical DRAM



- Control Signals (RAS_L, CAS_L, WE_L, OE_L) are all active low
- Din and Dout are combined (D):
  - WE_L is asserted (Low), OE_L is disasserted (High)
    - D serves as the data input pin
  - WE_L is disasserted (High), OE_L is asserted (Low)
    - D is the data output pin
- Row and column addresses share the same pins (A)
  - RAS_L goes low: Pins A are latched in as row address
  - CAS_L goes low: Pins A are latched in as column address
  - RAS/CAS edge-sensitive

DAP Fa97, © U.CB

Key DRAM Timing Parameters

- t_{RAC}: minimum time from RAS line falling to the valid data output.
  - Quoted as the speed of a DRAM
  - A fast 4Mb DRAM t_{RAC} = 60 ns
- t_{RC}: minimum time from the start of one row access to the start of the next.
  - t_{RC} = 110 ns for a 4Mbit DRAM with a t_{RAC} of 60 ns
- t_{CAC}: minimum time from CAS line falling to valid data output.
  - 15 ns for a 4Mbit DRAM with a t_{RAC} of 60 ns
- t_{PC}: minimum time from the start of one column access to the start of the next.
  - 35 ns for a 4Mbit DRAM with a t_{RAC} of 60 ns

DAP Fa97, © U.CB

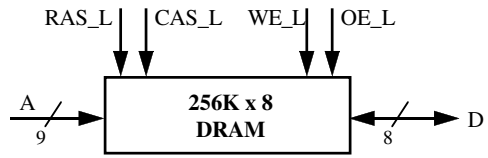
DRAM Performance

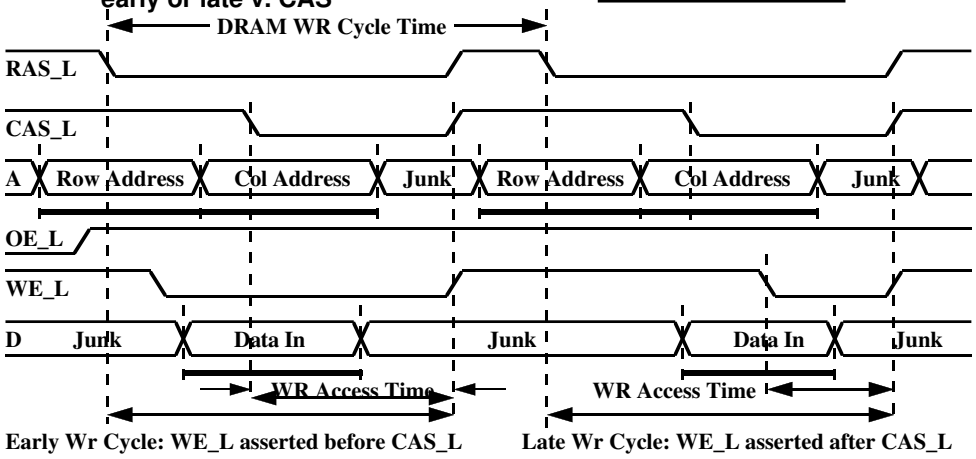
- A 60 ns (t_{RAC}) DRAM can
  - perform a row access only every 110 ns (t_{RC})
  - perform column access (t_{CAC}) in 15 ns, but time between column accesses is at least 35 ns (t_{PC}).
    - In practice, external address delays and turning around buses make it 40 to 50 ns
- These times do not include the time to drive the addresses off the microprocessor nor the memory controller overhead.
  - Drive parallel DRAMs, external memory controller, bus to turn around, SIMM module, pins…
  - 180 ns to 250 ns latency from processor to memory is good for a "60 ns" (t_{RAC}) DRAM

DAP Fa97, © U.CB

DRAM Write Timing

- Every DRAM access begins at:
  - The assertion of the RAS_L
  - 2 ways to write: early or late v. CAS

