Advanced Computer Architecture
Department of Computing, Imperial College London
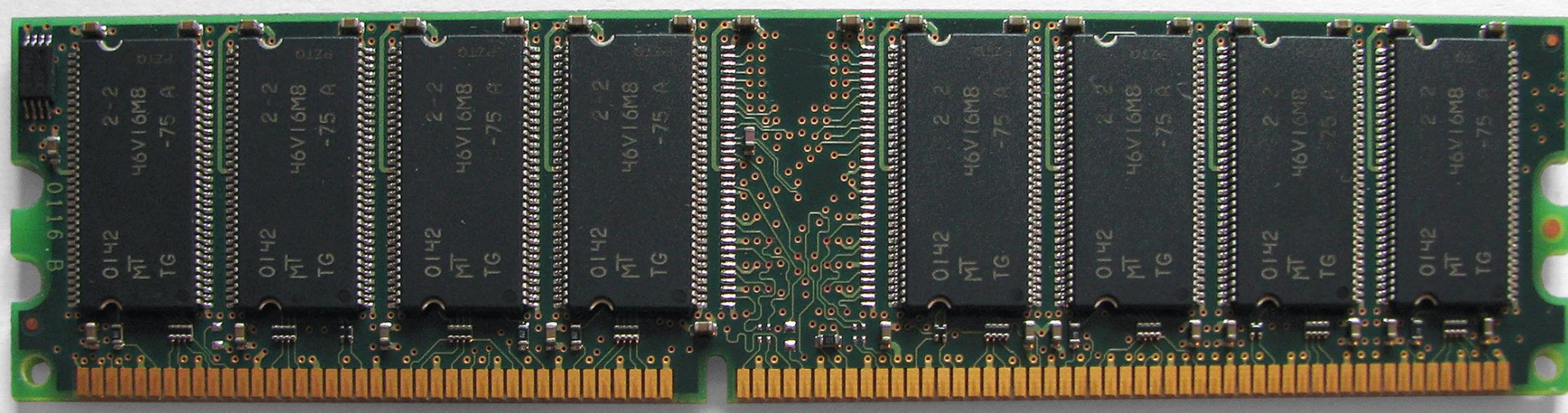
Chapter 4: Caches and Memory Systems
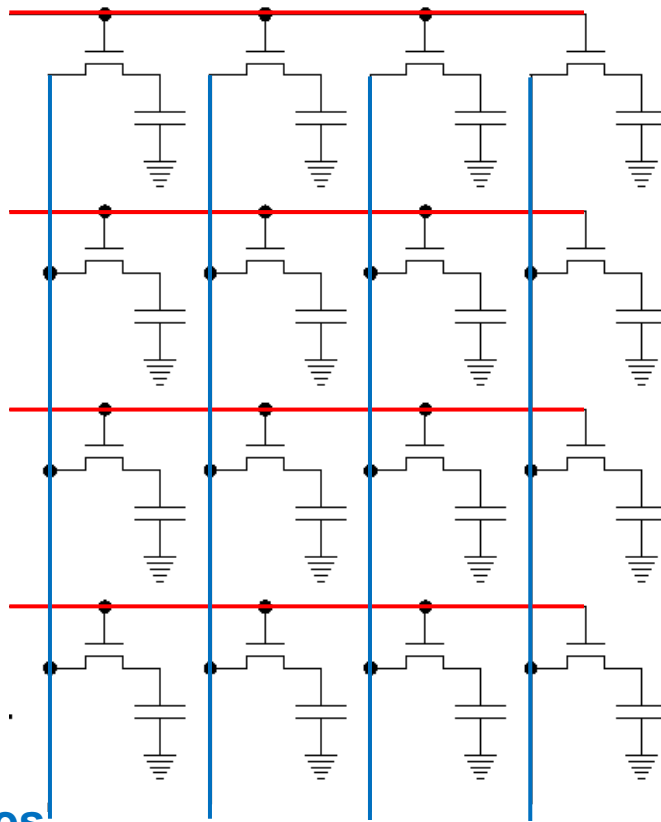# Part 5: DRAM and memory parallelism

Paul H J Kelly

These lecture notes are partly based on the course text, Hennessy and Patterson's *Computer Architecture, a quantitative approach (3rd, 4th, 5th and 6th eds),* and on the lecture slides of David Patterson and John Kubiatowicz's Berkeley course
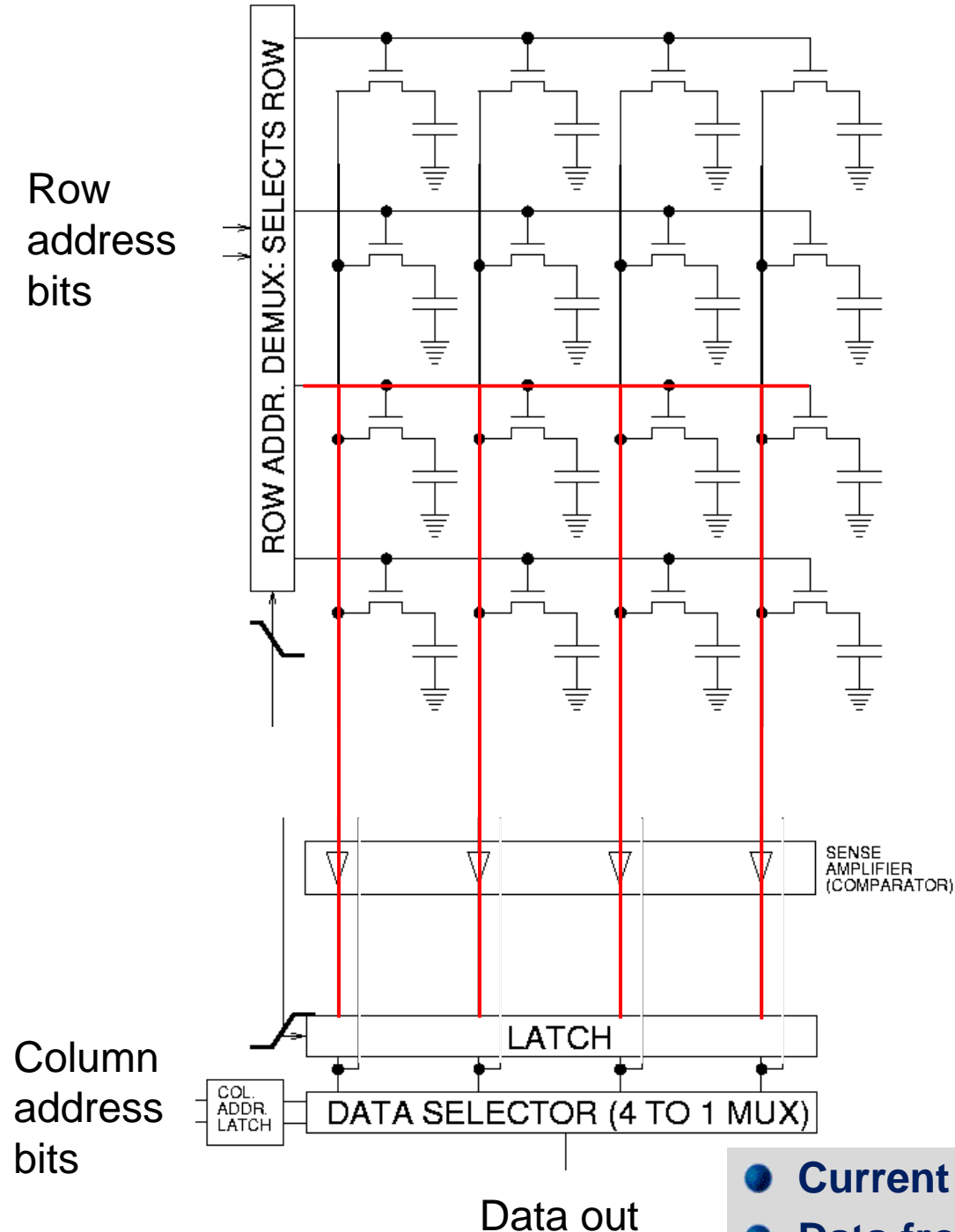
wordlines

bitlines

DRAM array design

- **Square array of cells**
- **Address split into Row address and Column Address bits**
- **Row address selects row of cells to be activated**
- **Cells discharge**
- **Cell state latched by per-column sense amplifiers**
- **Column address selects data for output**
- **Data must be written back to selected row**

