Performance & Technology

Todd C. Mowry
CS 740

Sept 15, 1998

Topics:

- Performance measures
- Relating performance measures
- Memory Technology
  - SRAM, DRAM
- Disk Technology

Performance expressed as a time

Absolute time measures

- difference between start and finish of an operation
- synonyms: running time, elapsed time, response time, latency, completion time, execution time
- most straightforward performance measure

Relative (normalized) time measures

- running time normalized to some reference time
- (e.g. time/reference time)

Guiding principle: Choose performance measures that track running time.

Performance expressed as a rate

Rates are performance measures expressed in units of work per unit time.

Examples:

- **millions of instructions / sec (MIPS)**
- **millions of floating point instructions / sec (MFLOPS)**
- **millions of bytes / sec (MBytes/sec)**
- **millions of bits / sec (Mbits/sec)**
- **images / sec**
- **samples / sec**
- **transactions / sec (TPS)**

Performance expressed as a rate(cont)

Key idea: Report rates that track execution time.

Example: Suppose we are measuring a program that convolves a stream of images from a video camera.

Bad performance measure: MFLOPS

- number of floating point operations depends on the particular convolution algorithm: n^2 matix-vector product vs nlogn fast Fourier transform. An FFT with a bad MFLOPS rate may run faster than a matrix-vector product with a good MFLOPS rate.

Good performance measure: images/sec

- a program that runs faster will convolve more images per second.

Performance expressed as a rate(cont)

Fallacy: Peak rates track running time.

Example: the i860 is advertised as having a peak rate of 80 MFLOPS (40 MHz with 2 flops per cycle).

However, the measured performance of some compiled linear algebra kernels (icc -O2) tells a different story:

Kernel	1d fft	sasum	saxpy	sdot	sgemm	sgemv	spvma
MFLOPS	8.5	3.2	6.1	10.3	6.2	15.0	8.1
%peak	11%	4%	7%	13%	8%	19%	10%

Relating time to system measures

Suppose that for some program we have:

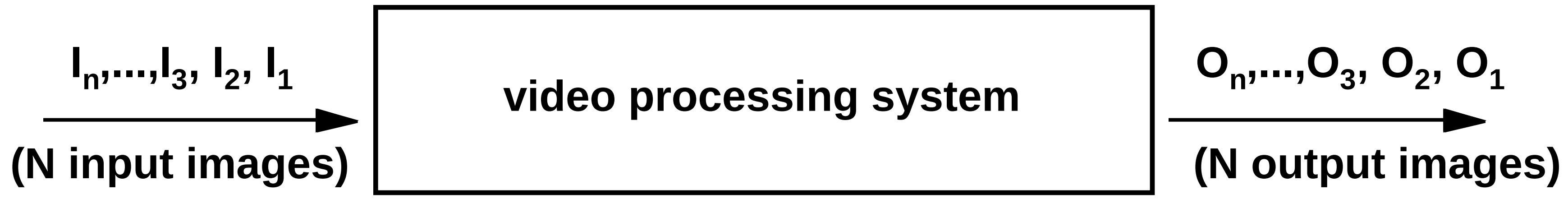
- **T seconds running time (the ultimate performance measure)**
- **C clock ticks, I instructions, P seconds/tick (performance measures of interest to the system designer)**

$$T \text{ secs} = C \text{ ticks} \times P \text{ secs/tick}$$

$$= (I \text{ inst}/I \text{ inst}) \times C \text{ ticks} \times P \text{ secs/tick}$$

$$\underbrace{T \text{ secs}}_{\text{running time}} = \underbrace{I \text{ inst}}_{\text{instruction count}} \times \underbrace{(C \text{ ticks}/I \text{ inst})}_{\text{avg clock ticks per instruction (CPI)}} \times \underbrace{P \text{ secs/tick}}_{\text{clock period}}$$

Pipeline latency and throughput



Latency (L): time to process an individual image.

Throughput (R): images processed per unit time

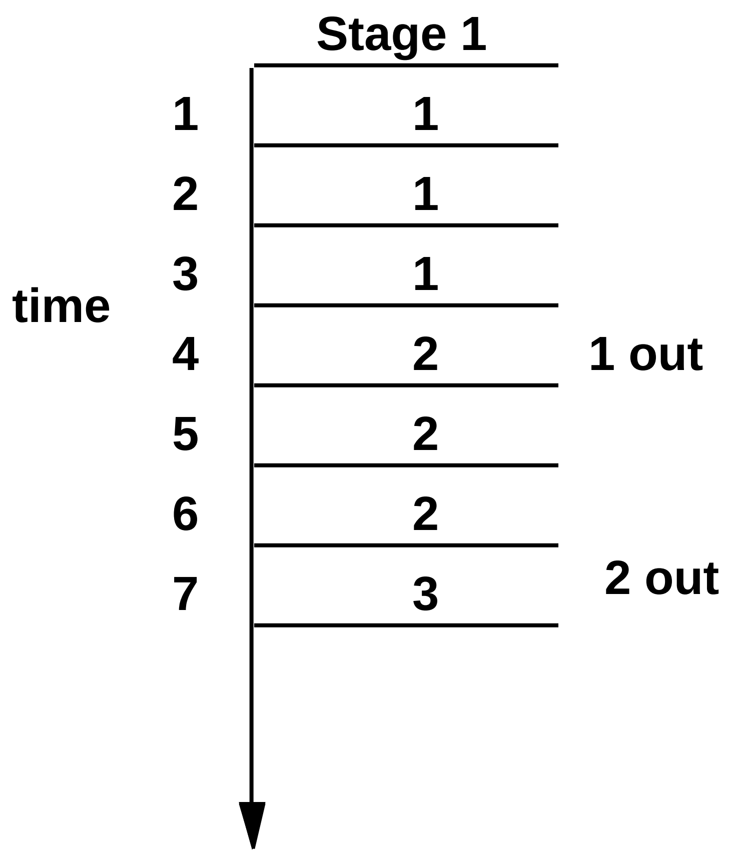
One image can be processed by the system at any point in time

Video system performance

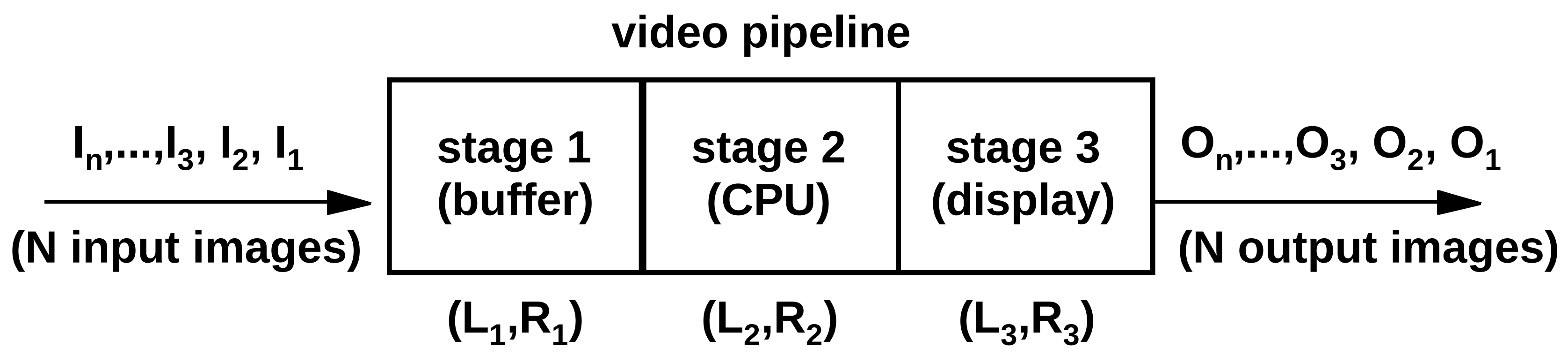
L = 3 secs/image.

R = 1/L = 1/3 images/sec.

T = L + (N-1)1/R
 = 3N



Pipelining the video system



One image can be in each stage at any point in time.

L_i = latency of stage i
R_i = throughput of stage i

$L = L_1 + L_2 + L_3$
$R = \min(R_1, R_2, R_3)$

Pipelined video system performance

Suppose:

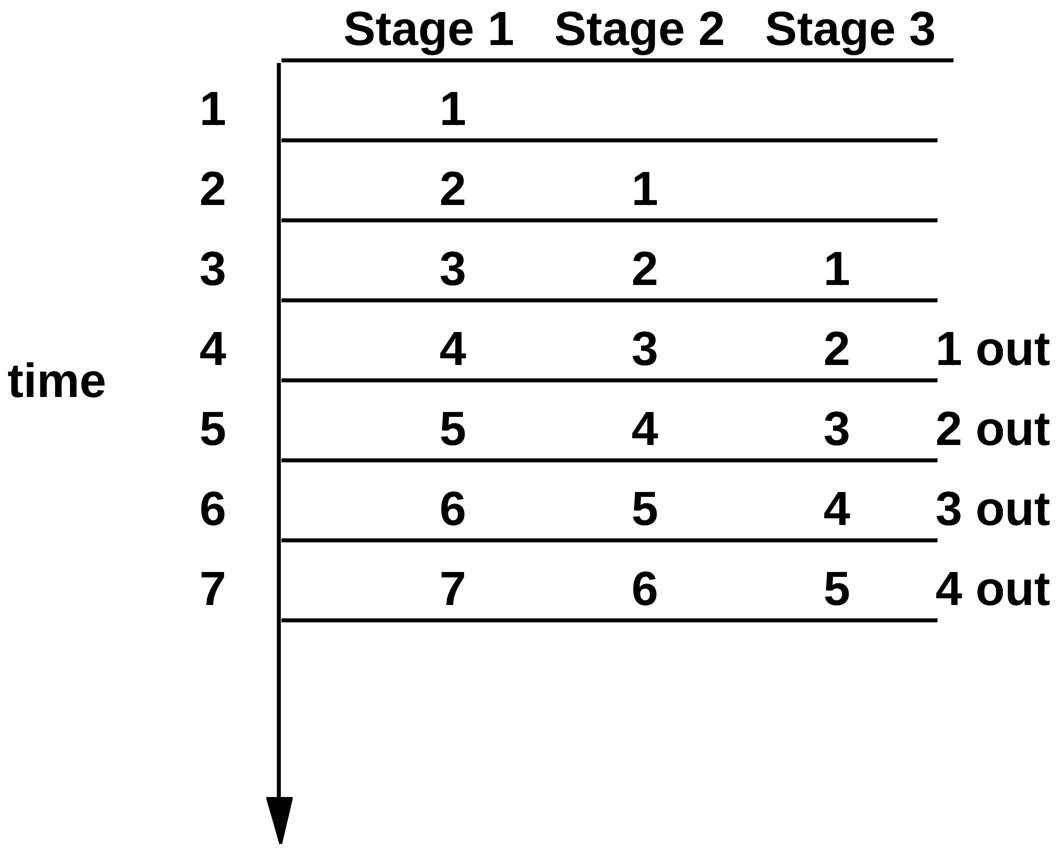
$L_1 = L_2 = L_3 = 1$

Then:

L = 3 secs/image.

