Commercial Memories & Cache

Peter Rounce
p.rounce@cs.ucl.ac.uk

What we will cover:

Static memory (SRAM)

- fast but expensive
- used in Main memory of very small systems
- used in cache

Dynamic memory (DRAM)

- cheap but slow
- used in Main memory of most systems

Cache Memory -
gives DRAM system the speed of SRAM (almost)

EEPROMS/Flash Ram –
non-volatile memory used in many modern devices

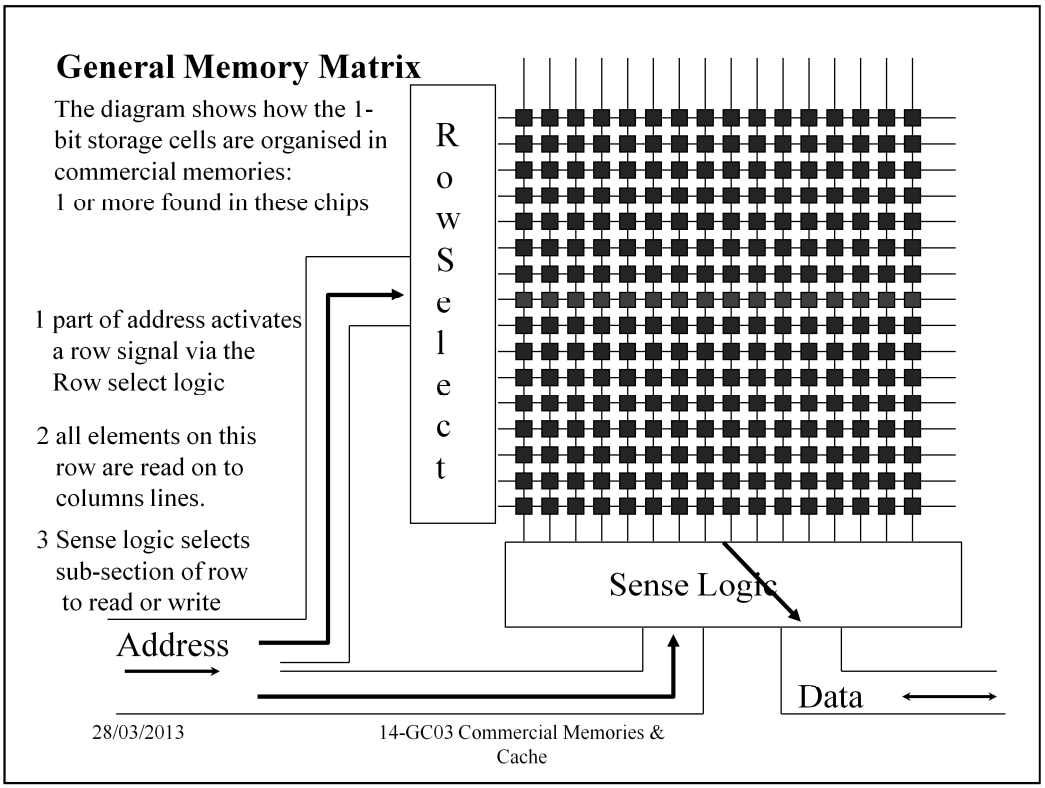
1st look at technology - some jargon to understand

28/03/2013 14-GC03 Commercial Memories & Cache

RAMs are writeable memories (they should be called Read-Write Memory(RWM)), which have the same read and write access times (unlike writeable ROMS, which are fast to read but slower to write). RAM stands for **R**andom **A**ccess **M**emory, which is what **r**ead-**w**rite **m**emory is usually called, despite the fact that ROM can also be randomly accessed - any location can be accessed in any order. RAMs come in 2 forms: *static* and *dynamic*.

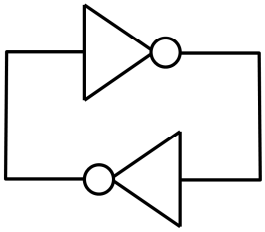
Both *static* and *dynamic* memories lose their data when power is removed.

Some systems use CMOS static RAM and keep it supplied with power even when the rest of the system is shut down, which maintains the contents of the RAM. This uses little power, since when not being switched CMOS logic uses only a very small amount of power, which can be provided by a very small battery. This is also done in small electronic diaries. Similar systems are used in most machines to keep the clock running and up-to-date while systems are powered off.

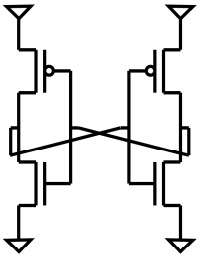

General Memory Matrix
The diagram shows how the 1-bit storage cells are organised in commercial memories:
1 or more found in these chips
1 part of address activates a row signal via the Row select logic
2 all elements on this row are read on to columns lines.
3 Sense logic selects sub-section of row to read or write
Row Select
Sense Logic
Address
Data
28/03/2013
14-GC03 Commercial Memories & Cache

SRAM

Remember basic binary storage element

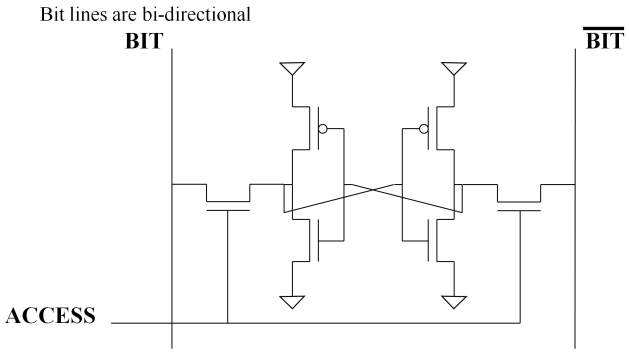


Implement this is CMOS



This is the core element in SRAM

Static Ram 1-bit memory cell - 6 transistors

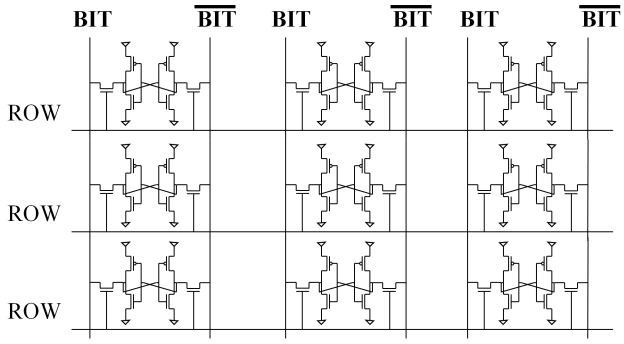


WRITING: Data value to be written into cell is placed on BIT line and inverse on $\overline{BIT}$ line
ACCESS set to 1 - values on BIT & $\overline{BIT}$ force cell to change to new value.