- **Wordlines activate transistors along a row**
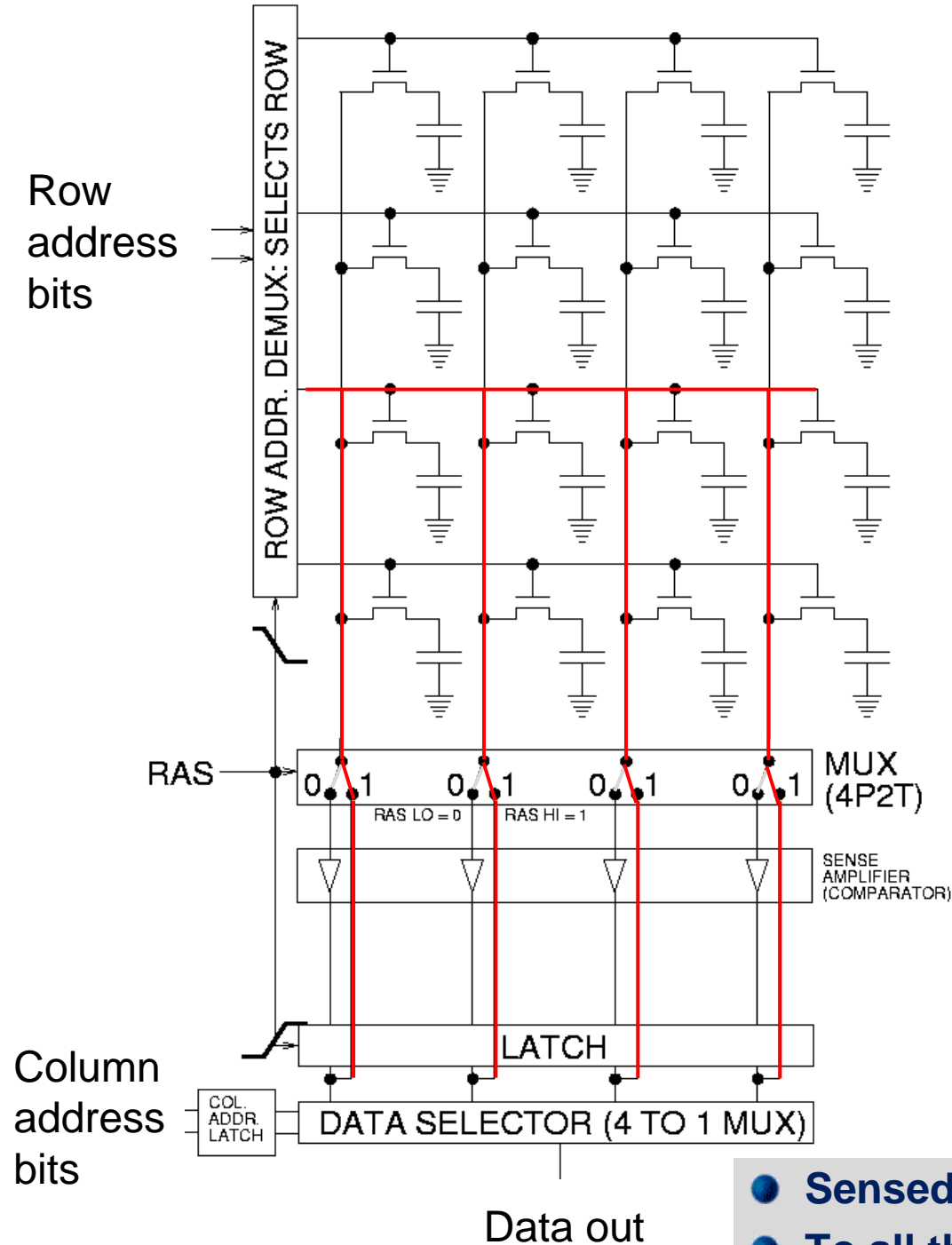- **Charge flows from each capacitor in the row along the bitlines**

DRAM array design

Row address bits

Column address bits

ROW ADDR. DEMUX: SELECTS ROW

SENSE AMPLIFIER (COMPARATOR)

LATCH

COL. ADDR. LATCH

DATA SELECTOR (4 TO 1 MUX)

Data out

- **Square array of cells**
- **Address split into Row address and Column Address bits**
- **Row address selects row of cells to be activated**
- **Cells discharge**
- **Cell state latched by per-column sense amplifiers**
- **Column address selects data for output**
- **Data must be written back to selected row**
- **Current flow is detected and latched**
- **Data from selected column is routed to output**

DRAM array design

Row address bits

ROW ADDR. DEMUX: SELECTS ROW

RAS

RAS LO = 0 RAS HI = 1

MUX (4P2T)

SENSE AMPLIFIER (COMPARATOR)

LATCH

Column address bits

COL. ADDR. LATCH

DATA SELECTOR (4 TO 1 MUX)

Data out

- **Square array of cells**
- **Address split into Row address and Column Address bits**
- **Row address selects row of cells to be activated**
- **Cells discharge**
- **Cell state latched by per-column sense amplifiers**
- **Column address selects data for output**
- **Data must be written back to selected row**
- **Sensed and latched data is written back**
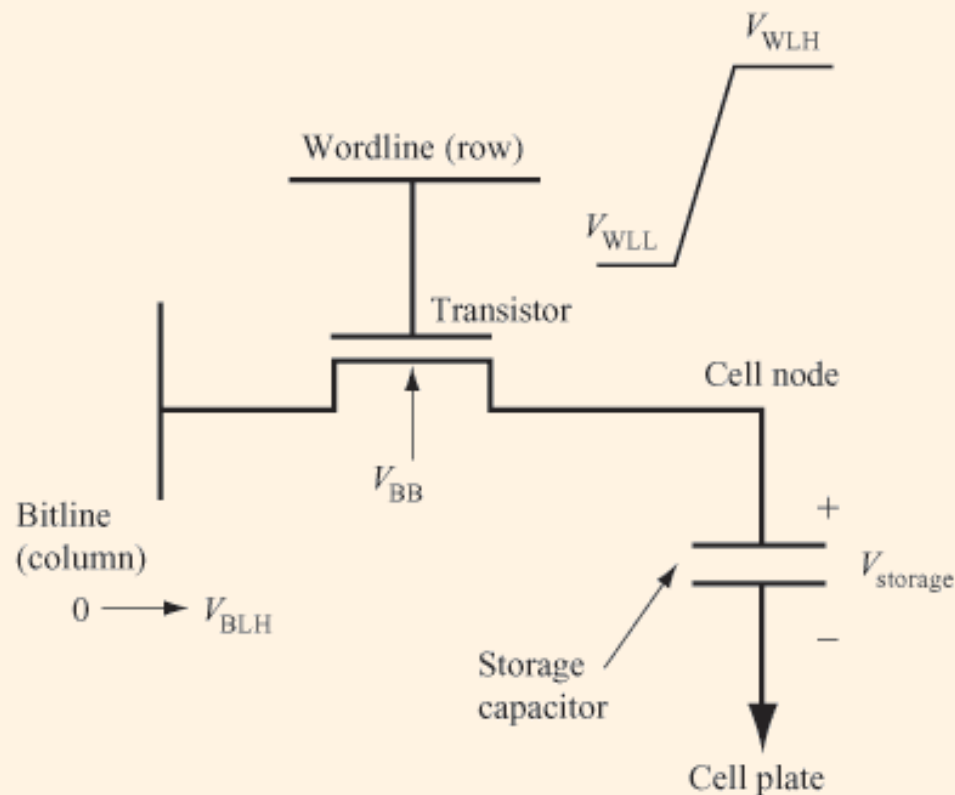- **To all the capacitors in the row**