R = 1 image/sec.

T = L + (N-1)1/R
   = N + 2



	Stage 1	Stage 2	Stage 3	
1	1			
2	2	1		
3	3	2	1	
4	4	3	2	1 out
5	5	4	3	2 out
6	6	5	4	3 out
7	7	6	5	4 out

Relating time to latency and thruput

In general:

- **$T = L + (N-1)/R$**

The impact of latency and throughput on running time depends on N:

- **$(N = 1) \Rightarrow (T = L)$**
- **$(N \gg 1) \Rightarrow (T = N-1/R)$**

To maximize throughput, we should try to maximize the minimum throughput over all stages (i.e., we strive for all stages to have equal throughput).

Amdahl’s law

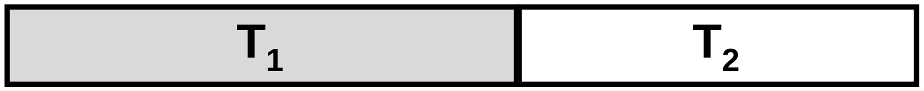
You plan to visit a friend in Normandy France and must decide whether it is worth it to take the Concorde SST ($3,100) or a 747 ($1,021) from NY to Paris, assuming it will take 4 hours Pgh to NY and 4 hours Paris to Normandy.

	time NY->Paris	total trip time	speedup over 747
747	8.5 hours	16.5 hours	1
SST	3.75 hours	11.75 hours	1.4

Taking the SST (which is 2.2 times faster) speeds up the overall trip by only a factor of 1.4!

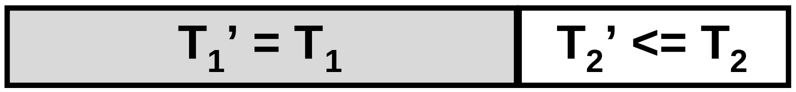
Amdahl's law (cont)

Old program (unenhanced)



Old time: $T = T_1 + T_2$

New program (enhanced)



New time: $T' = T_1' + T_2'$

T_1 = time that can NOT be enhanced.

T_2 = time that can be enhanced.

T_2' = time after the enhancement.

Speedup: $S_{overall} = T / T'$

Amdahl's law (cont)

Two key parameters:

$F_{enhanced} = T_2 / T$ (fraction of original time that can be improved)
$S_{enhanced} = T_2 / T_2'$ (speedup of enhanced part)

$$\begin{aligned} T' &= T_1' + T_2' = T_1 + T_2' = T(1-F_{enhanced}) + T_2' \\ &= T(1-F_{enhanced}) + (T_2/S_{enhanced}) && \text{[by def of } S_{enhanced}\text{]} \\ &= T(1-F_{enhanced}) + T(F_{enhanced}/S_{enhanced}) && \text{[by def of } F_{enhanced}\text{]} \\ &= T((1-F_{enhanced}) + F_{enhanced}/S_{enhanced}) \end{aligned}$$

Amdahl's Law:

$$S_{overall} = T / T' = 1/((1-F_{enhanced}) + F_{enhanced}/S_{enhanced})$$

Key idea: Amdahl's law quantifies the general notion of diminishing returns. It applies to any activity, not just computer programs.

Amdahl’s law (cont)

Trip example: Suppose that for the New York to Paris leg, we now consider the possibility of taking a rocket ship (15 minutes) or a handy rip in the fabric of space-time (0 minutes):

	time NY->Paris	total trip time	speedup over 747
747	8.5 hours	16.5 hours	1
SST	3.75 hours	11.75 hours	1.4
rocket	0.25 hours	8.25 hours	2.0
rip	0.0 hours	8 hours	2.1

Amdahl’s law (cont)

Useful corollary to Amdahl’s law:

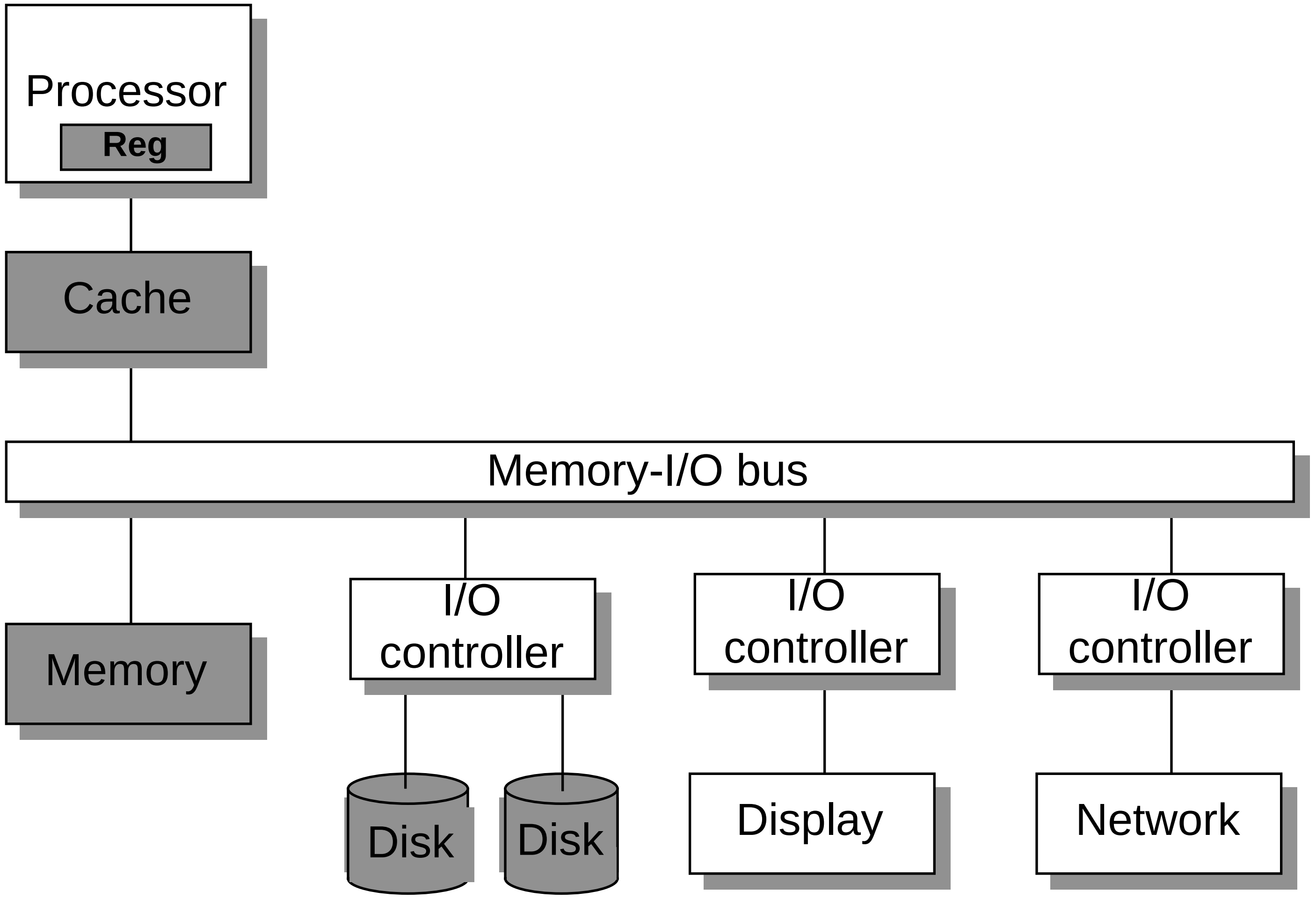
- $1 \le S_{overall} \le 1 / (1 - F_{enhanced})$

$F_{enhanced}$	Max $S_{overall}$	$F_{enhanced}$	Max $S_{overall}$
0.0	1	0.9375	16
0.5	2	0.96875	32
0.75	4	0.984375	64
0.875	8	0.9921875	128

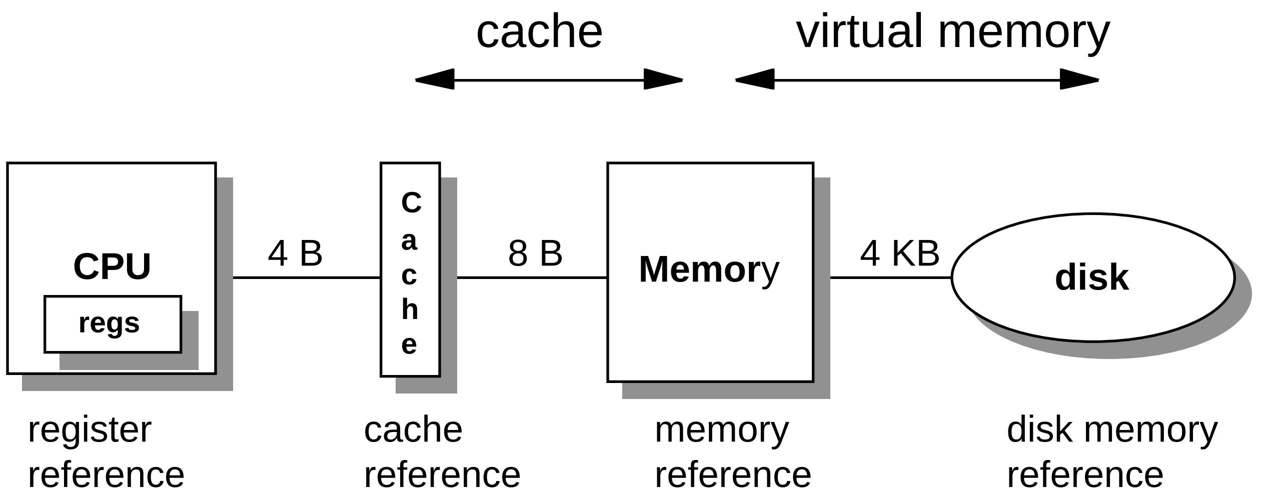
Moral: It is hard to speed up a program.