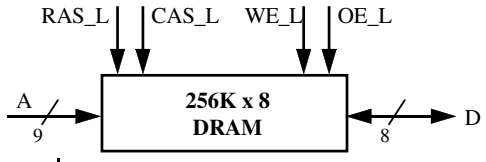


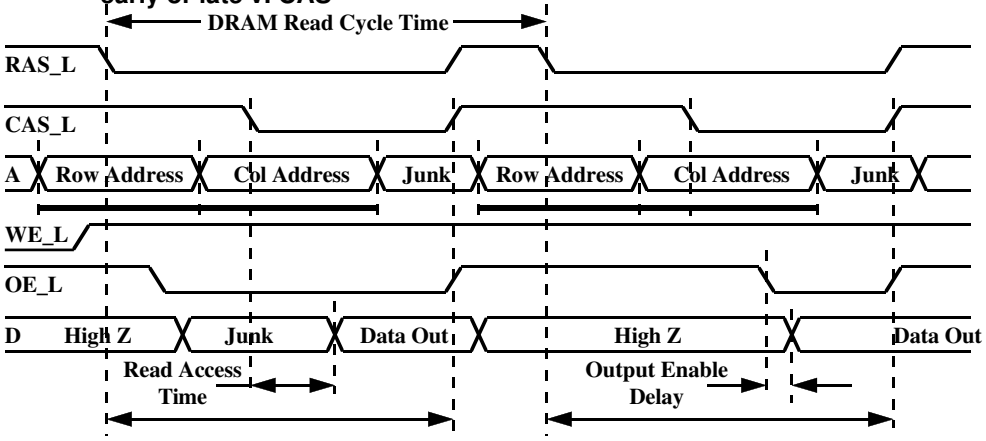


DAP Fa97, © U.CB

DRAM Read Timing

- **Every DRAM access begins at:**
  - The assertion of the RAS_L
  - 2 ways to read: early or late v. CAS



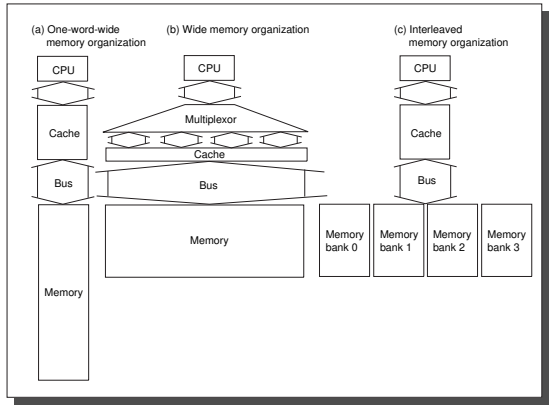


Early Read Cycle: OE_L asserted before CAS_L Late Read Cycle: OE_L asserted after CAS_L

DAP Fa97, © U.CB

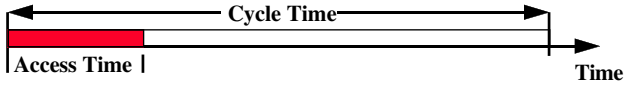
Main Memory Performance

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



DAP Fa97, © U.CB

Cycle Time versus Access Time

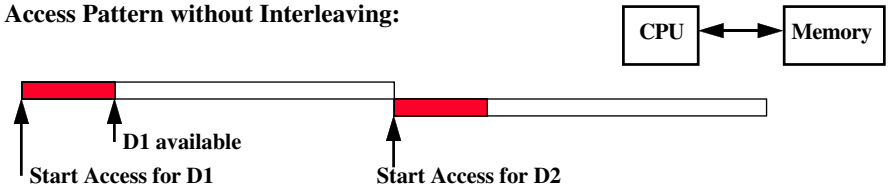


- **DRAM (Read/Write) Cycle Time >> DRAM (Read/ Write) Access Time**
  - ≈ 2:1; why?
- **DRAM (Read/Write) Cycle Time :**
  - How frequent can you initiate an access?
  - Analogy: A little kid can only ask his father for money on Saturday
- **DRAM (Read/Write) Access Time:**
  - How quickly will you get what you want once you initiate an access?
  - Analogy: As soon as he asks, his father will give him the money
- **DRAM Bandwidth Limitation analogy:**
  - What happens if he runs out of money on Wednesday?