Figure 1

Schematic of a one-transistor DRAM cell [1]. The array device (transistor) is addressed by switching the wordline voltage from V_{WLL} (wordline-low) to V_{WLH} (wordline-high), enabling the bitline and the capacitor to exchange charge. In this example, a data state of either a "0" (0 V) or a "1" (V_{BLH}) is written from the bitline to the storage capacitor. V_{BB} is the electrical bias applied to the p-well.
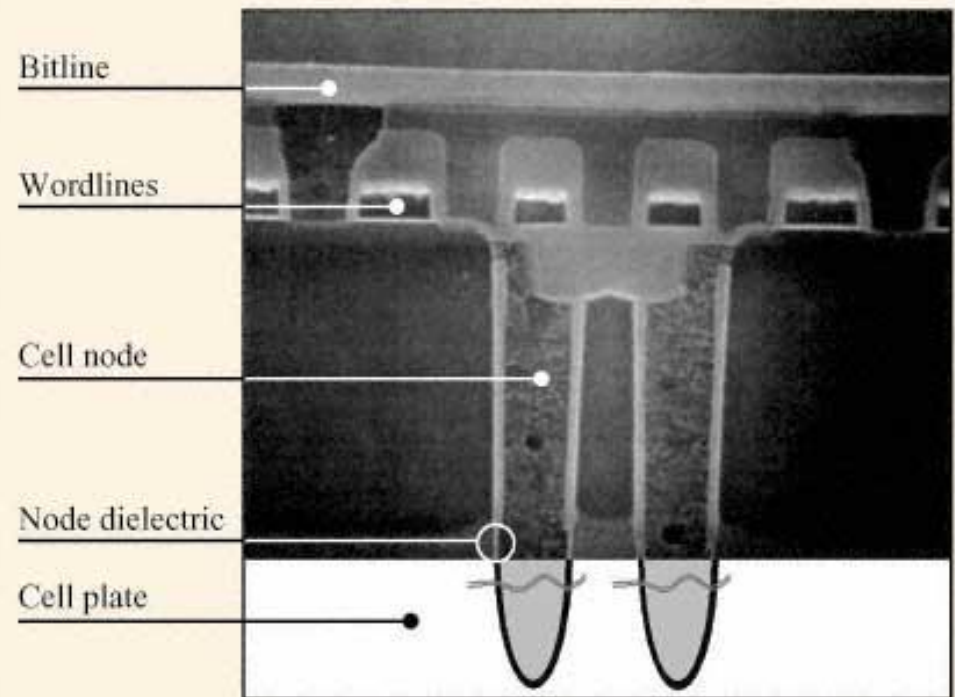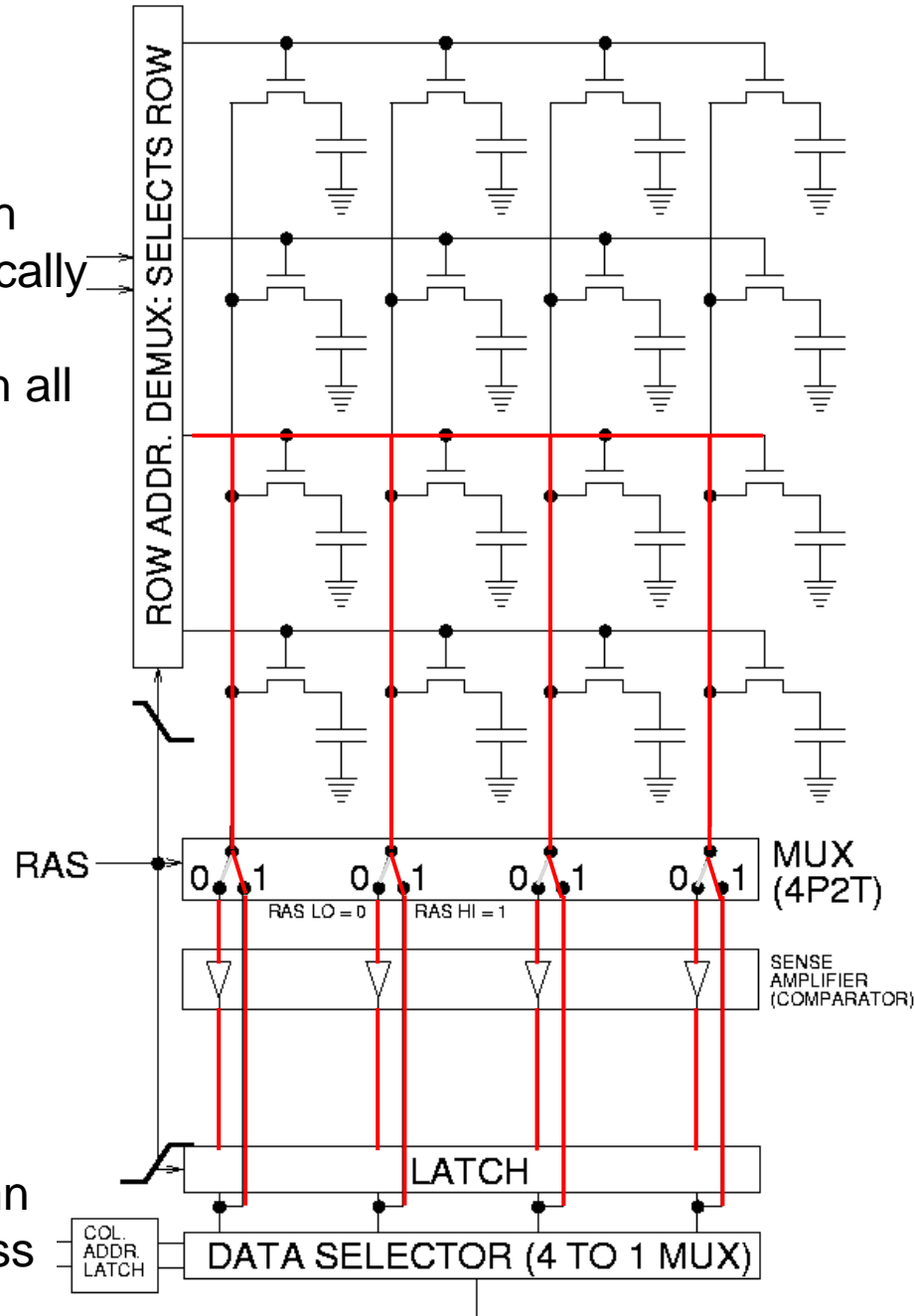
Figure 4

SEM photomicrograph of 0.25-μm trench DRAM cell suitable for scaling to 0.15μm and below. Reprinted with permission from [17]; © 1995 IEEE.

- Single transistor
- Capacitor stores charge
- Decays with time
- Destructive read-out

DRAM refresh

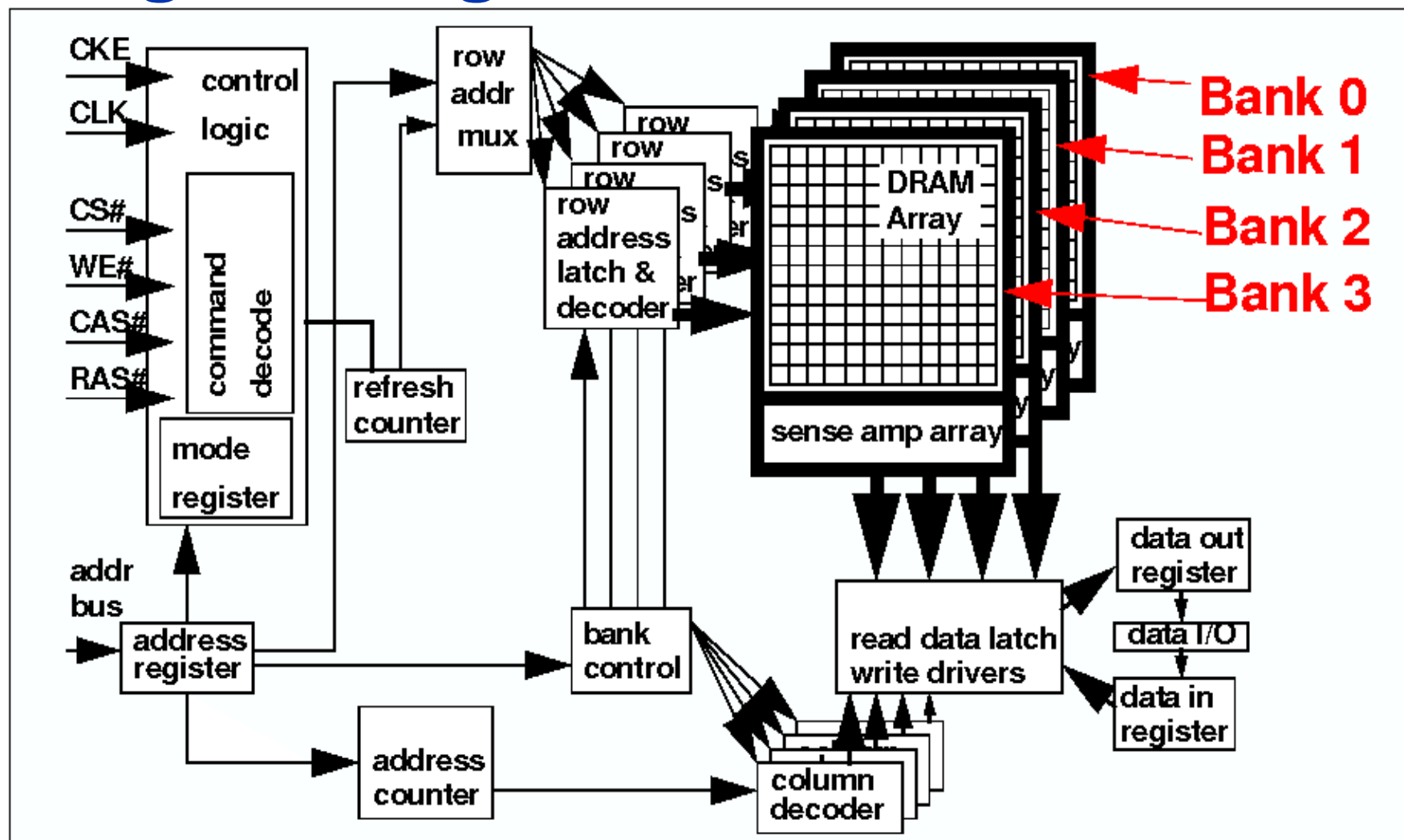Refresh periodically cycles through all rows