Moral++ : It is easy to make premature optimizations.

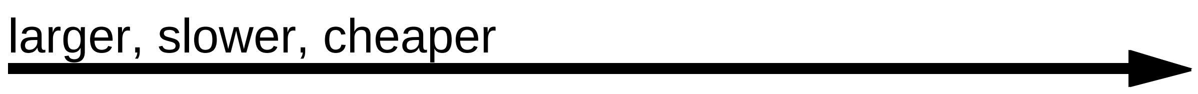
Computer System



Levels in a typical memory hierarchy



size:	200 B	32 KB / 4MB	128 MB	20 GB
speed:	3 ns	6 ns	100 ns	10 ms
$/Mbyte:		$256/MB	$2/MB	$0.8/MB
block size:	4 B	8 B	4 KB	



Scaling to 0.1µm

- **Semiconductor Industry Association, 1992 Technology Workshop**

Year	1992	1995	1998	2001	2004	2007
Feature size	0.5	0.35	0.25	0.18	0.12	0.10
DRAM cap	16M	64M	256M	1G	4G	16G
Gates/chip	300K	800K	2M	5M	10M	20M
Chip cm^2	2.5	4.0	6.0	8.0	10.0	12.5
I/Os	500	750	1500	2000	3500	5000
off chip MHz	60	100	175	250	350	500
on chip MHz	120	200	350	500	700	1000

Static RAM (SRAM)

Fast

- **~10 ns [1995]**

Persistent

- **as long as power is supplied**
- **no refresh required**

Expensive

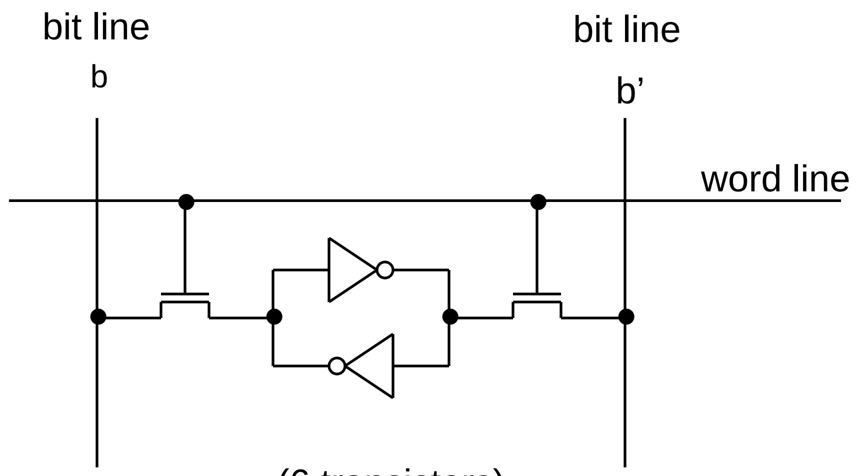
- **~$256/MByte [1995]**
- **6 transistors/bit**

Stable

- **High immunity to noise and environmental disturbances**

Technology for caches

Anatomy of an SRAM bit (cell)



(6 transistors)

Read:
- set bit lines high
- set word line high
- see which bit line goes low

Write:
- set bit lines to opposite values
- set word line
- Flip cell to new state

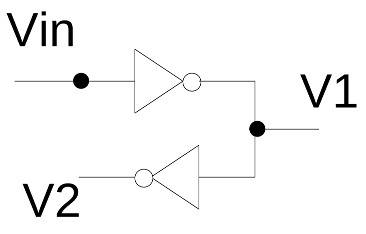
SRAM Cell Principle

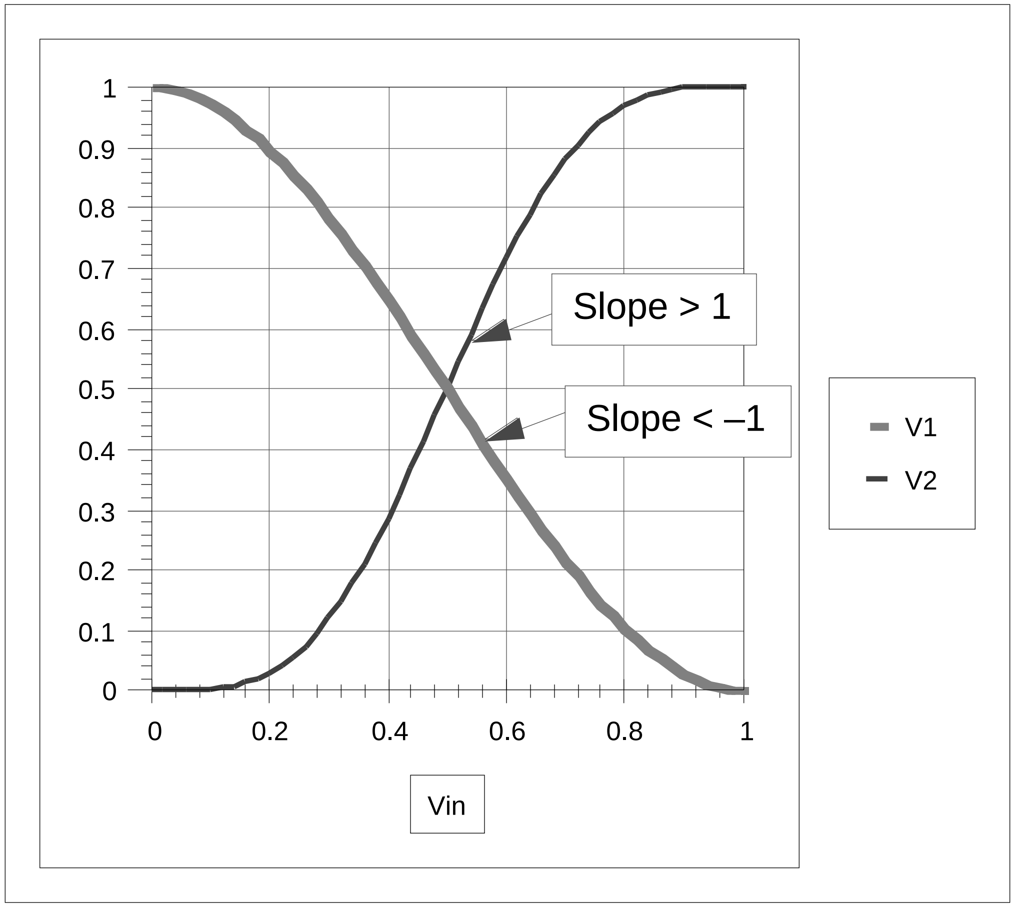
Inverter Amplifies

- Negative gain
- Slope < –1 in middle
- Saturates at ends

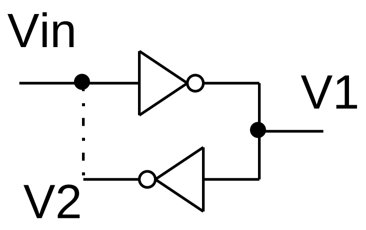
Inverter Pair Amplifies

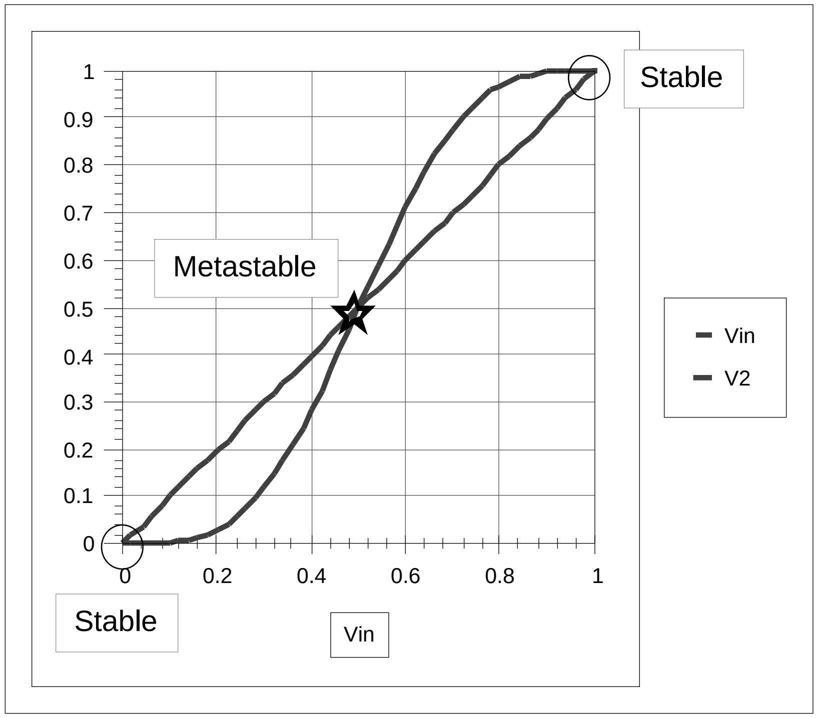
- Positive gain
- Slope > 1 in middle
- Saturates at ends