READING: BIT & $\overline{BIT}$ lines are not driven –
ACCESS = 1 - BIT & $\overline{BIT}$ lines are driven by outputs of 1-bit cell
data output on and read from BIT lines

28/03/2013 14-GC03 Commercial Memories & Cache

Matrix arrangement of Static Memory cells

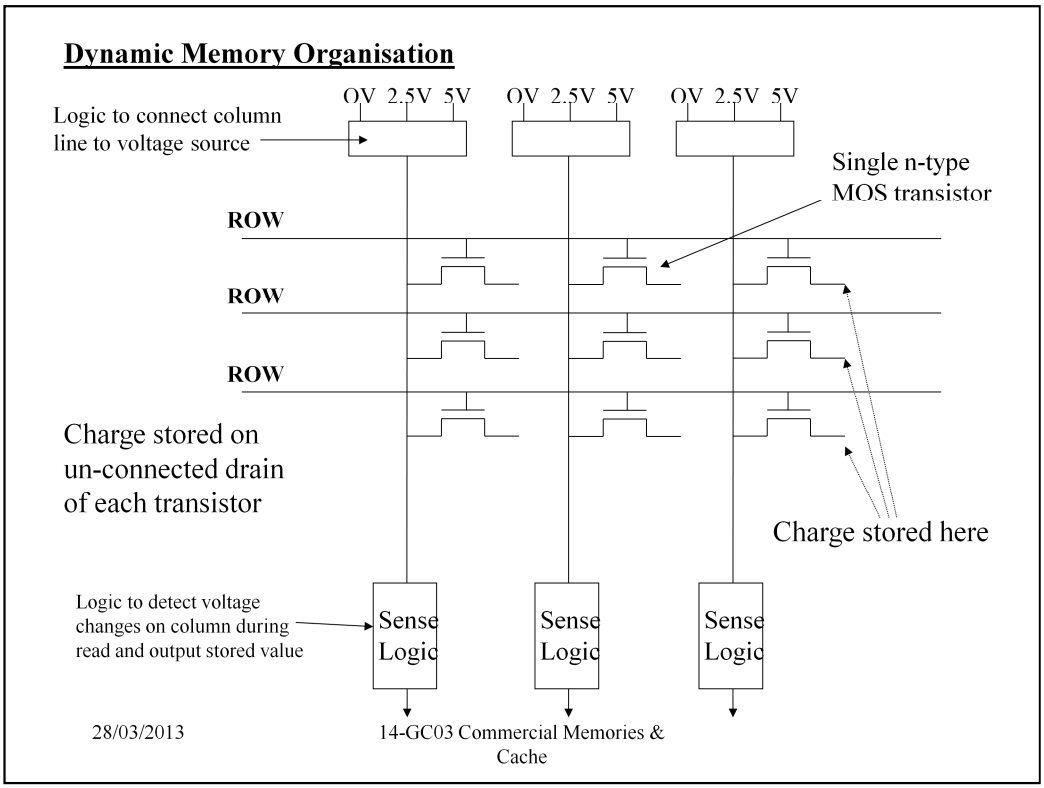


Part of address is used to activate a ROW line.
Whole row of cells is read.
Rest of address selects sub-set of row to output.
For writing, the addressed sub-set of selected row is driven with new data.

28/03/2013 14-GC03 Commercial Memories & Cache

Static memory is the memory form examined above and the basic 1-bit store is usually based upon a pair of inverters with feedback to store a data bit as earlier.

The static memory above has 6 transistors per bit of storage plus some transistors for address decoding and other purposes. For large storage capacities 128Kbits etc, these extra transistors are only small fraction of the total number of transistor, so that the average number of transistors per bit is only fractionally above 6 transistors per storage bit. There are some static memory designs that require only 4 or 5 bits per bit of storage: these designs therefore get more storage on the same area of sillicon and are therefore cheaper bit for bit than 6 transistor designs.



Dynamic memory uses a completely different method to store a bit: it stores a small amount of electrical charge in each 1-bit store. The presence or absence of the charge is used to determine the state of the memory, 0 or 1.

Since charge tends to leak away over a short period, these memories have to be periodically read and re-written to keep their data.

Measuring charge is difficult and takes some time. Thus the design and operation of these devices is more complex.

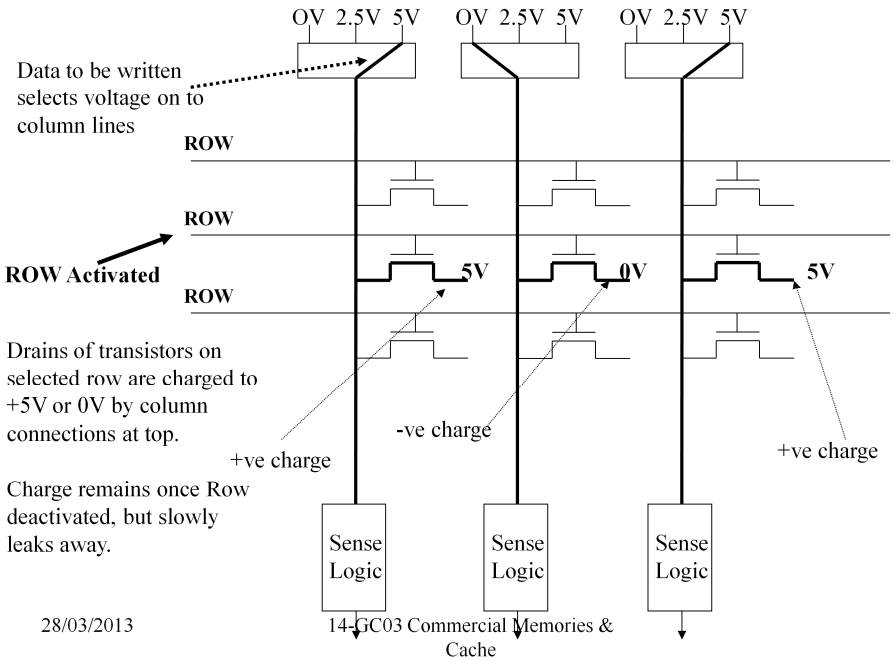
Thus, these devices are more complicated and slower (access time ~ 70 nsec) than static memory (access time 2-10 nsec), although their organisation in a matrix of rows and columns is similar to that of static memory.