Column address bits

- **After a while the charge leaks – and the data cannot be reliably read**
- **So we must refresh every cell periodically**
- **Eg every 64ms**
- **Usually managed by a microcontroller on the DRAM chip**

- **DRAM is unavailable during refresh – so every few microseconds a transaction may be delayed**
- **Refresh may be triggered more frequently if device is hot**
- **Refresh is a significant energy cost – and we could think about how to reduce it**

DRAM timing characteristics

- **Once a row has been selected, the whole row is latched**
- **So different elements of the row (ie from different columns) can be accessed with lower latency**

- **For example: 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
    - Excluding memory controller overhead!

- **Row access *cycle* time is longer than row access time**
  - **Because data needs to be written back after it is read**

Putting it all together...

- **Architecture of SDRAM chip (based on Micron MT48LC32M4A2 data sheet)**
- From David Taiwei Tang's PhD thesis (https://www.ece.umd.edu/~blj/papers/thesis-PhD-wang--DRAM.pdf)
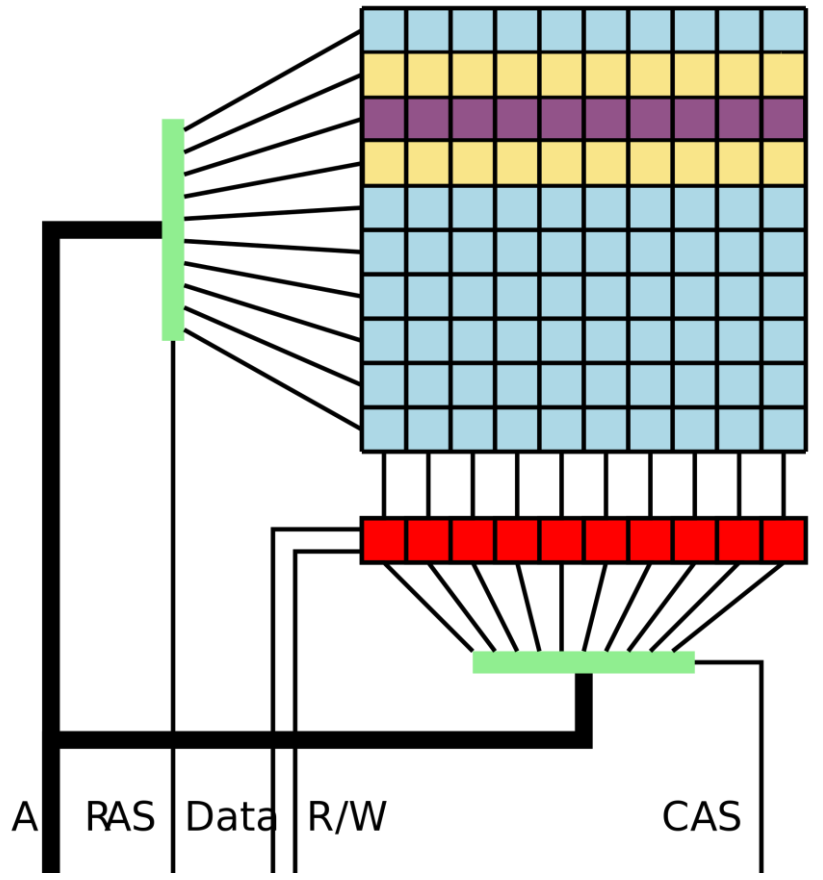
Error correction

Motivation:

- **Failures/time *proportional* to number of bits!**
- **As DRAM cells shrink, more vulnerable**
- **Various causes**
  - **Interference from neighbouring cells**
  - **radiation – such as high-energy cosmic rays ("single-event upsets", SEUs)**

Basic idea: add redundancy through parity/ECC bits

- **Common configuration: Random error correction**
  - **SEC-DED (single error correct, double error detect)**
    - **A Hamming code – see https://en.wikipedia.org/wiki/Hamming_code**
  - **One example: 64 data bits + 8 parity bits (11% overhead)**
  - **Substantial space overhead, and hardware to check and correct**
- **Really want to handle failures of whole chips, not just upset bits**
  - **Organization is multiple DRAMs/SIMM, multiple SIMMs**
  - **Want to recover from failed DRAM and failed SIMM!**
  - **Cf RAID (https://en.wikipedia.org/wiki/RAID)**

Can we *cause* memory upsets?

Rowhammer

- Suppose we write a program that repeatedly writes the yellow rows
- Can we flip bits in the purple row?

- This is how you *test* a DRAM!

- But common DRAMs can still be flipped despite passing manufacturing tests
- With determination and many writes

- This may enable a program to change data that belongs to another process, or the OS
- Such as the page table
- And gain access to private data
  - March 2015: successful attack revealed by security team at Google