Bistable Element

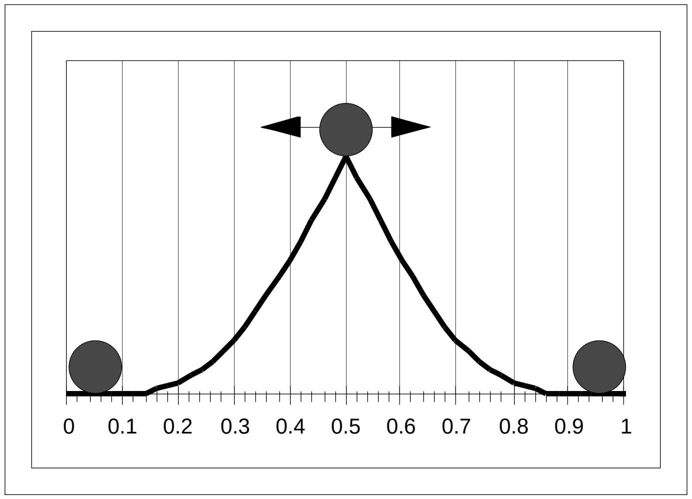




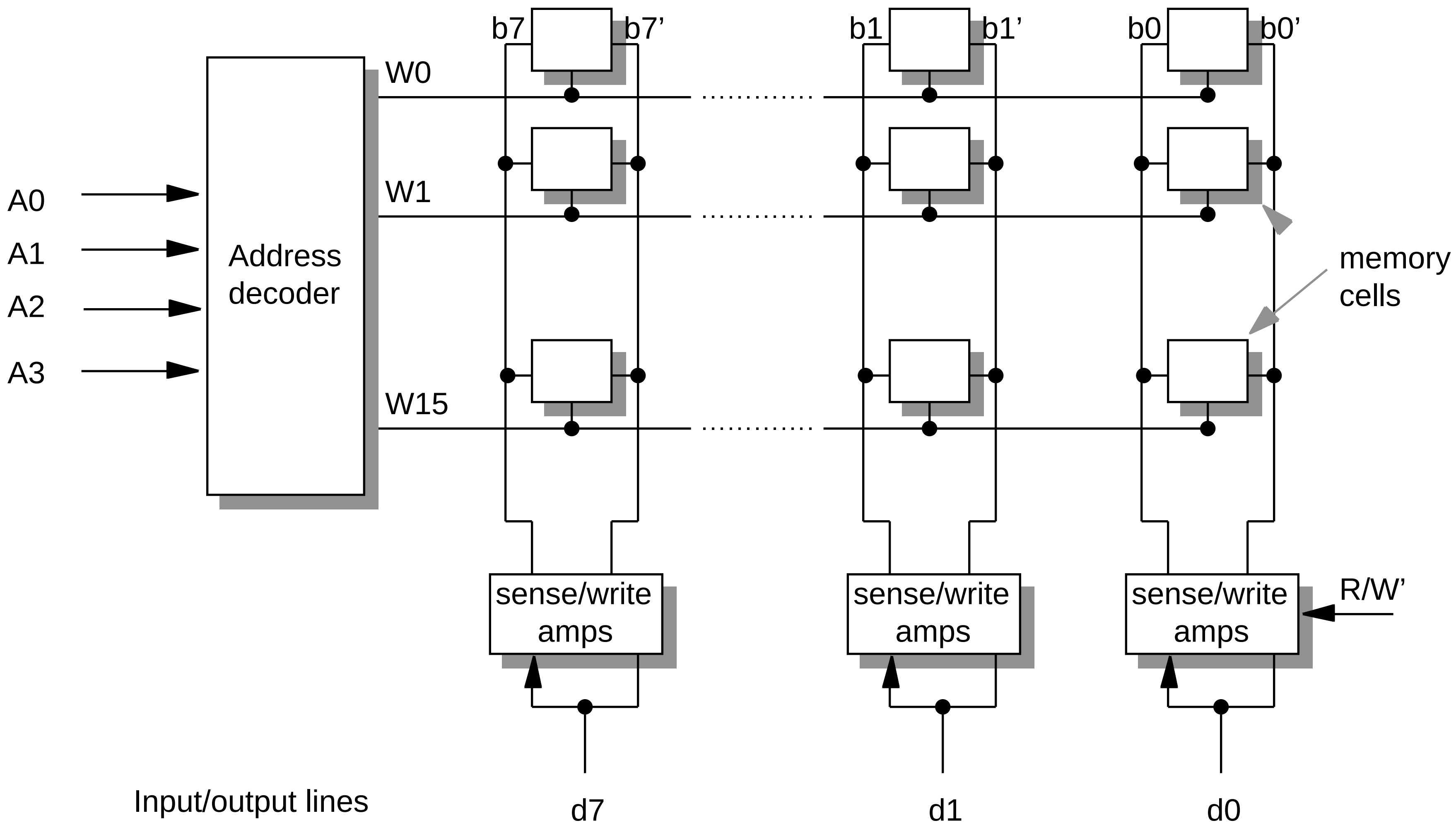
Stability

- **Require Vin = V2**
- **Stable at endpoints**
  - recover from pertubation
- **Metastable in middle**
  - Fall out when perturbed

Ball on Ramp Analogy



Example 1-level-decode SRAM (16 x 8)



Dynamic RAM (DRAM)

Slower than SRAM

- access time ~70 ns [1995]

Nonpersistant

- every row must be accessed every ~1 ms (refreshed)