Their big asset is that the only one transistor is needed in each memory location to store 1-bit rather than the 6 transistors used in a *static* memory. Thus for the same number of transistors, 6 times more data can be stored on the same chip area in a *dynamic* memory.

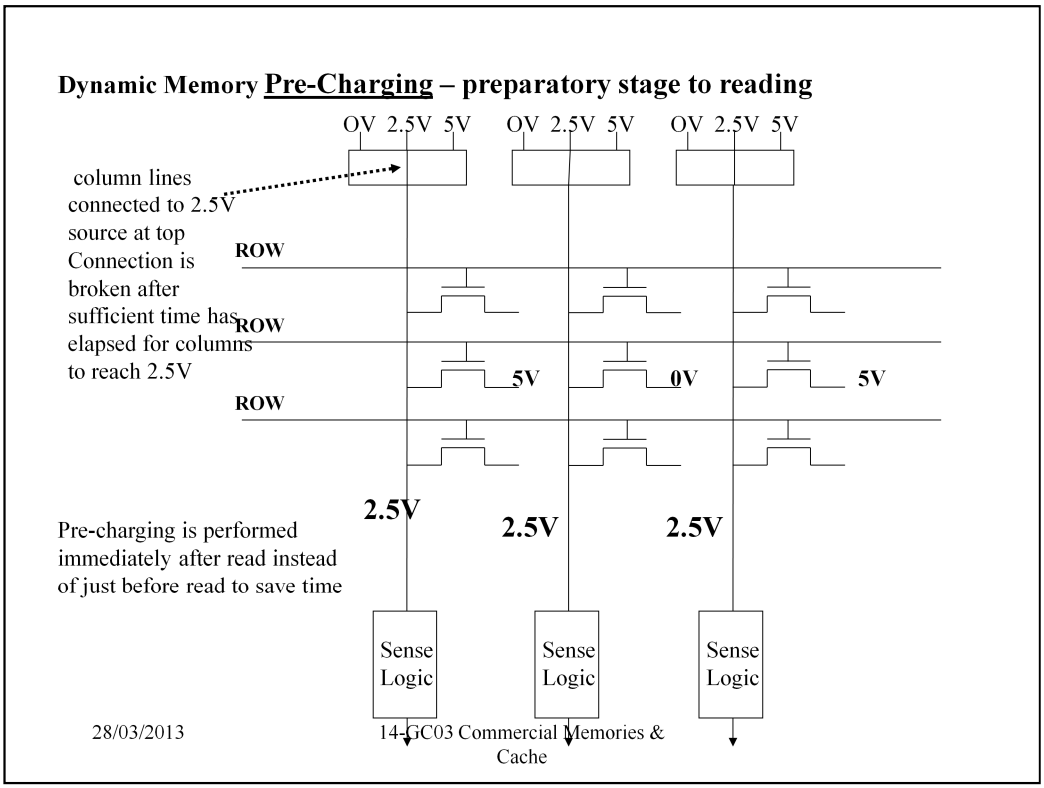
Thus dynamic memory is cheaper to buy, and you get a lot more on each chip. Because of both of these, and particularly the price, dynamic memory has been the memory of choice for main memory, so that it is produced in very large volumes, which makes the price even cheaper still. It is the leading edge silicon technology with transistors sizes smaller than other designs because there is a lot of money to made from it.

Thus *dynamic* memories are preferred for large memory systems, e.g. main memory in personal computers and workstations, but not for small or fast memory systems, e.g. cache memory.

Dynamic Memory Writing

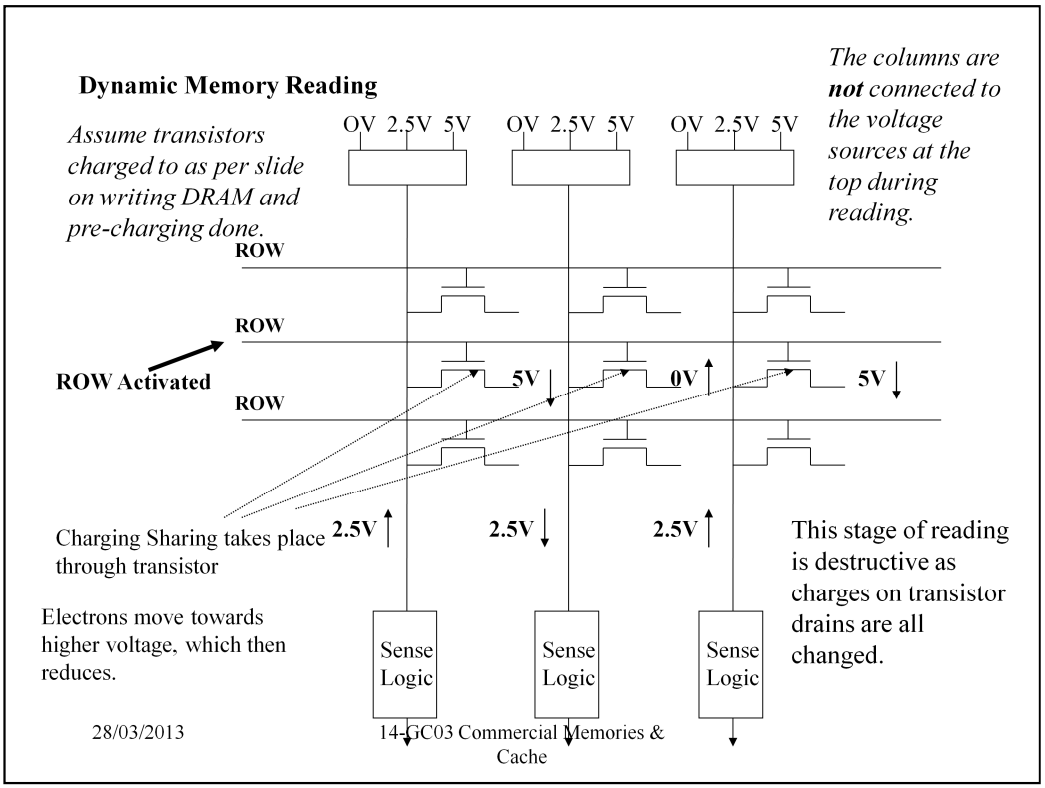


28/03/2013 14-GC03 Commercial Memories & Cache



Pre-charging of the column lines to 2.5V (in a 0V and 5V system) is necessary in order to detect the charge on the transistors to be read.