A   RAS   Data   R/W   CAS

https://en.wikipedia.org/wiki/Row_hammer

Mitigations

- ECC provides some protection
- Adaptive refresh "target row refresh" (TRR) – count accesses and refresh potential victim rows early
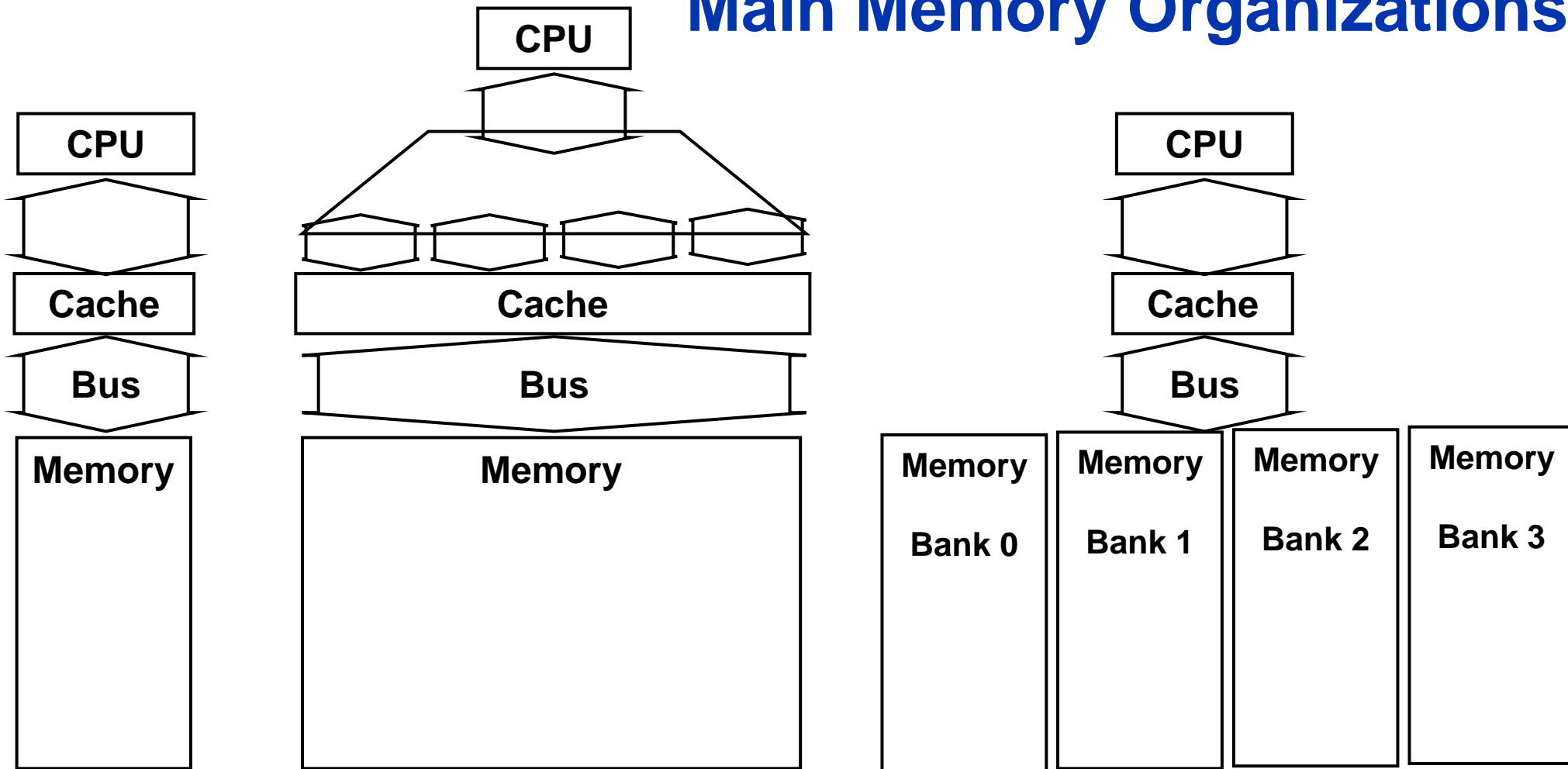- Some evidence that both ECC and TRR can be overcome

Further topics

DRAM topics we don't have time for…

- DRAM energy optimisations

- Stacking, eg Micron's Hybrid Memory Cube (HMC)

- Processing-in-memory, near-memory processing
- Bulk copy and zeroing (easy intra subarray, trickier inter) – Rowclone.

- Non-volatile memory (NVM), storage-class memory (SCM), Phase-change memory (PCM), Flash, Crosspoint, Optane
- Why NVM isn't a filesystem

More cache topics:
- Energy optimisations in the memory hierarchy
  - Sparsh Mittal, **A survey of architectural techniques for improving cache power efficiency**, Sustainable Computing: Informatics and Systems, Volume 4, Issue 1, 2014,
- Content locality, upper bit content locality
- Compressed skewed caches
  - Somayeh Sardashti, André Seznec, and David A. Wood. **Skewed Compressed Caches**. MICRO 2014

Main Memory Organizations

CPU

CPU

CPU

Cache

Cache

Cache

Bus

Bus

Bus

Memory

Memory

Memory	Memory	Memory	Memory
Bank 0	Bank 1	Bank 2	Bank 3

● *Simple*:

● *Wide*:

● *Interleaved*:

- ● **CPU, Cache, Bus, Memory same width (32 or 64 bits)**

- ● **Parallel data transfer**
  - ● **Same address in all banks**

- ● **Parallel *addressing***
  - ● **Different address in each bank**
- ● **Parallel data transfer**
- ● **Bank selection strategy?**

Main Memory Summary

- DRAM arrays have separate row address latency
- DRAM reads are destructive – needs to be written back
- DRAM needs periodic refresh
- Memory parallelism:
  - DRAM packages typically include multiple DRAM arrays, which can be row-addressed in parallel
  - Wider Memory – increase transfer bandwidth to transfer up to a whole row (or more?) in a single transaction
  - Interleaved Memory: allowing multiple addresses to be accessed in parallel
- Bank conflicts occur when two different rows in the same DRAM array are required at the same time
  - Hardware strategies, software strategies…
- Need error correction (which costs in space and time)
- Non-volatile memory technologies – fast-moving field
  - Flash – requires block erasure, suffers burnout
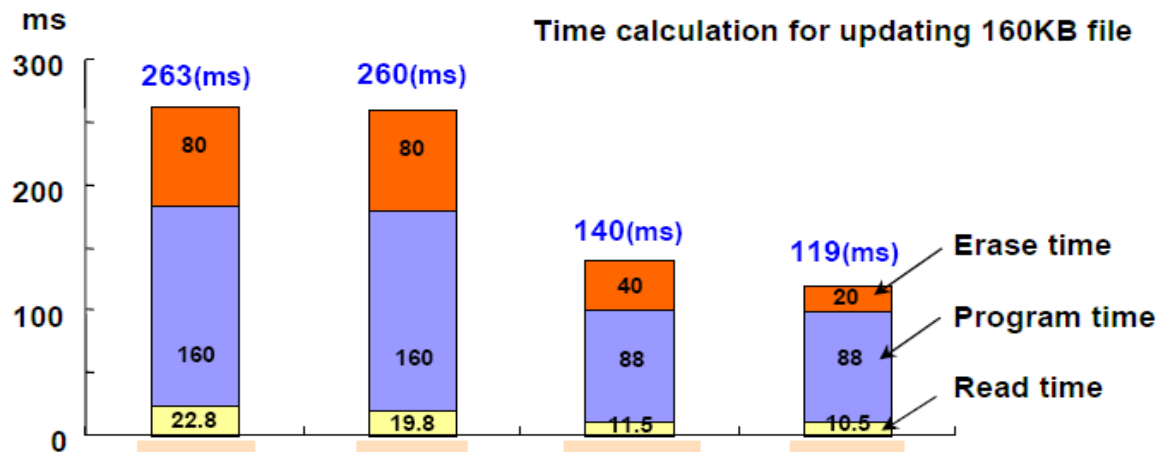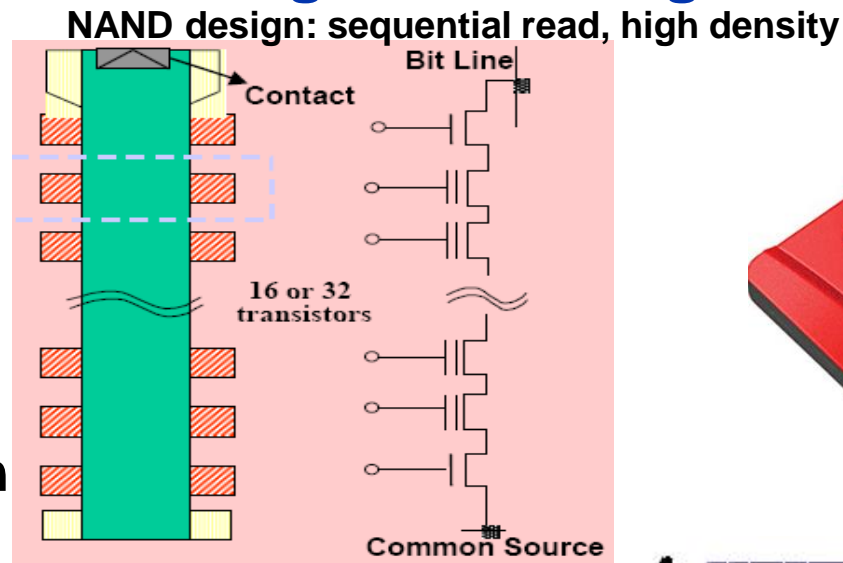 - Phase-change – slow writes, bitwise addressable
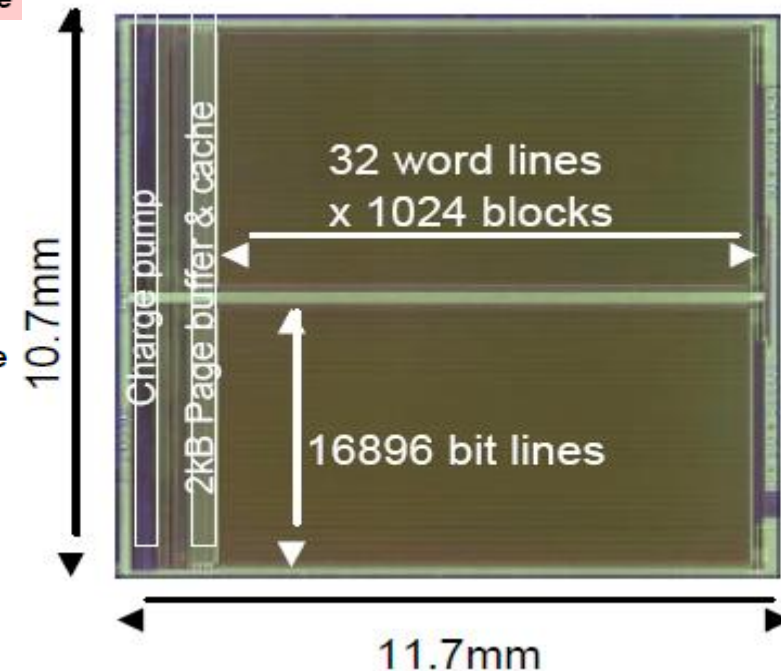
Extra material for interest