Cheaper than SRAM

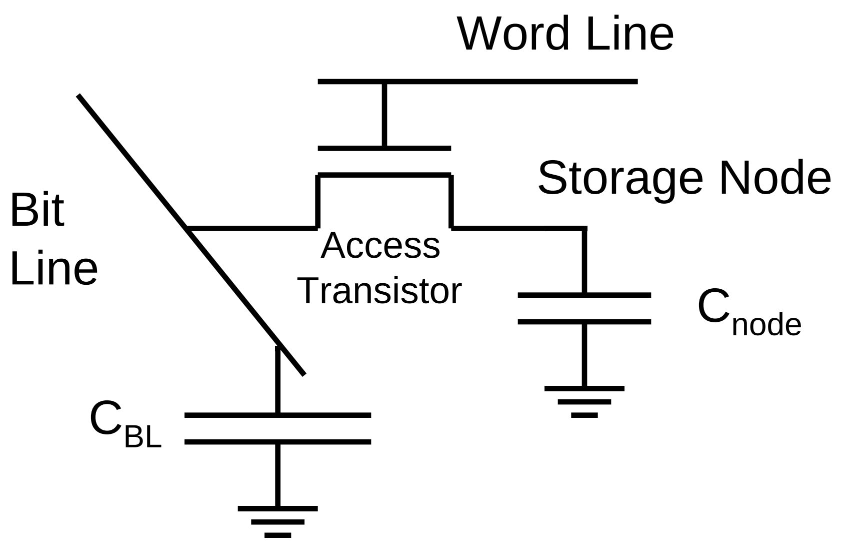
- ~$2/MByte [1997]
- 1 transistor/bit

Fragile

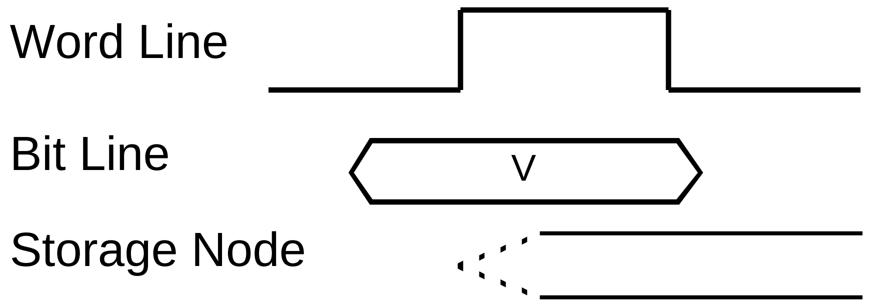
- electrical noise, light, radiation

Workhorse memory technology

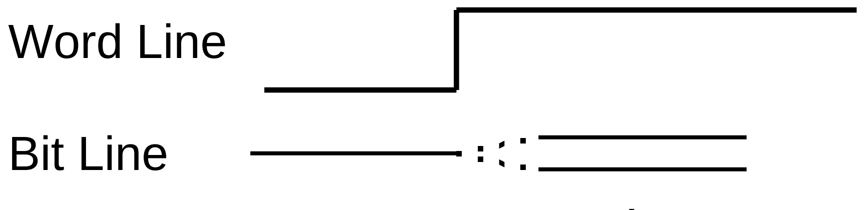
Anatomy of a DRAM Cell



Writing



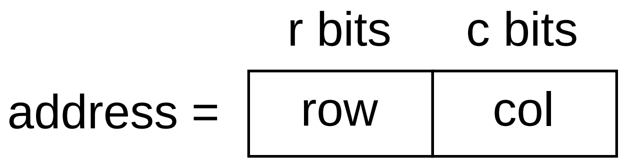
Reading



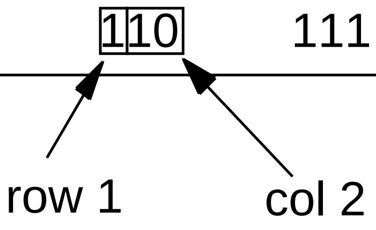
$\Delta V \sim C_{node} / C_{BL}$

Addressing arrays with bits

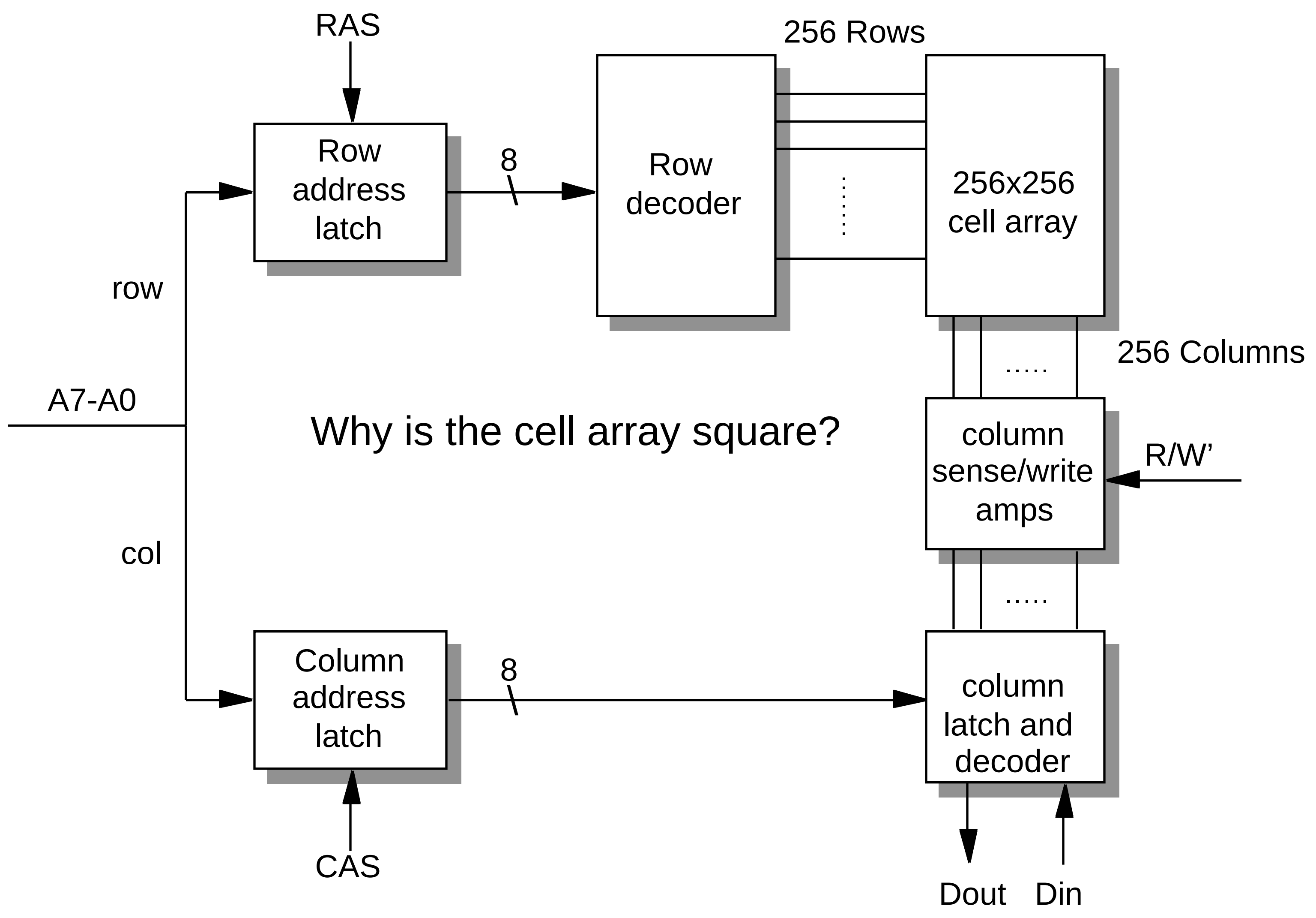
Consider an R x C array of addresses, where R = 2^r and C = 2^c.
Then for each address,
  row(address) = address / C = leftmost r bits of address
  col(address) = address % C = righmost c bits of address



	0	1	2	3
0	000	001	010	011
1	100	101	$\boxed{1}\boxed{10}$	111



Example 2-level decode DRAM (64Kx1)



Why is the cell array square?

DRAM Operation

Row Address (~50ns)

- **Set Row address on address lines & strobe RAS**
- **Entire row read & stored in column latches**
- **Contents of row of memory cells destroyed**

Column Address (~10ns)

- **Set Column address on address lines & strobe CAS**
- **Access selected bit**
  - READ: transfer from selected column latch to Dout
  - WRITE: Set selected column latch to Din

Rewrite (~30ns)

- **Write back entire row**

Observations About DRAMs

Timing

- Access time = 60ns < cycle time = 90ns
- Need to rewrite row

Must Refresh Periodically

- Perform complete memory cycle for each row
- Approx. every 1ms
- Sqrt(n) cycles
- Handled in background by memory controller

Inefficient Way to Get Single Bit

- Effectively read entire row of Sqrt(n) bits

Enhanced Performance DRAMs

Conventional Access

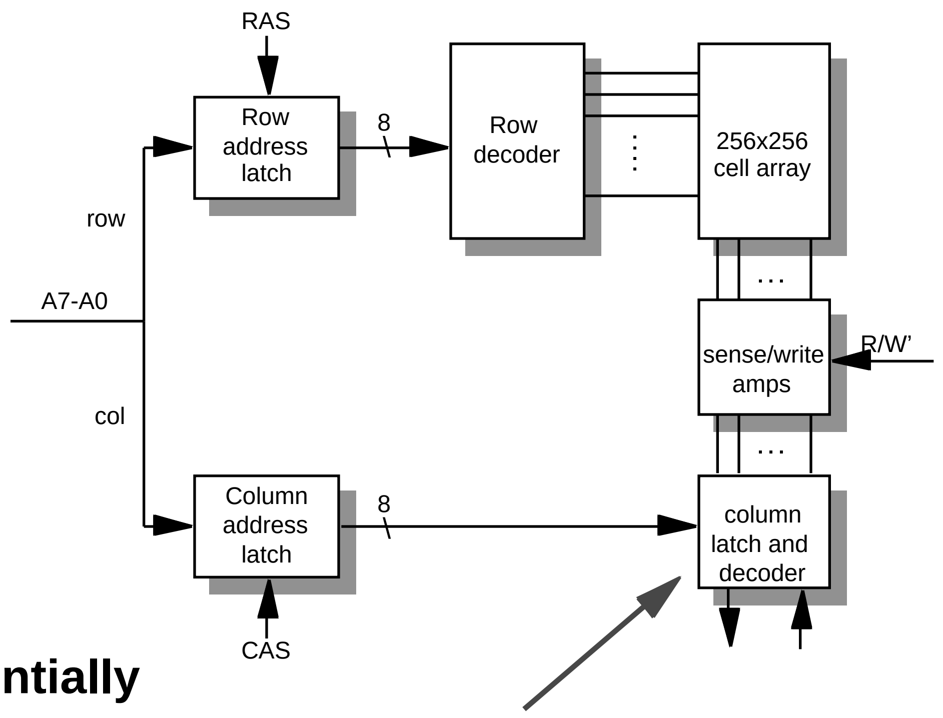
- Row + Col
- RAS CAS RAS CAS ...

Page Mode

- Row + Series of columns
- RAS CAS CAS CAS ...
- Gives successive bits

Video RAM

- Shift out entire row sequentially
- At video rate



Entire row buffered here

Typical Performance

row access time	col access time	cycle time	page mode cycle time
50ns	10ns	90ns	25ns

DRAM Driving Forces

Capacity

- **4X per generation**
  - Square array of cells
- **Typical scaling**
  - Lithography dimensions 0.7X
    - » Areal density 2X
  - Cell function packing 1.5X
  - Chip area 1.33X
- **Scaling challenge**
  - Typically $C_{node} / C_{BL} = 0.1–0.2$
  - Must keep C_{node} high as shrink cell size

Retention Time

- **Typically 16–256 ms**
- **Want higher for low-power applications**

DRAM Storage Capacitor

Planar Capacitor

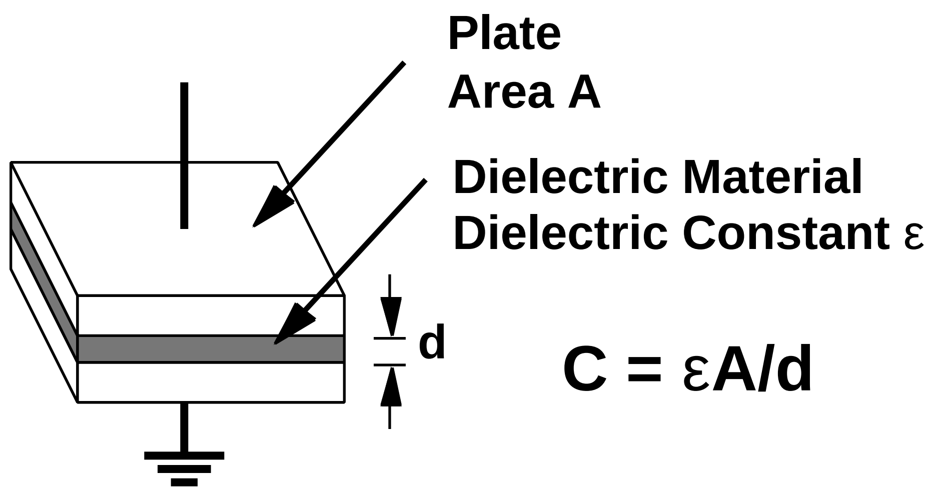
- Up to 1Mb
- C decreases linearly with feature size