This process has to be done before every read, adding to the time it takes to access the memory.



Reading is a charge sharing process: the charge on the drain of a transistor to be read is shared with the attached column line attached to the transistor source. The charge sharing occurs through the channel between source and drain, when the channel is created via the gate voltage.

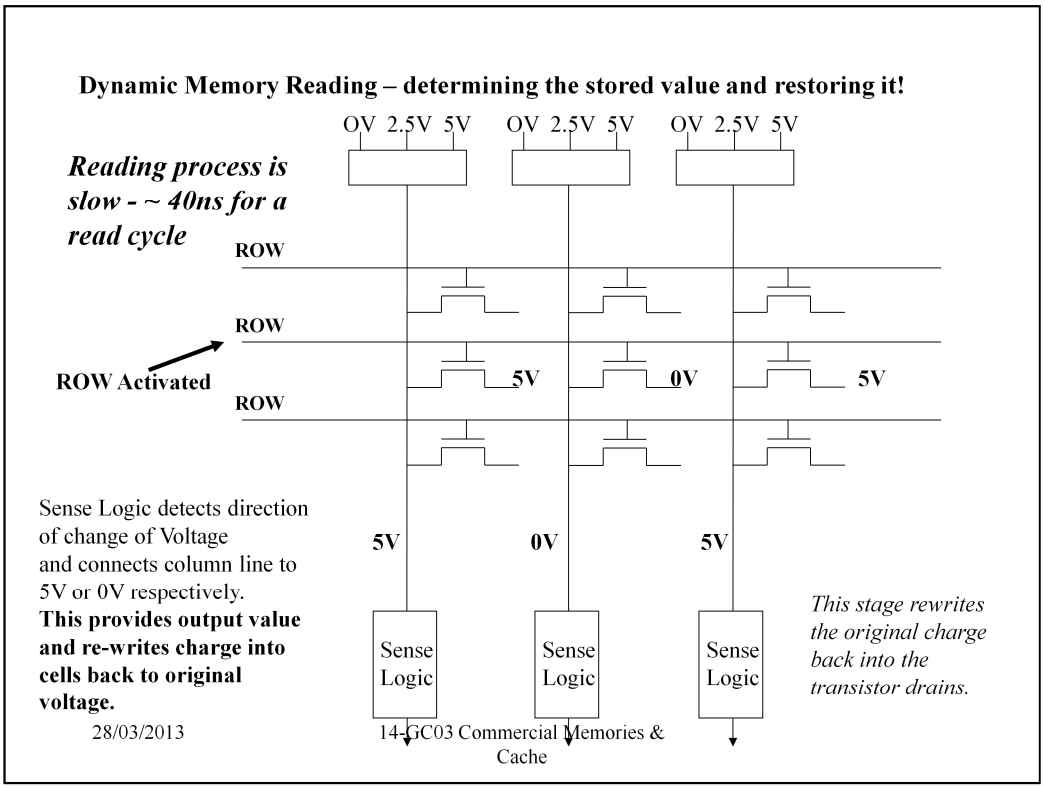
It is not possible to measure how much charge moves across the channel, but it is possible to detect the direction in which the voltage on the column moves:

if the voltage on the drain is greater than 2.5V then electrons will move from the column line onto the drain with the column voltage increasing (drain voltage decreases);

if the voltage on the drain is less than 2.5V then electrons will move from the drain onto the column line with the column voltage decreasing (drain voltage increases).

A clever circuit in the sense logic (I was really impressed when I saw it, but I can't remember it exactly), detects the direction of voltage change and makes a connection between the column line and the power supply terminal toward which the column line voltage is changing: if the later is decreasing the connection is to 0V; if the latter is increasing it is to 5V. The voltage to which the column line is connection is the same as that on the drain of the transistor being read.

The drain voltage can change from that originally written and the value will still be read correctly as long as it doesn't change by not much more than 1V.



Reading initially destroys the stored charged on the transistor drain, but then writes back the original charge: it also makes up for any lost charge.

DRAM Refresh

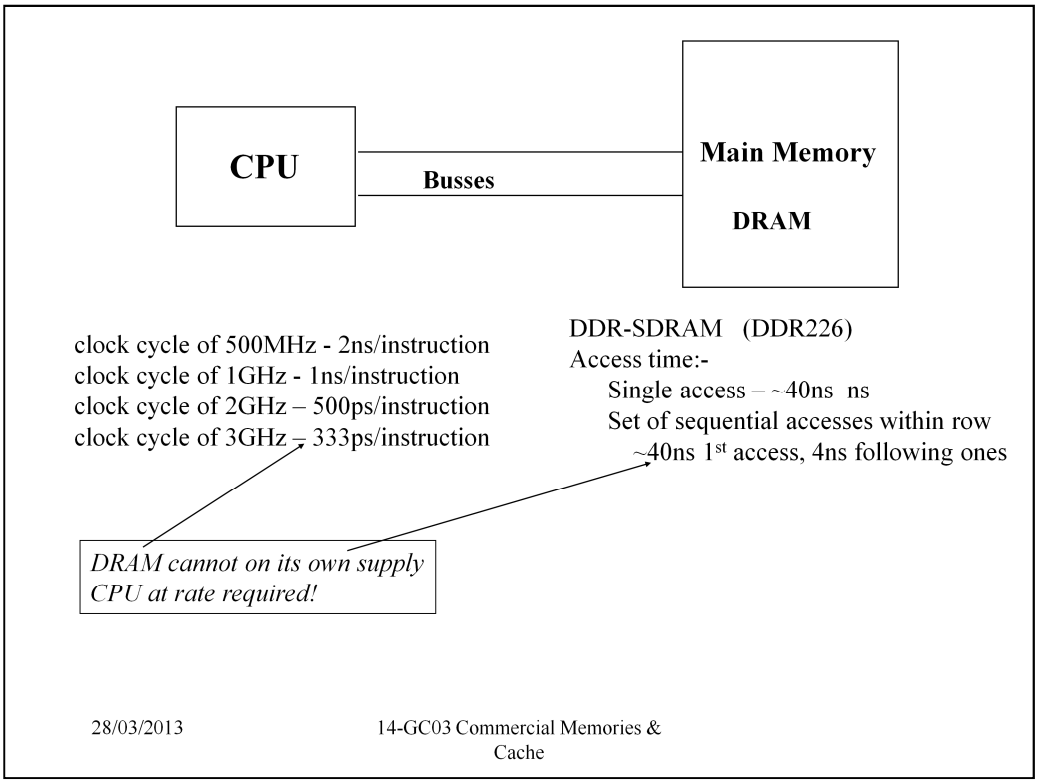
Charge on transistor drain
- leaks away slowly to uncharged state
- all stores look to have same value!!
- data is lost after sufficient time