FLASH
Storage Technologies: dense, non-volatile

- **Mosfet cell with two gates**
- **One "floating"**
- **To program, charge tunnels via <7nm dielectric**
- **Cells can only be erased (reset to 0) in blocks**

NAND design: sequential read, high density

Contact
Bit Line
16 or 32 transistors
Common Source

Time calculation for updating 160KB file

263(ms) — 80, 160, 22.8
260(ms) — 80, 160, 19.8
140(ms) — 40, 88, 11.5
119(ms) — 20, 88, 10.5

Erase time
Program time
Read time

	8Mb	16Mb	32/64Mb	128Mb
Page size	256Byte	256Byte	512Byte	512Byte
Block size	4KByte	4KByte	8KByte	16KByte
Read/page	10us	10us	10/7us	10us
Program/page	250us	250us	250/200us	200us
Erase/block	2ms	2ms	2ms	2ms

Charge pump
2kB Page buffer & cache
32 word lines x 1024 blocks
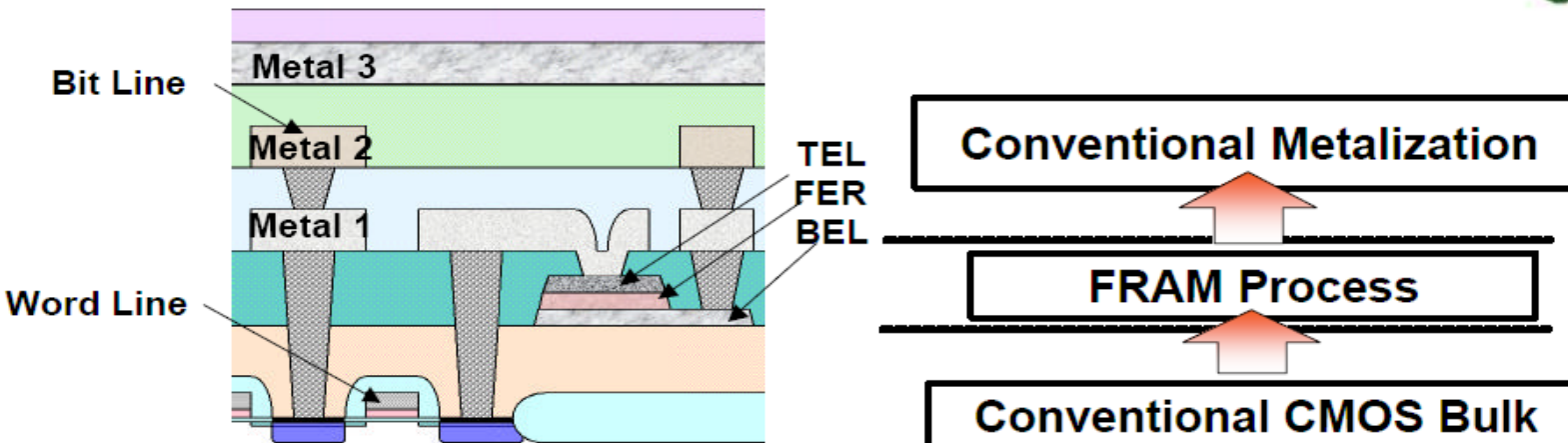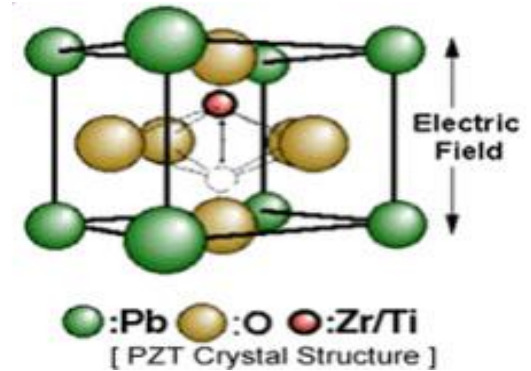16896 bit lines
10.7mm
11.7mm

I Gbit NAND Flash memory

Diverse non-volatile memory technologies

- **FRAM**

  - Perovskite ferroelectric crystal forms dielectric in capacitor, stores bit via phase change
  - 100ns read, 100ns write
  - Very low write energy (ca.1nJ)

Electric Field

○ :Pb ○ :O ● :Zr/Ti

[PZT Crystal Structure]