Trench Capacitor

- 4–256 Mb
- Lining of hole in substrate

Stacked Cell

- > 1Gb
- On top of substrate
- Use high ε dielectric

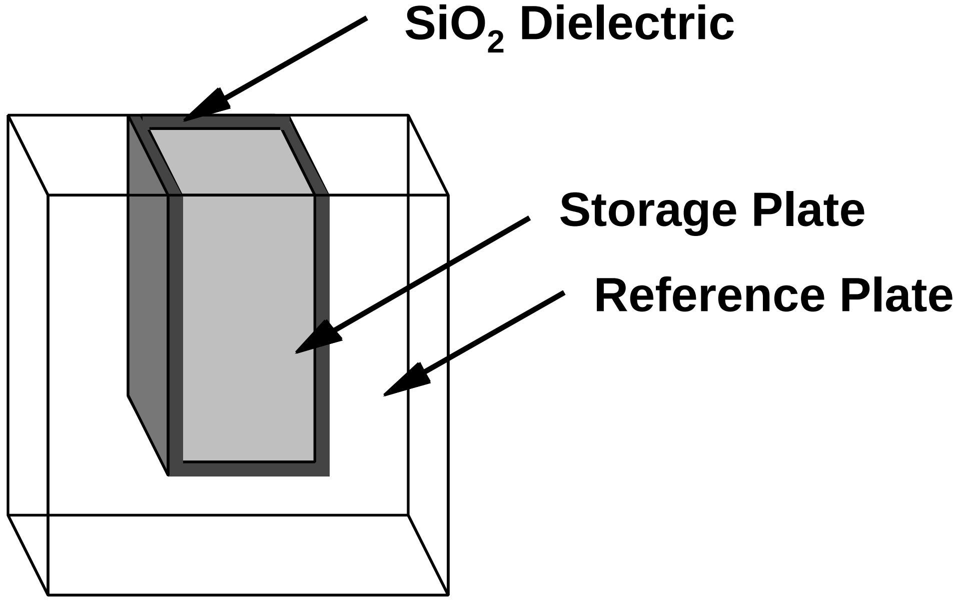


$$C = \varepsilon A/d$$

Trench Capacitor

Process

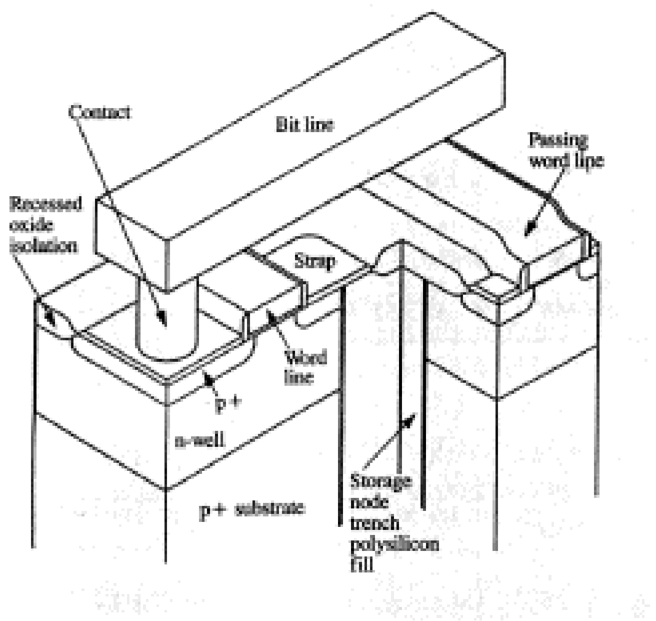
- **Etch deep hole in substrate**
  - Becomes reference plate
- **Grow oxide on walls**
  - Dielectric
- **Fill with polysilicon plug**
  - Tied to storage node



IBM DRAM Evolution

- IBM J. R&D, Jan/Mar '95
- Evolution from 4 – 256 Mb
- 256 Mb uses cell with area 0.6 µm^2

4 Mb Cell Structure



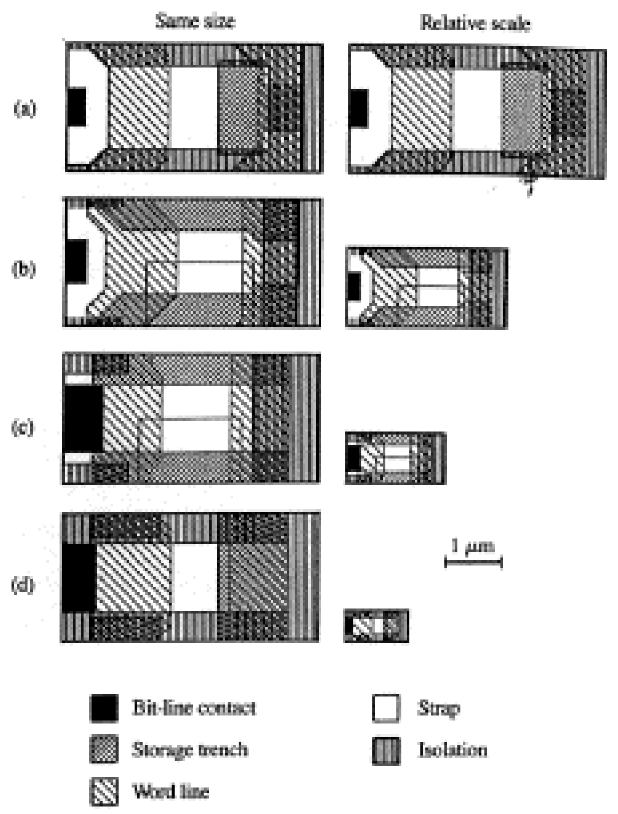
Cell Layouts

4Mb

16Mb

64Mb

256Mb



Mitsubishi Stacked Cell DRAM

- **IEDM '95**
- **Claim suitable for 1 – 4 Gb**

Technology

- **0.14 µm process**
  - Synchrotron X-ray source
- **8 nm gate oxide**
- **0.29 µm² cell**

Storage Capacitor

- **Fabricated on top of everything else**
- **Rubidium electrodes**
- **High dielectric insulator**
  - 50X higher than SiO_2
  - 25 nm thick
- **Cell capacitance 25 femtofarads**

Cross Section of 2 Cells

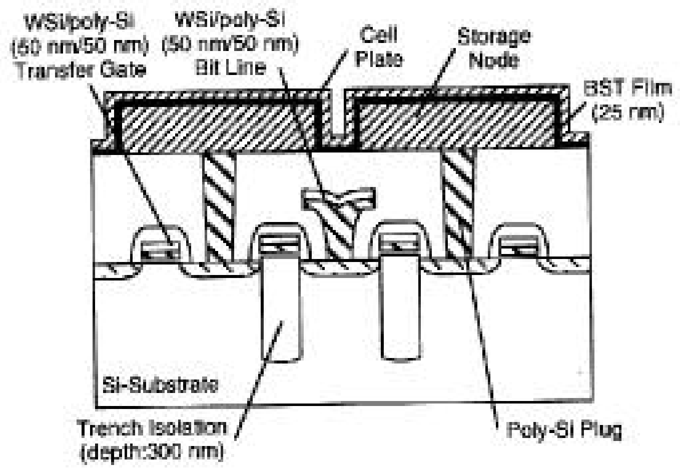

WSi/poly-Si (50 nm/50 nm) Transfer Gate
WSi/poly-Si (50 nm/50 nm) Bit Line
Cell Plate
Storage Node
BST Film (25 nm)
Si-Substrate
Trench Isolation (depth:300 nm)
Poly-Si Plug

Fig. 2 Schematic cross-sectional view of DRAM memory cells with Ru/BST/Ru stacked capacitors.

Mitsubishi DRAM Pictures

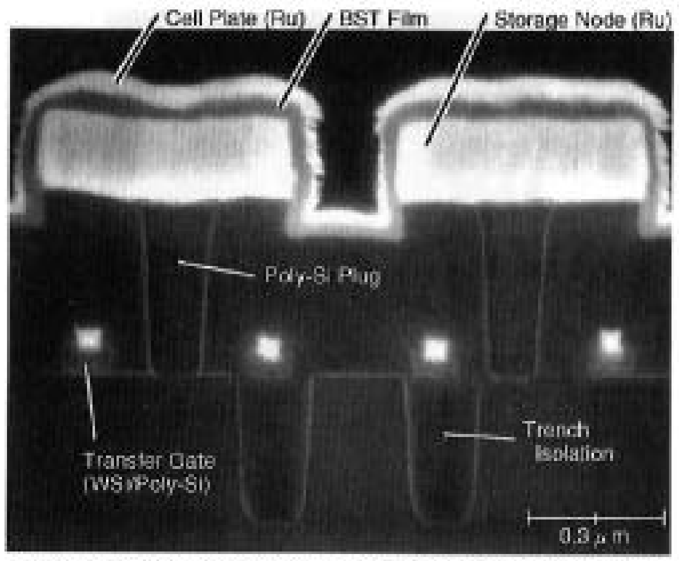


Fig. 3 SEM cross-sectional photograph of the fabricated 0.29-μm² memory cell with Ru/BST/Ru stacked capacitor. The facet was fabricated by focused ion beam etching.

- Active Area - Transfer Gate

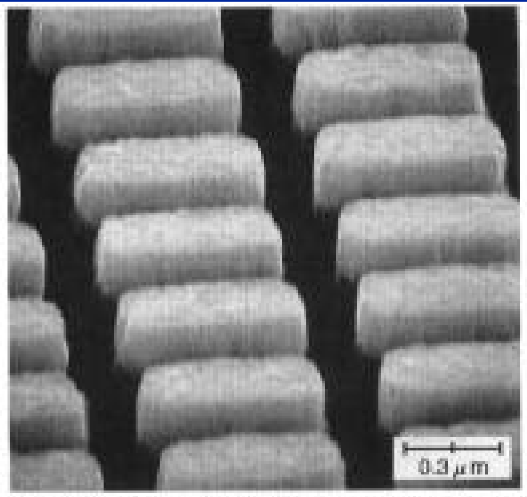


Fig. 8 SEM photograph of a Ru-metal storage node array with a projection a height of 0.2 μm.