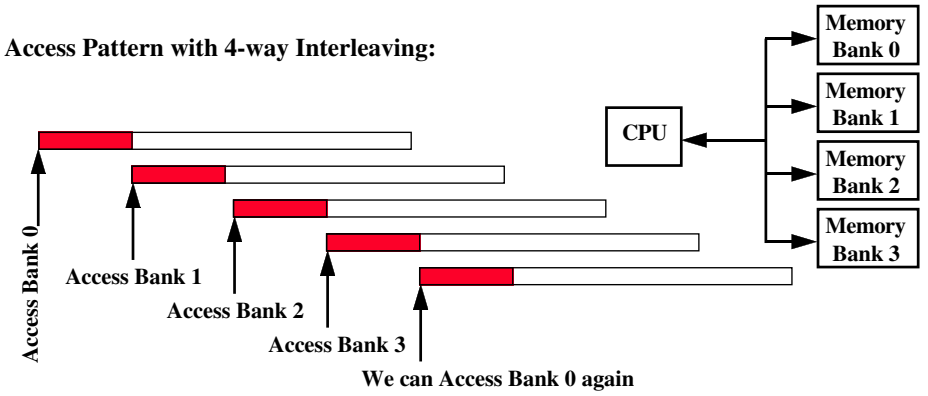
DAP Fa97, © U.CB

Increasing Bandwidth - Interleaving

Access Pattern without Interleaving:



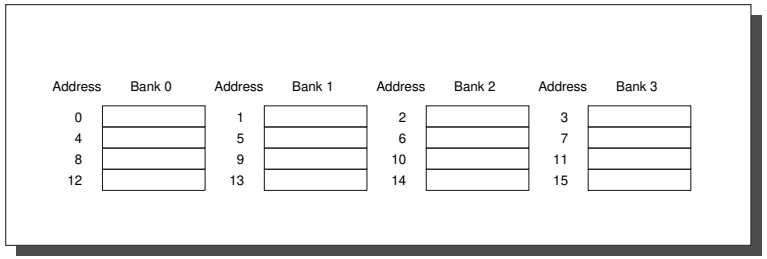
Access Pattern with 4-way Interleaving:



DAP Fa97, © U.CB

Main Memory Performance

- Timing model
  - 1 to send address,
  - 6 access time, 1 to send data
  - Cache Block is 4 words
- *Simple M.P.* = 4 x (1+6+1) = 32
- *Wide M.P.* = 1 + 6 + 1 = 8
- *Interleaved M.P.* = 1 + 6 + 4x1 = 11



DAP Fa97, © U.CB

Independent Memory Banks

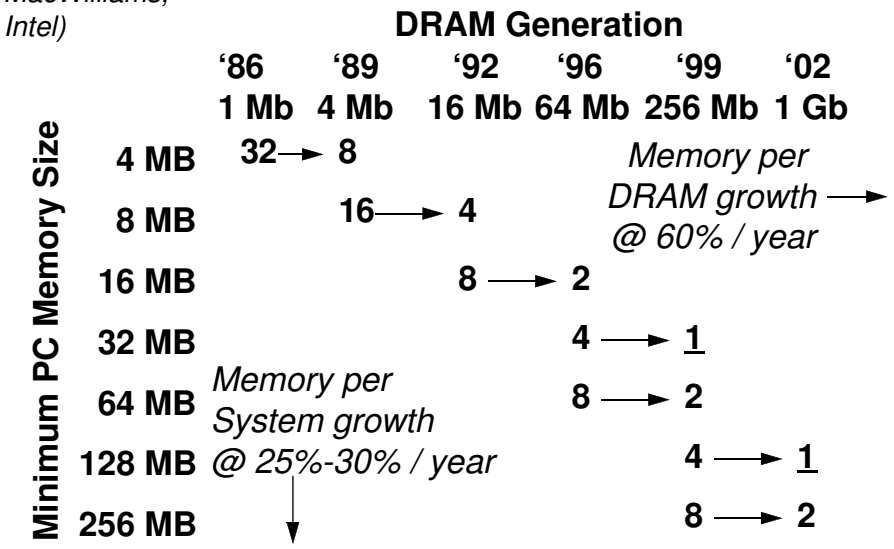
- How many banks?