Bit Line

Metal 3

Metal 2

Metal 1

Word Line

TEL
FER
BEL

Conventional Metalization

FRAM Process
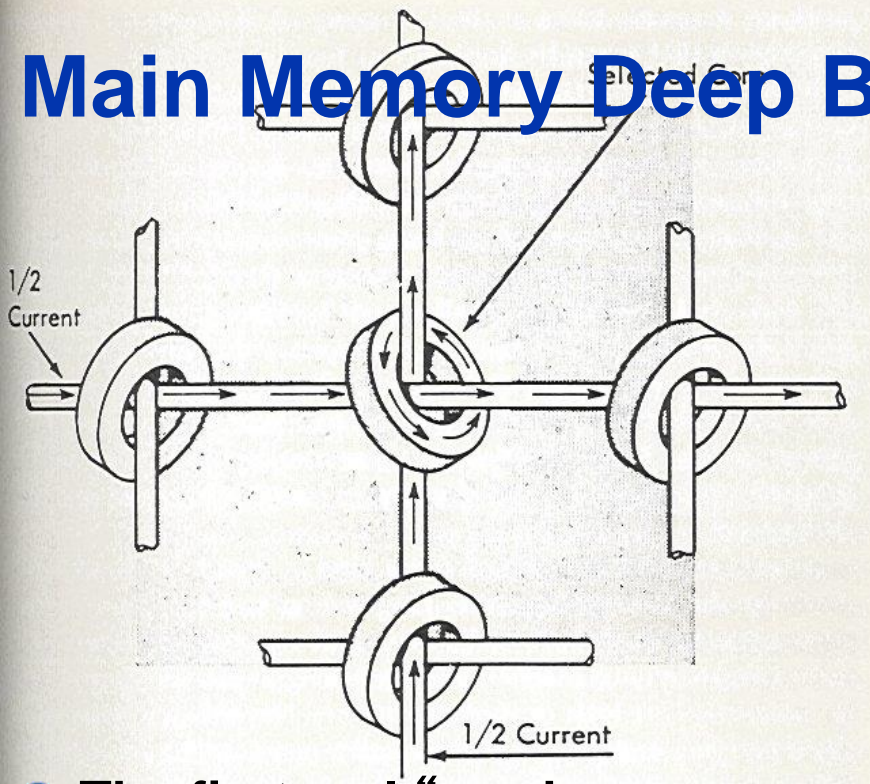
Conventional CMOS Bulk

Additional FRAM process
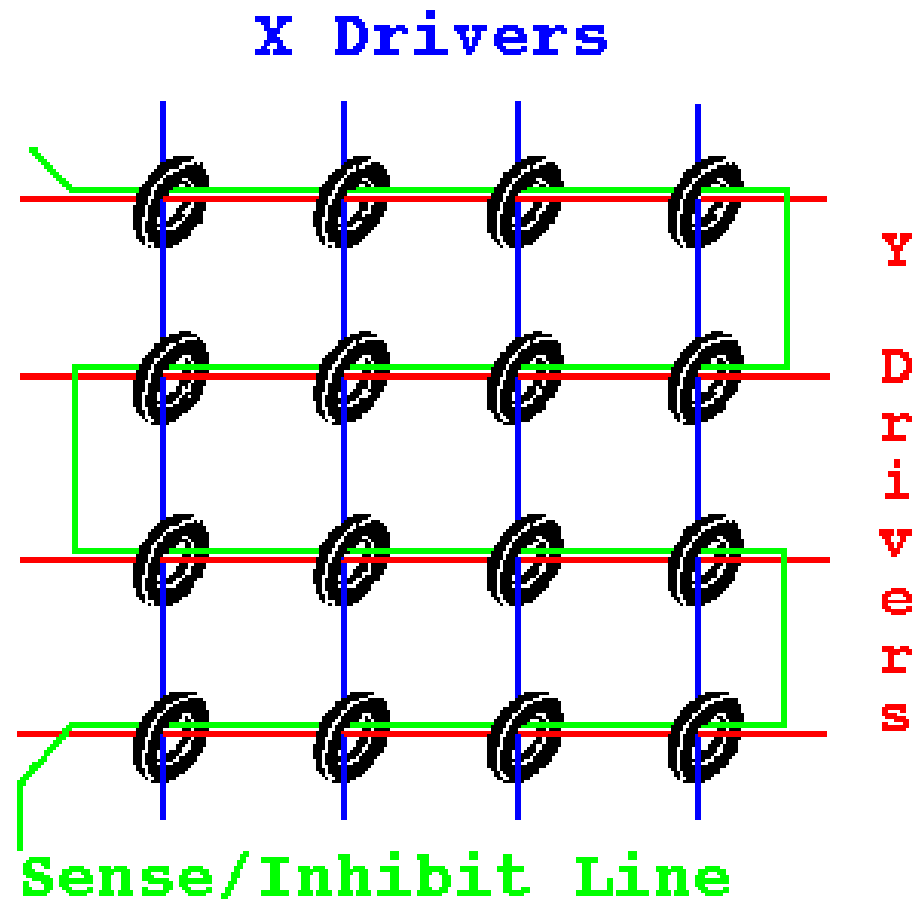between conventional CMOS bulk and metalization

Compatible with conventional CMOS technology

and existing CMOS cell libraries

- **Fully integrated with logic fab process**
- **Currently used in Smartcards/RFID**
- **Soon to overtake Flash?**
- **See also phase change RAM**

http://www.fma.fujitsu.com/fram/framDocs01.asp?grOut=Documentation&sec=Documentation
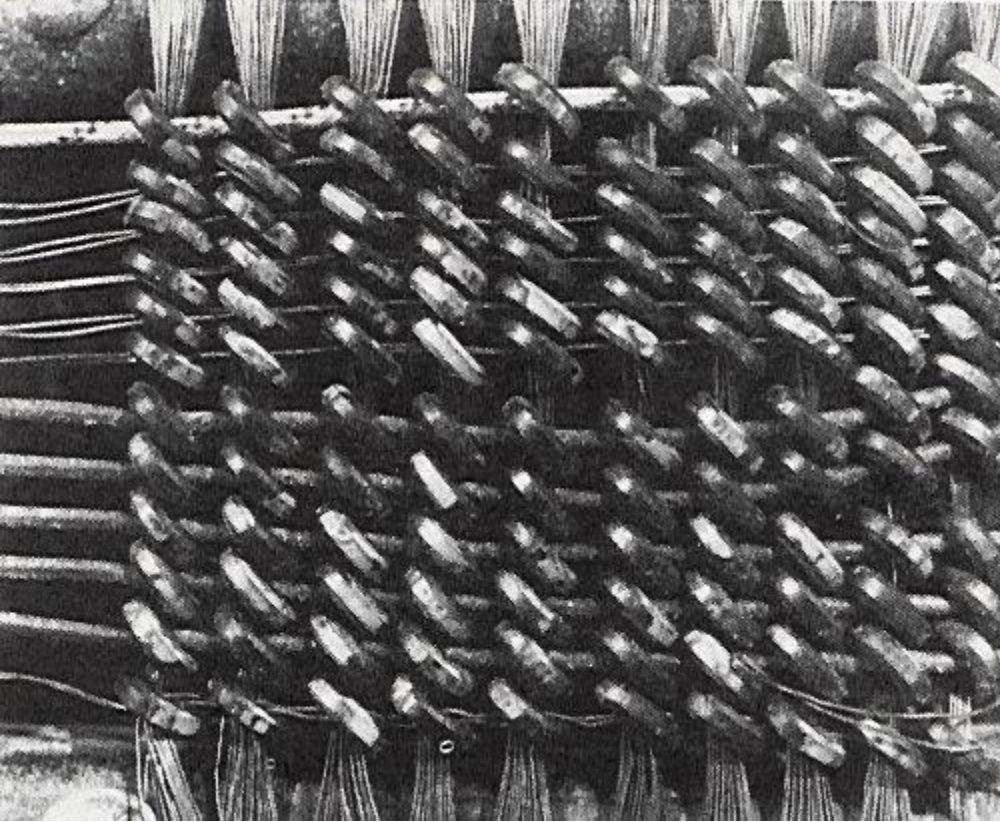
Main Memory Deep Background

The first real "random-access memory" technology was based on magnetic "cores" – tiny ferrite rings threaded with copper wires

That's why people talk about "Out-of-Core", "In-Core," "Core Dump"

Non-volatile, magnetic

Lost out when 4 Kbit DRAM became available

Access time 750 ns, cycle time 1500-3000 ns

X Drivers

Y Drivers

Sense/Inhibit Line

Pulse on sense line if any core flips its magnetisation state

http://www.faqs.org/docs/electric/Digital/DIGI_15.html
http://www.psych.usyd.edu.au/pdp-11/core.html