Data is re-written (refreshed) when read –
- in fact, all data in row is refreshed if any bit in row is read or written

DRAM have every row refreshed every 8 milliseconds
done automatically by logic associated with DRAM
DRAMS are designed such that only 256 or 512 refresh cycles need to refresh DRAM chip regardless of size – these cycles are spread over 8 ms period.

Read and write accesses are suspended during refresh cycle

28/03/2013 14-GC03 Commercial Memories & Cache



Modern CPUs attempt to execute one instruction per CPU clock cycle.

Main memory is always DRAM
low transistor count / bit of memory → highest density -> largest sizes)
high volumes sales → lowest cost/bit → much cheaper per bit than static memory\

Comes in a range of versions:-
EDO DRAM, SDRAM, RDRAM

Single access time contains 38 ns for access + 27ns recovery time before next access

What's wrong with using all fast static memory?

Example: using 1995 prices for example

16Mbyte 60ns DRAM main memory - cost ~£500

16Mbyte of 20 ns Static RAM - £3584 (~£28 for 128kbytes)
16Mbyte of 15 ns Static RAM - £5632 (~£44 for 128kbytes)

DRAM prices have dropped considerably from 1995:
Nov 2011: £16 for 1Gbyte of PC3200 DDR SDRAM?

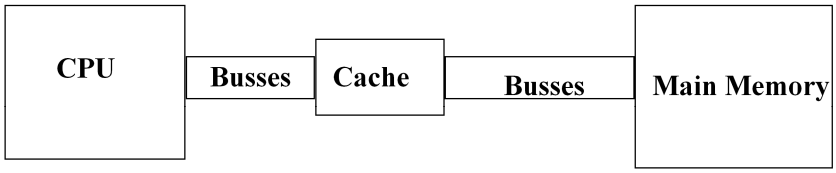
SRAM Nov 2011: 512Kbytes (256kx16) of 8 ns Static RAM - £5
1 GByte = 2000*£5 → £10000!!

Solution:
Get similar performance to static main memory, but use DRAM for main memory with a small cache memory made from static RAM!
2010: I**ntel® Pentium® Processor P6100 has a 3MByte on-chip Cache**

What is cache……..?

28/03/2013 14-GC03 Commercial Memories & Cache

System Organisation with Cache



First read at a memory address: **read from main memory, copy to Cache**

Second read at a memory address **read from cache - a cache hit**

Use small cache: get overall memory access time close to cache access time

Locality of Reference – get lots of "hits" in the cache

The cache will only be successful in speeding up memory accesses, if code/data items are **referred to more than once** or if it is **possible to prefetch code/data** into the cache.

The first arises in most programs due to what is normally referred to as `locality of reference' - the tendency of programs to localise their memory accesses over a short time period, e.g. through loops.

Cache Performance

The purpose of the cache is ***to reduce the access time to memory*** for the CPU, so a useful measure is:-

average access time, t_{av} = cache access time (t_c) * ***hit rate***

+ memory access time(t_m) * (1 - ***hit rate***)

Example:
Memory Access Time = 50ns
Cache Access Time = 5 ns

Hit Rate	Average Access Time
40%	32 ns
60%	23 ns
80%	14 ns
90%	9.5 ns
95%	7 ns

28/03/2013 14-GC03 Commercial Memories & Cache

hit rate: this is fraction of the number of successful accesses to the cache to the total number of accesses. Hit ratios of 90-95% are commonly quoted.

The average access time is what it says: an average the time of all the memory accesses made. A short access time is dependent on a high hit rate.

Near cache performance can be obtained with Slow DRAMS and a small fast cache.

This is much cheaper and much smaller in size than an all static RAM solution.

Size of Cache

This influences the hit rate.

hit rate increases with increasing cache size

hit rate is a constant above a certain cache size

Cache size is only very loosely related to size of main memory:
it is probably pointless having a cache that is close to the main memory size

Optimum cache size for a particular depends on the particular memory referencing pattern of the program. It will also depend on how long a program runs before it is suspended and other programs are scheduled.

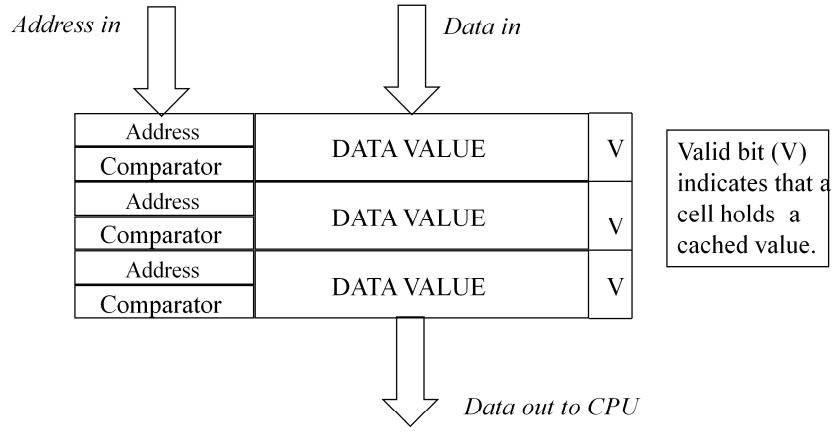
Typical cache size: 16Kbytes - 512 Kbytes [Most PCs used to have 256Kbytes]
Note 2008: Intel Core i7: 64Kbyte L1/core, 256Kbyte L2/core, 4-12MByte shared

A hit rate of 75% (miss rate of 25%) is easy to achieve with a wide range of cache sizes.