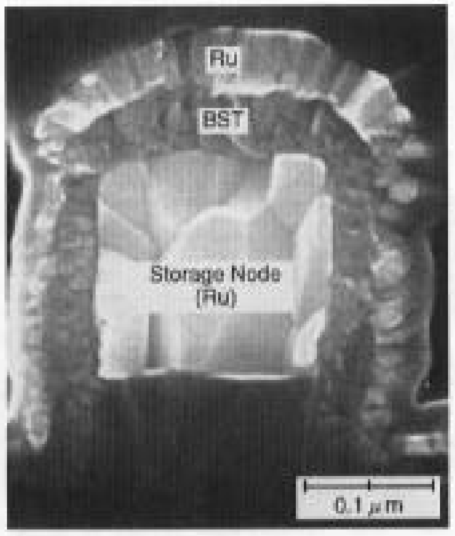
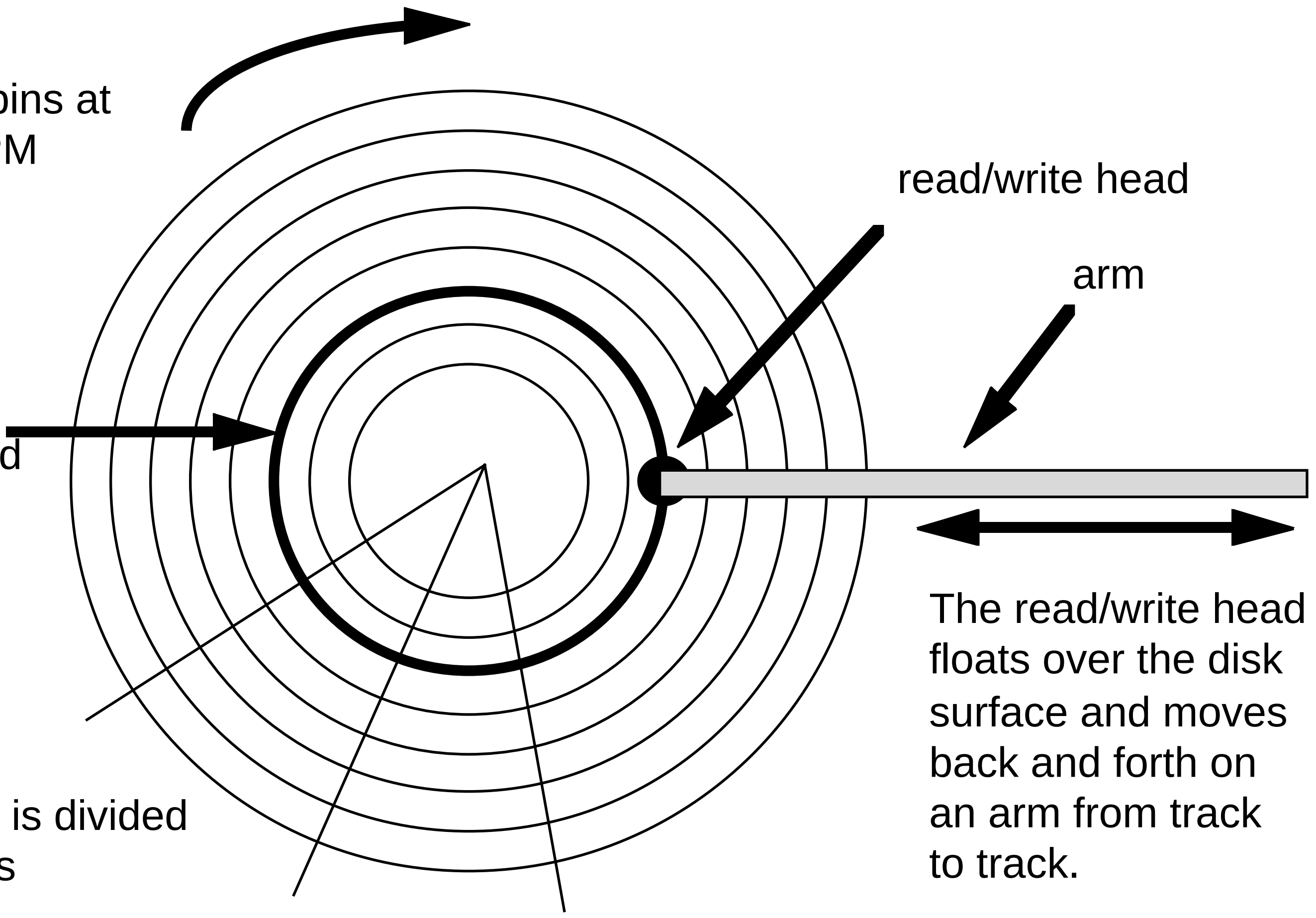


Fig. 10 SEM cross-sectional view of a Ru/BST/Ru capacitor cell. The facet shown is a cleaved facet.

Magnetic Disks

Disk surface spins at
3600–7200 RPM

The surface consists
of a set of concentric
magnetized rings called
tracks



The read/write head
floats over the disk
surface and moves
back and forth on
an arm from track
to track.

Each track is divided
into sectors

Disk Capacity

Parameter	540MB Example
• Number Platters	8
• Surfaces / Platter	2
• Number of tracks	1046
• Number sectors / track	63
• Bytes / sector	512
Total Bytes	**539,836,416**

Disk Operation

Operation

- Read or write complete sector

Seek

- Position head over proper track
- Typically 10ms

Rotational Latency

- Wait until desired sector passes under head
- Worst case: complete rotation
  - 3600RPM: 16.7 ms

Read or Write Bits

- Transfer rate depends on # bits per track and rotational speed
- E.g., 63 * 512 bytes @3600RPM = 1.9 MB/sec.

Disk Performance

Getting First Byte

- **Seek + Rotational latency 10,000 – 27,000 microseconds**

Getting Successive Bytes

- **~ 0.5 microseconds each**

Optimizing

- **Large block transfers more efficient**
- **Try to do other things while waiting for first byte**
  - Switch context to other computing task
  - Disk controller buffers sector
  - Interrupts processor when transfer completed

Disk Technology

Seagate ST-12550N Barracuda 2 Disk

• **Linear density**	**52,187.**	**bits per inch (BPI)**
– Bit spacing	0.5	microns
• **Track density**	**3,047.**	**tracks per inch (TPI)**
– Track spacing	8.3	microns
• **Total tracks**	**2,707.**	**tracks**
• **Rotational Speed**	**7200.**	**RPM**
• **Avg Linear Speed**	**86.4**	**kilometers / hour**
• **Head Floating Height**	**0.13**	**microns**

Analogy

- **Put Sears Tower on side**
- **Fly around world 2.5 cm off ground**
- **8 seconds per orbit**

Storage trends (memory)

SRAM

metric	1980	1985	1990	1995	1995:1980
$/MB	19,200	2,900	320	256	75
access (ns)	300	150	35	15	20

DRAM

metric	1980	1985	1990	1995	1995:1980
$/MB	8,000	880	100	30	266
access (ns)	375	200	100	70	5
typical size(MB)	0.064	0.256	4	16	250

culled from back issues of Byte and PC Magazine

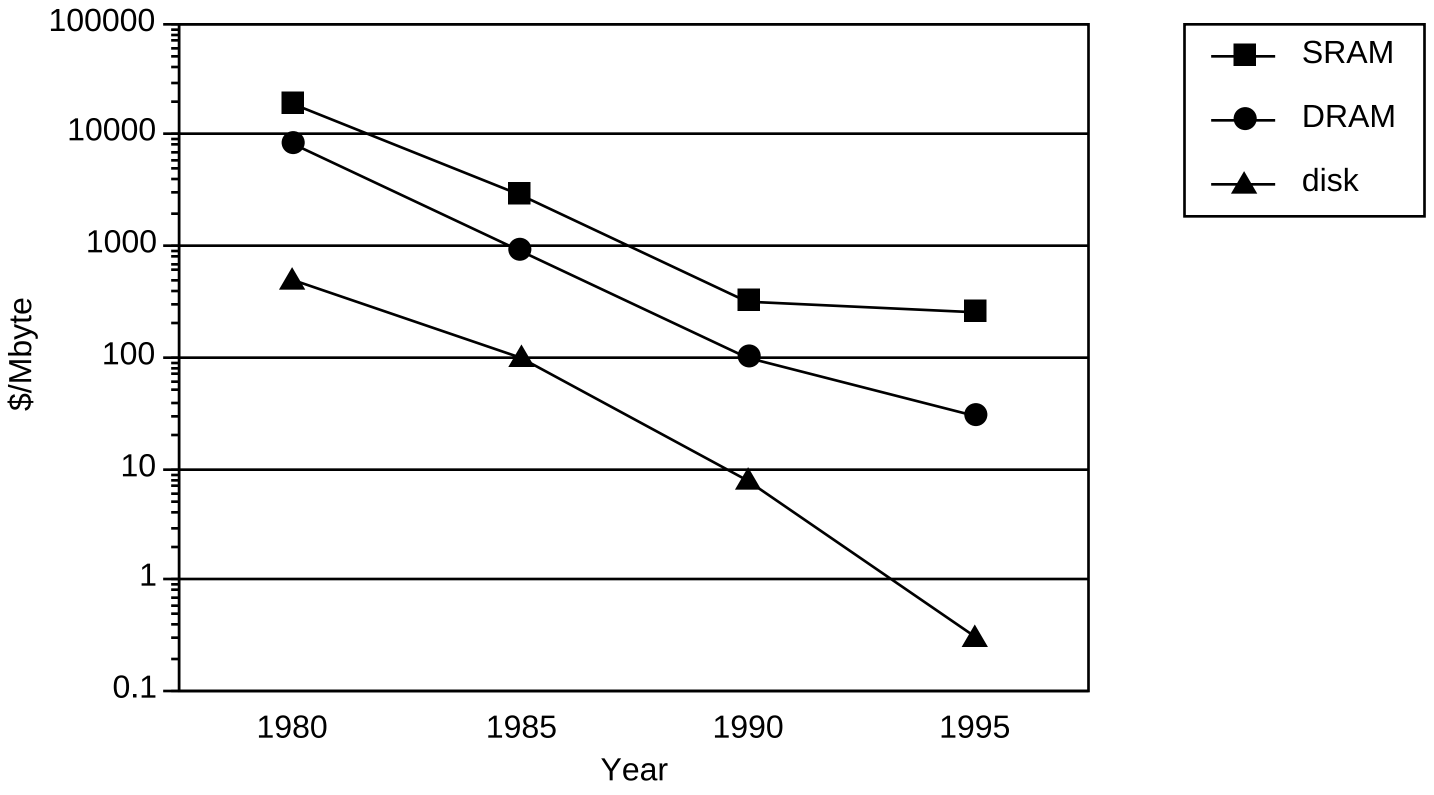
Storage trends (disk)