Figure 10. IBM 2361 Core Storage

- The first magnetic core memory, from the **IBM 405 Alphabetical Accounting Machine**. The photo shows the single drive lines through the cores in the long direction and fifty turns in the short direction. The cores are 150 mil inside diameter, 240 mil outside, 45 mil high. This experimental system was tested successfully in April 1952.
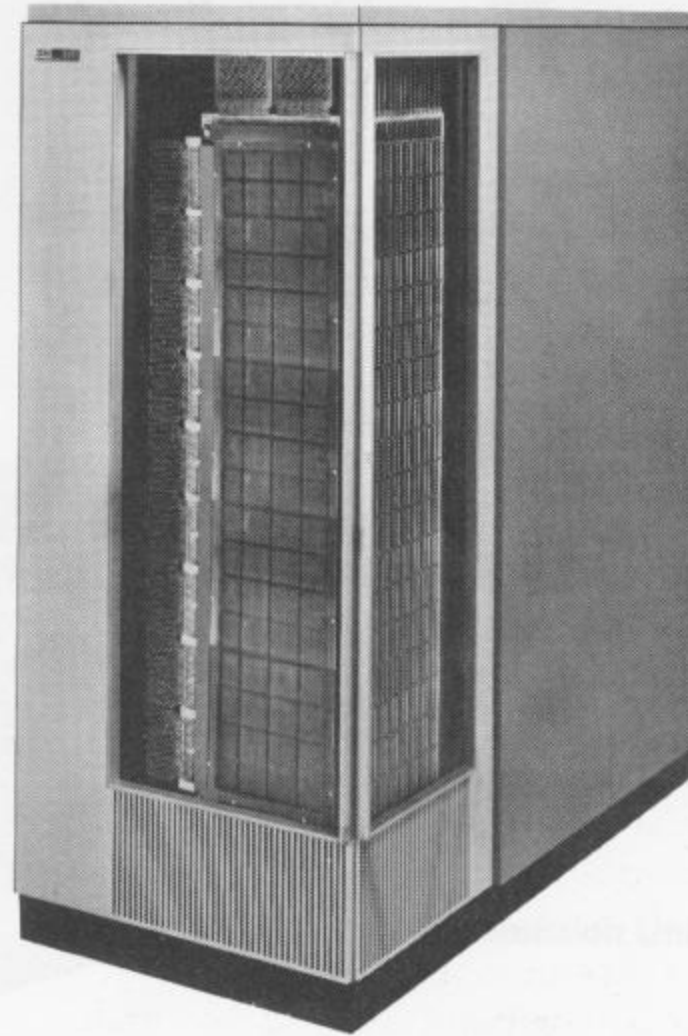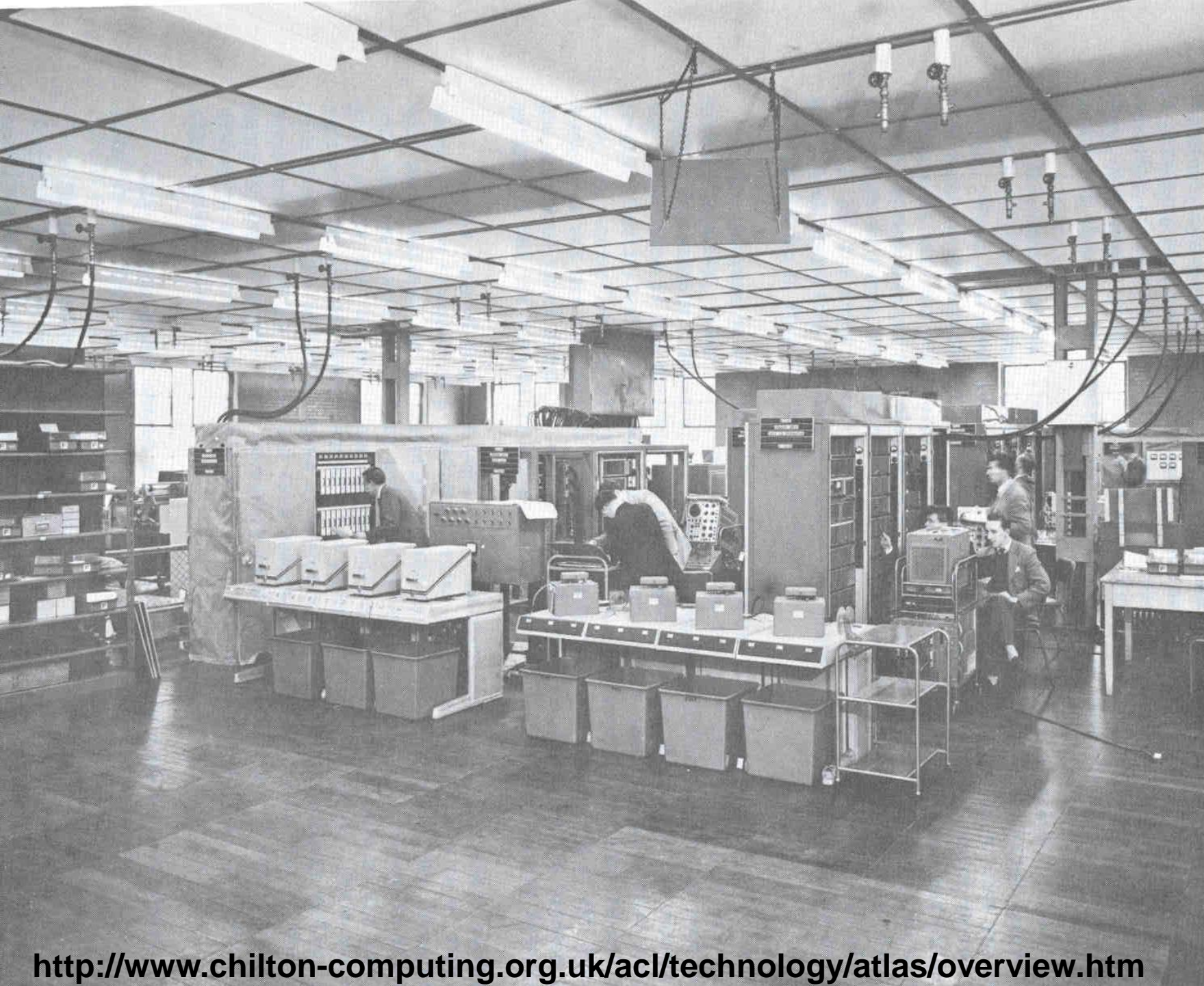
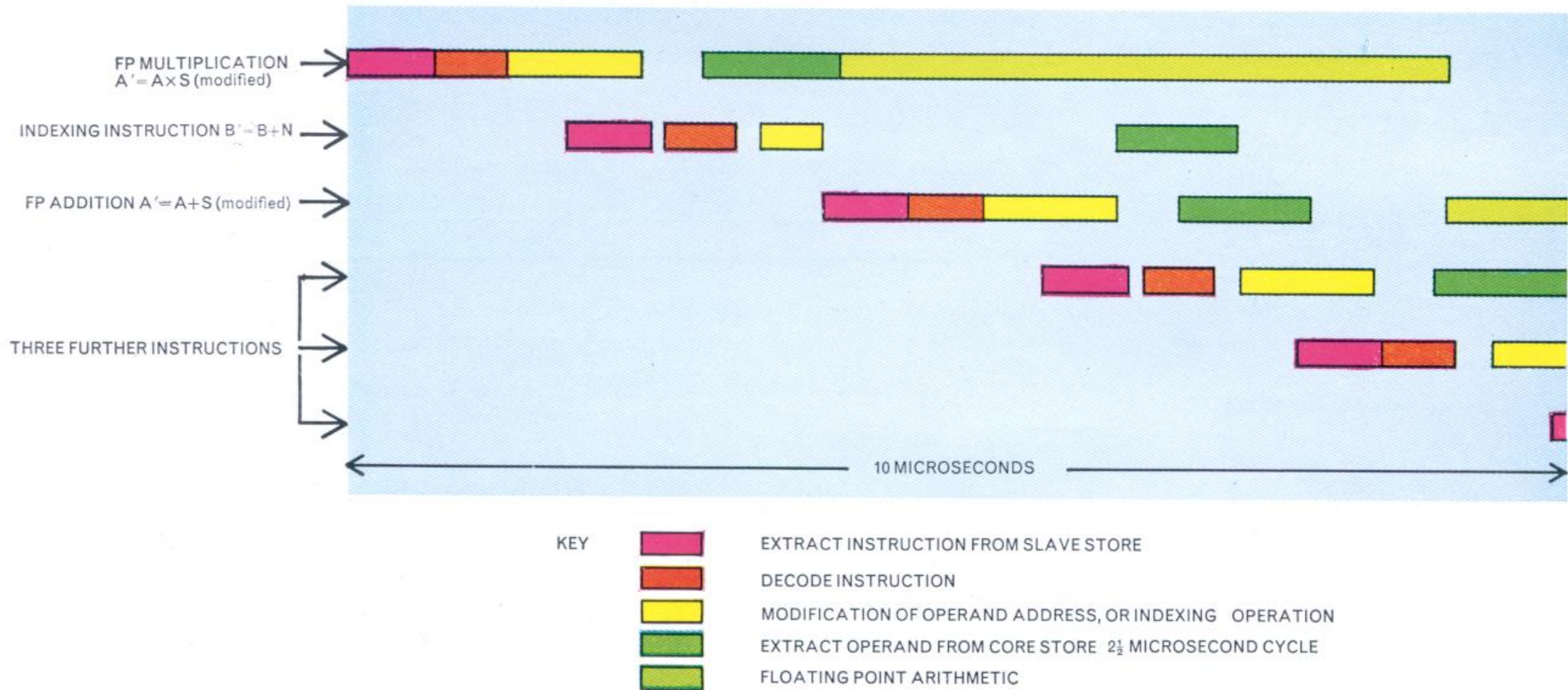- 524,000 36-bit words and a total cycle time of eight microseconds in each memory (1964 – for the IBM7094)

Atlas

- **First was at University of Manchester**
- **University of London had the second one**
- **Commissioned May 1964**
- **Shut down Sept 1972**

http://www.chilton-computing.org.uk/acl/technology/atlas/overview.htm

Pipelined instruction processing in Atlas

FP MULTIPLICATION
A´= A × S (modified)

INDEXING INSTRUCTION B´= B + N

FP ADDITION A´= A + S (modified)

THREE FURTHER INSTRUCTIONS

10 MICROSECONDS

KEY
- EXTRACT INSTRUCTION FROM SLAVE STORE
- DECODE INSTRUCTION
- MODIFICATION OF OPERAND ADDRESS, OR INDEXING OPERATION
- EXTRACT OPERAND FROM CORE STORE 2½ MICROSECOND CYCLE
- FLOATING POINT ARITHMETIC

http://www.chilton-computing.org.uk/acl/technology/atlas/overview.htm

- **Atlas is most famous for pioneering virtual memory**
- **Also**
  - **Pipelined execution**
  - **Cache memory ("slave store") – 32 words**
  - **Floating point arithmetic hardware**