Cache Implementations

Fully Associative Cache

(most expensive - usually only small cache sizes are built)

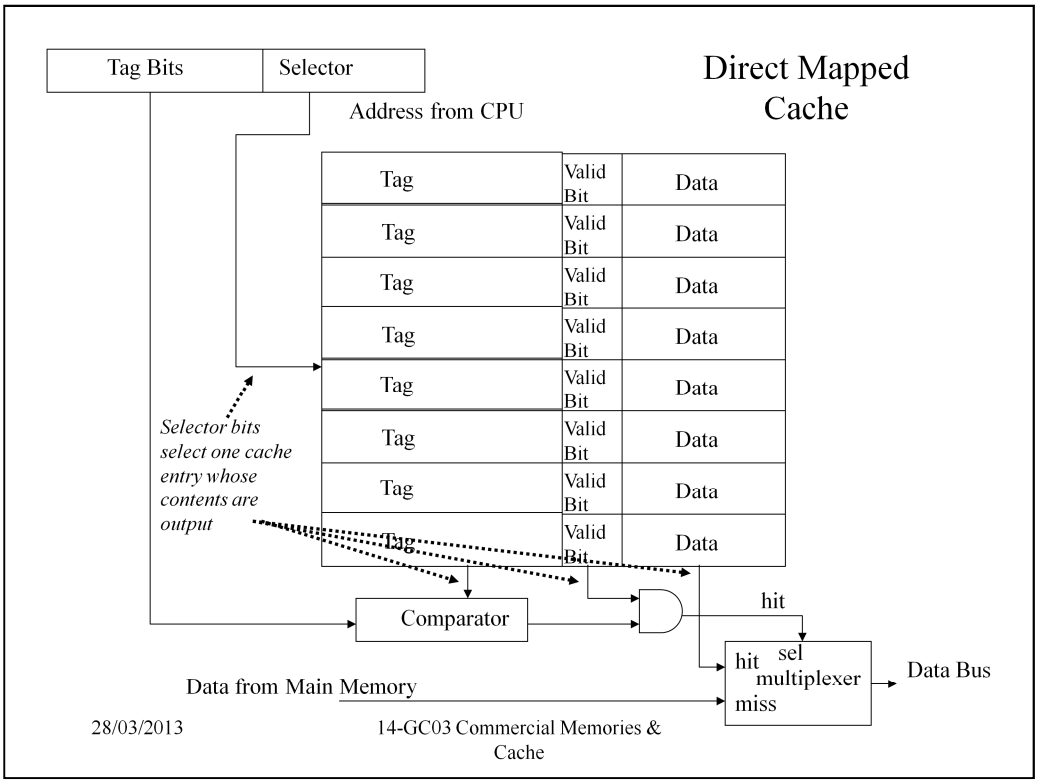


28/03/2013 14-GC03 Commercial Memories & Cache

The implementation and optimisation of caches is a complex field with a large range of possibilities - a field of active research.

This can be thought of as the *ideal* cache. Each cache cell comprises an address cell holding the address of the memory that is cached, a data cell holding the data, and an address comparator. [The cell will also hold a validity or use flag to indicate whether the cache holds valid information or not - reset at initialisation naturally.]

When an address is presented at the cache, the comparators in all the cache cells start up in parallel, comparing their address cell contents with the input address. If a comparator signals an address match the contents of the data cell in that cache cell are output, otherwise a miss is signalled and an access is made to the main memory.



Address from CPU is split into 2 parts. The low part, the selector, selects one entry in the cache.

The high part of the address, the tag, is compared with the tag field of the entry using the comparator.

If there is a match, then the data is read out.

In this system each address can only go into a particular cache line: the selector field of the address must match the line number.

Example: 16-bit address, 256 lines in cache. The lower 8 bits of the address are the selector.

Addresses 0000_{16}, 0100_{16}, 0200_{16}, 0300_{16} must be cache in entry zero!

Addresses 0001_{16}, 0102_{16}, 0203_{16}, 0304_{16} must be cache in entry one!

This mechanism reduces the number of comparators and the number of address bits stored: more chip area for data storage is available, and thus more cache lines are available.

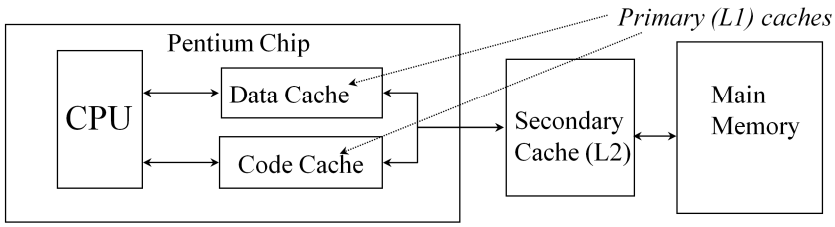
Disadvantages, more conflicts: can have loops where 2 successive memory addresses access the same cache line. The advantages of cache are not obtained because repeat accesses miss because of the intervening access to the same entry.

When the comparator generates a hit, there is a check that the line as valid data (the AND only outputs a hit in this case).

If there is a hit, the data from the selected line is routed through the multiplexer to the data bus and on to the CPU. If there is a miss, the data from Main memory goes to the CPU.

Note: the cache has to be written with new data at times and there are issues of what to do when there is write to a cached or uncached location. These are topics that are not covered here.