Disks

metric	1980	1985	1990	1995	1995:1980
$/MB	500	100	8	0.30	1,600
access (ms)	87	75	28	10	9
typical size(MB)	1	10	160	1,000	1,000

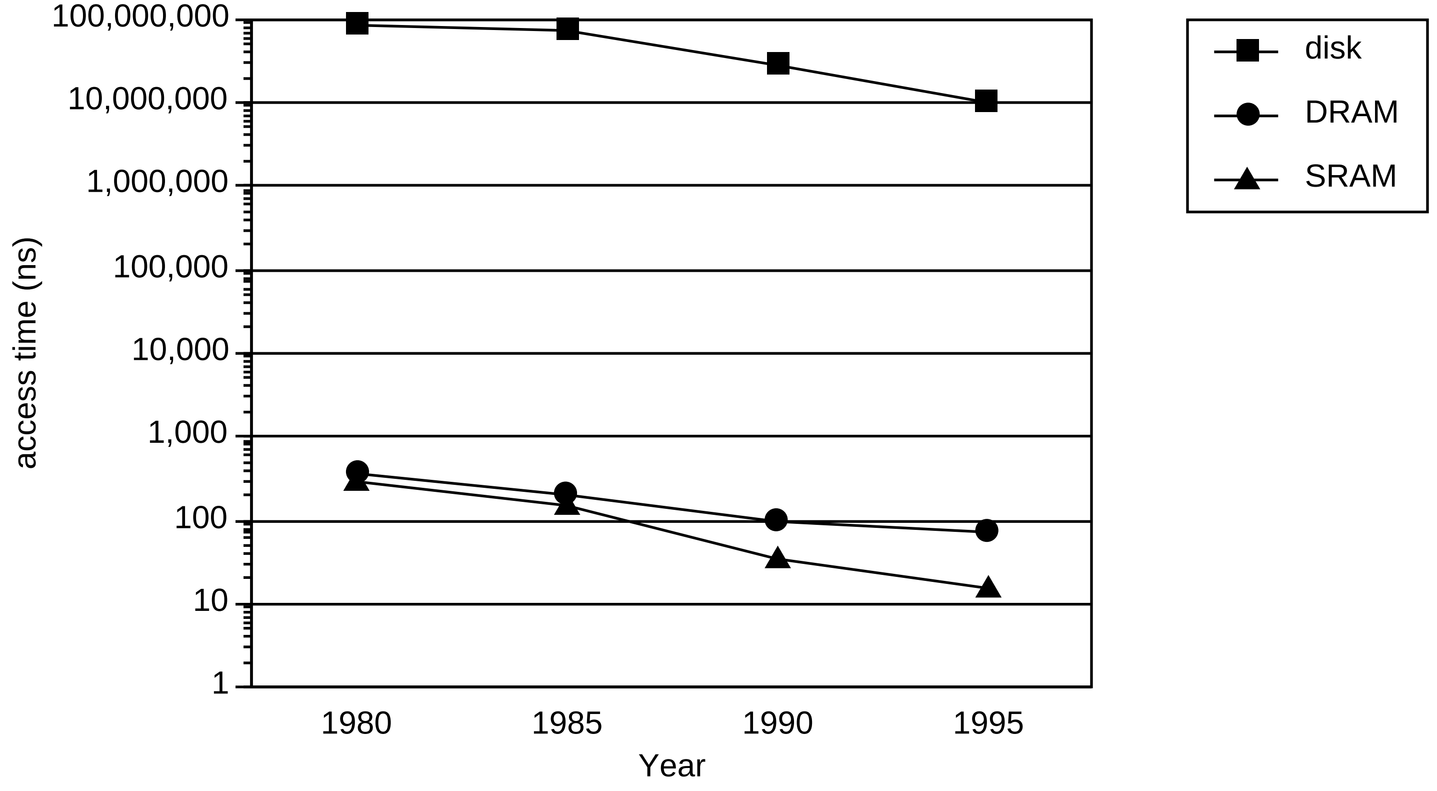
culled from back issues of Byte and PC Magazine

Storage price/MByte



culled from back issues of Byte and PC Magazine

Storage access times



culled from back issues of Byte and PC Magazine

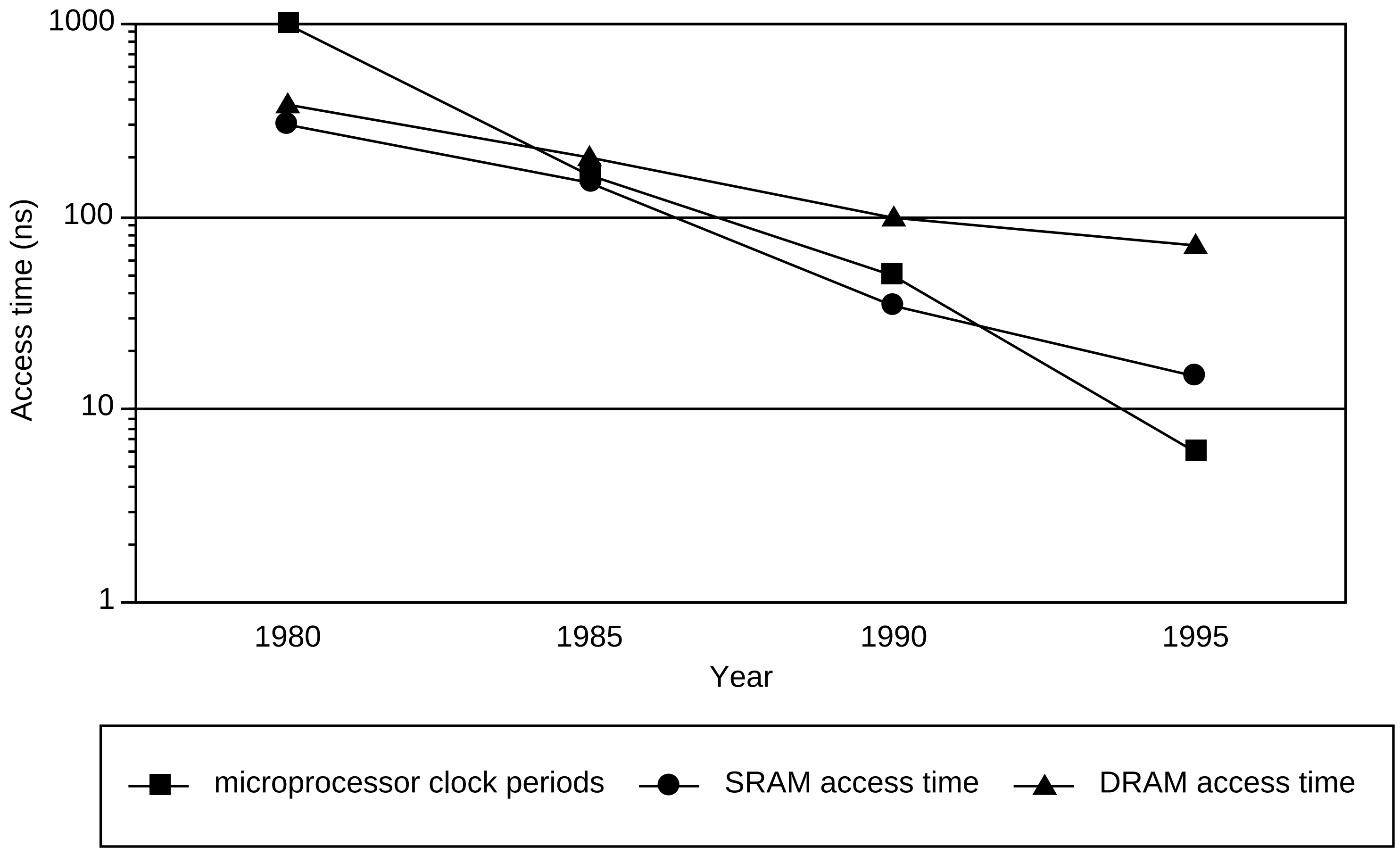
Processor clock rates

Processors

metric	1980	1985	1990	1995	1995:1980
typical clock(MHz)	1	6	20	150	150
processor	8080	286	386	pentium	

culled from back issues of Byte and PC Magazine

The widening processor/memory gap



culled from back issues of Byte and PC Magazine

Memory technology summary

Cost and density improving at enormous rates.

Speed lagging processor performance

Memory hierarchies help narrow the gap:

- small fast SRAMS (cache) at upper levels
- large slow DRAMS (main memory) at lower levels
- Incredibly large & slow disks to back it all up

Locality of reference makes it all work

- Keep most frequently accessed data in fastest memory