  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 => harder to have banks
  - Growth bits/chip DRAM : 50%-60%/yr
  - Nathan Myrvold M/S: mature software growth (33%/yr for NT) ≈ growth MB/$ of DRAM (25%-30%/yr)

DAP Fa97, © U.CB

Fewer DRAMs/System over Time

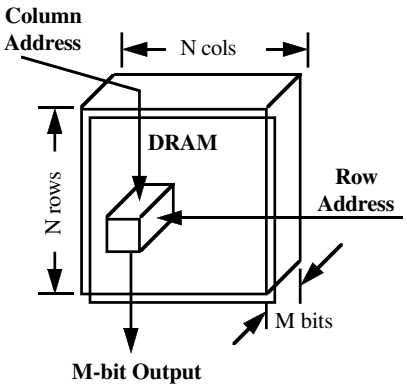
(from Pete MacWilliams, Intel)

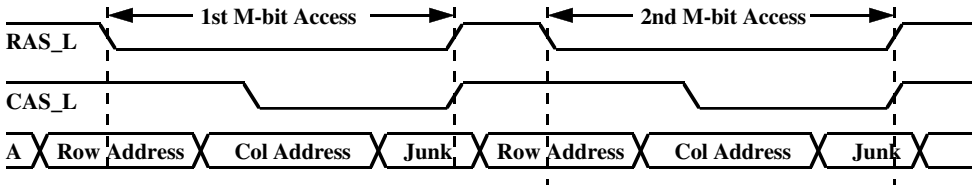


DAP Fa97, © U.CB

Page Mode DRAM: Motivation

- Regular DRAM Organization:
  - N rows x N column x M-bit
  - Read & Write M-bit at a time
  - Each M-bit access requires a RAS / CAS cycle
- Fast Page Mode DRAM
  - N x M "register" to save a row

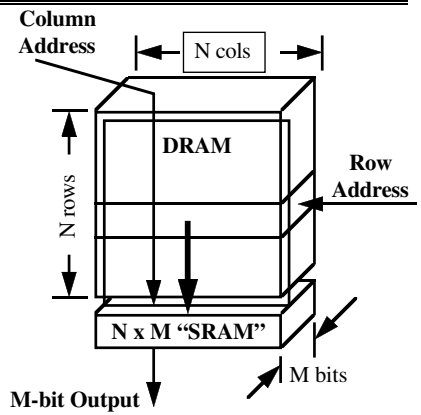


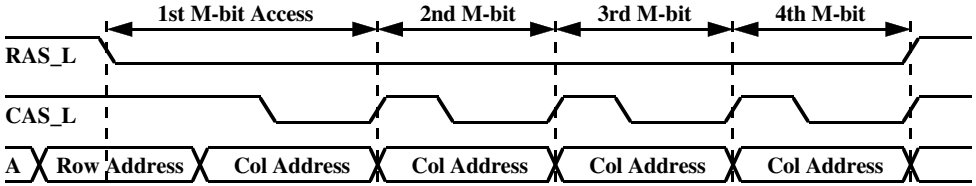


DAP Fa97, © U.CB

Fast Page Mode Operation

- Fast Page Mode DRAM
  - N x M "SRAM" to save a row
- After a row is read into the register
  - Only CAS is needed to access other M-bit blocks on that row
  - RAS_L remains asserted while CAS_L is toggled





DAP Fa97, © U.CB

DRAM v. Desktop Microprocessors Cultures

Standards	pinout, package, refresh rate, capacity, ...	binary compatibility, IEEE 754, I/O bus
Sources	Multiple	Single
Figures of Merit	1) capacity, 1a) $/bit 2) BW, 3) latency	1) SPEC speed 2) cost
Improve Rate/year	1) 60%, 1a) 25%, 2) 20%, 3) 7%	1) 60%, 2) little change

DAP Fa97, © U.CB

DRAM Design Goals

- Reduce cell size 2.5, increase die size 1.5
- Sell 10% of a single DRAM generation
  - 6.25 billion DRAMs sold in 1996
- 3 phases: engineering samples, first customer ship(FCS), mass production
  - Fastest to FCS, mass production wins share
- Die size, testing time, yield => profit
  - Yield >> 60% (redundant rows/columns to repair flaws)