Pentium

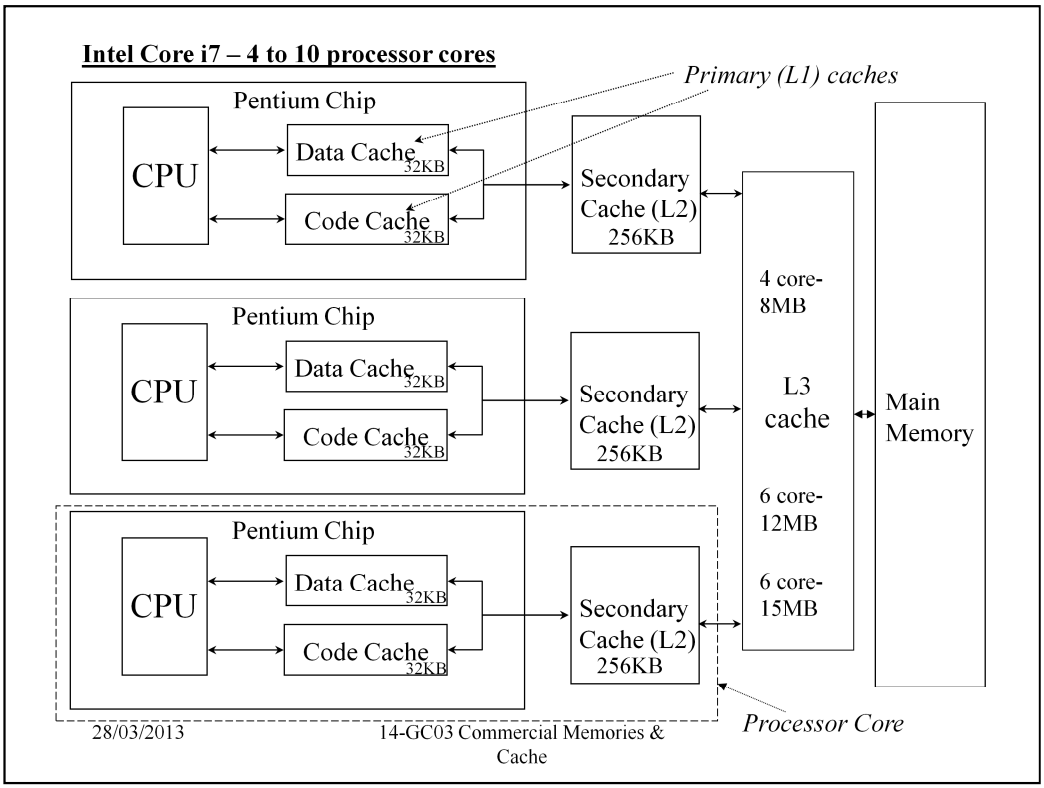


Pentium has

- on-chip Level 1 (L1) caches for code (instructions) and data: these are accessed in parallel in order to overlap instruction execution
- a second level (L2) cache is usual: too big a difference in access time between accessing L1 cache and Main Memory

L1 caches are accessed in a single CPU clock cycle – 3.3GHz
L2 cache (off-chip) access rate is external bus rate – 400/500MHz
L2 cache (on-chip) is CPU clock Rate - 3.33GHz (probably – may be ½ speed)
Main Memory access rate single random access - 25 MHz
Main Memory access rate – burst rate - 266MHz (peak rate)
Burst mode is used to read several locations in a sequence to fill a cache quickly

28/03/2013 14-GC03 Commercial Memories & Cache

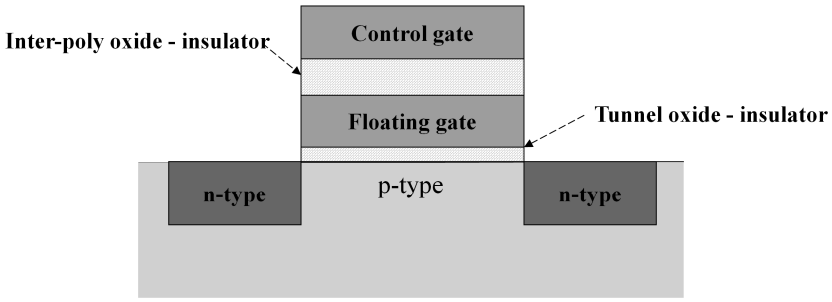


Need arbitration logic to allow cores to access L3 cache and main memory which are shared components.

Also need mechanism to deal with situation where 2 or more L1/L2 caches have a shared data value and one core updates this shared value – new value must be distributed to other caches or other copies must be marked as invalid.

EEPROM – Electrically Erasable, Programmable, Read-Only-Memory
and Flash RAM
Both non-volatile memory – keeps contents when power removed – long term storage

based on ***floating-gate*** transistor

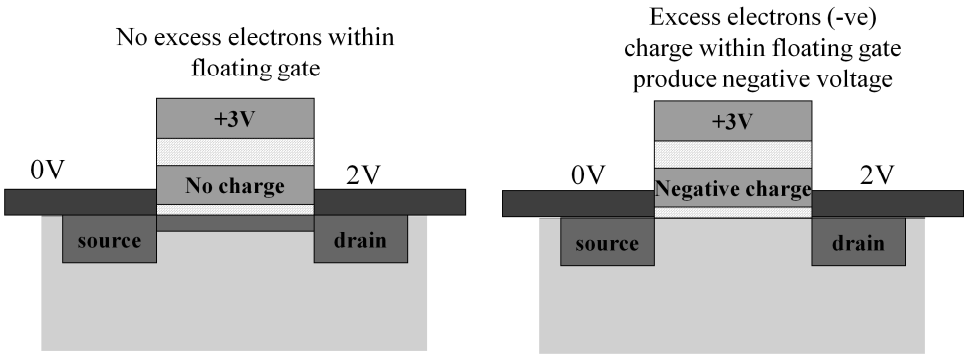


Floating gate is surrounded by insulator and not connected electrically to anything

28/03/2013 14-GC03 Commercial Memories & Cache

How does it work……

by storing extra electrons within floating gate or not



Erased state: Apply control voltage: channel forms and electrons flow from source (0v) to drain
This state represents a "1".

Programmed state: Apply control voltage: negative voltage on floating gate shields channel region from 3V on control gate. No channel forms. No electrons flow from source (0v) to drain.
This state represents a "0".

2 state device – a 1-bit memory with a single transistor

28/03/2013 14-GC03 Commercial Memories & Cache

In the erased state, all storage elements hold a "1".

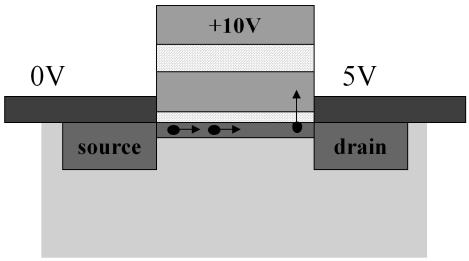
Programming a storage element changes its state to hold a "0".

To turn a storage ellement back to holding a "1" from the programmed state needs an "erase" cycle.

Some devices can only do an "erase" cycle on a block not on an individual storage element.

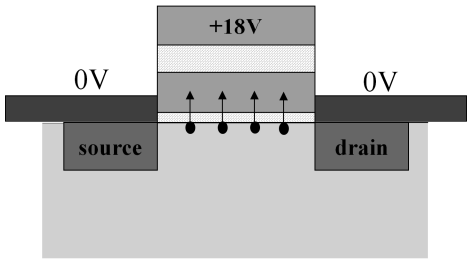
How to programme…….

Program by hot electron injection



10V on control gate. Channel forms and electrons accelerate to drain at 5V.
Pull of 10V field on fast moving (hot) electrons near drain allow them to cross tunnel oxide into floating gate.

Program by Fowler-Nordheim Tunneling



With a high enough voltage on the control gate, electrons can "tunnel" through the tunnel oxide layer.

Once stored in floating gate extra electrons will remain for many years.

NB: To remove charge from floating gate need to "erase"!!!

28/03/2013 14-GC03 Commercial Memories & Cache

Quantum-dynamically the exact position of an electron is unknown – it can be anywhere. Or rather it has a probability of being at any particular position in the universe. The probability of a particular electron in a transistor being a billion light years is almost vanishingly small, nut is non-zero.

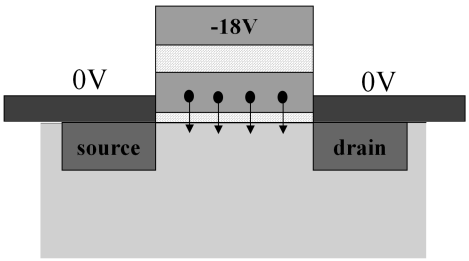
The strong electric field from the control gate pulling on the electrons in the channel region increases the probability that some electrons will "tunnel" through the tunnel oxide insulating layer.

Tunneling:- Imagine you have walked across the Sahara desert with no food and water, and you come across the Atlas mountains towering up a huge wall in front of you. You have nowhere near enough energy left to climb up and over the other side to fertile lands of the coastal region. What you need is a tunnel!

The tunnel oxide insulator presents such a high energy barrier to electrons in the channel region. Electrons have this ability to create a tunnel through the barrier and thus to reach the other side. The high electric field (many millions of volts per meter) increases the probability that any particular electron will tunnel through the insulator. Without the field the probability is very much lower, which is why it takes many years before a significant number of excess electrons stored in the floating gate tunnel there away out.

Erase

Erase by Fowler-Nordheim Tunneling



With a large negative voltage on the control gate, electrons can "tunnel" through the tunnel oxide layer out of the floating gate

EEPROM

General: Write protect input to stop all writes or to limit to access.

Data retention is guaranteed for years, e.g.10 years.

Endurance (number of write cycles at each location) is upwards of 100,000 cycles.

Parallel EEPROM: accessed like static RAM i.e. Random Access and has similar connections to the processor busses.

Individual bytes can be read, with reasonable speed (~150 ns)

Individual writes can be made, which includes an erase cycle and the write (programme) cycle. Writing is much slower than reading (e.g. 10ms).

Used for non-volatile code stores, e.g. start-up code, bios.

Serial EEPROM: a smaller pin-out (8 pins against 32 pins for similar parallel EEPROM).

The address is passed into the EEPROM one bit at time:

for a write the data is then input one bit at a time;

for a read the data from the addressed location is output 1-bit at a time.

Write access times are milliseconds.

Read cycles are slow ~10 µs (with a serial clock rate ~400KHz)

Not used in general computer products.

28/03/2013 14-GC03 Commercial Memories & Cache

NOR-FLASH:

Parallel NOR Flash: **example data for AMIC A29800 1Mx8 Parallel NOR FLASH (48 pin package)– 0ct 2011**

accessed like static RAM, i.e. Random Access and has similar connections to the processor busses.

Used as EEPROM or other ROM replacement.

Data retention is guaranteed for years, example - 20 years.

Individual bytes can be read or written with similar times (example – 55/70/90 ns)

Individual writes can be made to set 1s to 0s: need to erase to be able to set 0s to 1s.

Memory divided into sectors

(example: 1M locations divided into 19 sectors –14 with 64 KB, 1 with 32KB 1 with 16KB, 1 with 8KB

Only sectors (8secs/sector) or whole chip can be erased – not individual locations.

Upwards of 100,000 erase cycles/sector.

Sectors may be protected against writing or erasure by writing control info to device

Serial NOR FLASH: **example data for Numonyx M25P64 8Mx8 Serial NOR FLASH (8 pin package)– March 2010**

Address and data transferred 1-bit at a time. 128 sectors of 64KB organised as 256 pages of 256 bytes.

A single read access can be just one byte (~520ns) or multiple sequential bytes up to entire memory.

A write access is made to a single page and may write from 1 byte to all bytes in page.

write time:– 1.4 ms for 256 bytes.

erase time: 1 sector – 0.5 seconds; whole chip ~68 seconds

28/03/2013 14-GC03 Commercial Memories & Cache

NAND-FLASH:

Parallel NAND Flash: **example data for Micron MT294F16 2Gx8 (48 pin package – 26 not-used)–2006**

Organised by planes, blocks and pages. Example: 2 planes; 8192 blocks/plane, 64 pages/block, 2112 bytes/page.

Accessed by page: page write ~220µs. Used as disk replacement.

Page write:– write page of data into internal cache memory and then write this to flash page

Read: read page into internal cache memory,

then read out whole page byte-at-a-time in sequence (25ns/byte) with no addresses needed

or randomly read bytes (max 25 µs/byte) from page in cache (needs address within cache to read).

Only block erasure – all 64 pages are erased – not individual pages: time 1.5ms for the block.

Upwards of 100,000 erase cycles/block with suitable external error correction management.

Data retention – 10 years.

Serial NAND FLASH: **example data for Micron MT294F1 1Gx8 Serial NOR FLASH (8 pin package)– 2009**

Address and data transferred 1-bit at a time. Basically a parallel NAND Flash with a serial interface.

Slower to operate but smaller package and may have error correction management on chip.

Nand Flash-Overview

Speed: no moving parts much faster than disk drives; reliability and erase cycle limit are issues.

Cost (30/11/2011): Flash solid state drive: £85 for 64Gbytes; Sata hard drive: £100 for 1Tb

28/03/2013 14-GC03 Commercial Memories & Cache