DAP Fa97, © U.CB

DRAM History

- DRAMs: capacity +60%/yr, cost –30%/yr
  - 2.5X cells/area, 1.5X die size in ≈3 years
- '97 DRAM fab line costs $1B to $2B
  - DRAM only: density, leakage v. speed
- Rely on increasing no. of computers & memory per computer (60% market)
  - SIMM or DIMM is replaceable unit => computers use any generation DRAM
- Commodity, second source industry => high volume, low profit, conservative
  - Little organization innovation in 20 years page mode, EDO, Synch DRAM
- Order of importance: 1) Cost/bit 1a) Capacity
  - RAMBUS: 10X BW, +30% cost => little impact

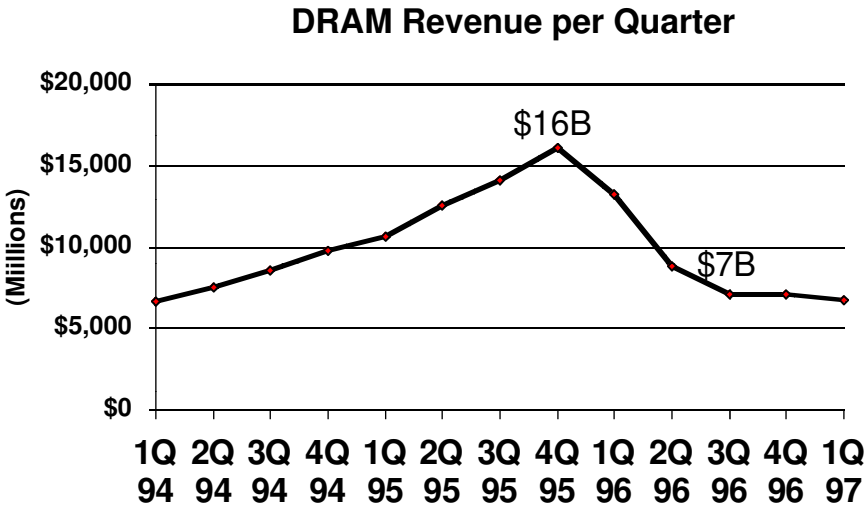
DAP Fa97, © U.CB

Today's Situation: DRAM

- Commodity, second source industry
  ⇒ high volume, low profit, conservative
  - Little organization innovation (vs. processors) in 20 years: page mode, EDO, Synch DRAM
- DRAM industry at a crossroads:
  - Fewer DRAMs per computer over time
    - Growth bits/chip DRAM : 50%-60%/yr
    - Nathan Myrvold M/S: mature software growth (33%/yr for NT) ≈ growth MB/$ of DRAM (25%-30%/yr)
  - Starting to question buying larger DRAMs?

DAP Fa97, © U.CB

Today's Situation: DRAM



- Intel: 30%/year since 1987; 1/3 income profit

DAP Fa97, © U.CB

Summary:

- Two Different Types of Locality:
  - Temporal Locality (Locality in Time): If an item is referenced, it will tend to be referenced again soon.
  - Spatial Locality (Locality in Space): If an item is referenced, items whose addresses are close by tend to be referenced soon.
- By taking advantage of the principle of locality:
  - Present the user with as much memory as is available in the cheapest technology.
  - Provide access at the speed offered by the fastest technology.
- DRAM is slow but cheap and dense:
  - Good choice for presenting the user with a BIG memory system
- SRAM is fast but expensive and not very dense:
  - Good choice for providing the user FAST access time.

DAP Fa97, © U.CB

Summary: Processor-Memory Performance Gap "Tax"

Processor	% Area *(≈cost)*	%Transistors *(≈power)*
Alpha 21164	37%	77%
StrongArm SA110	61%	94%
Pentium Pro	64%	88%

- 2 dies per package: Proc/I$/D$ + L2$
- Caches have no inherent value, only try to close performance gap

DAP Fa97, © U.CB