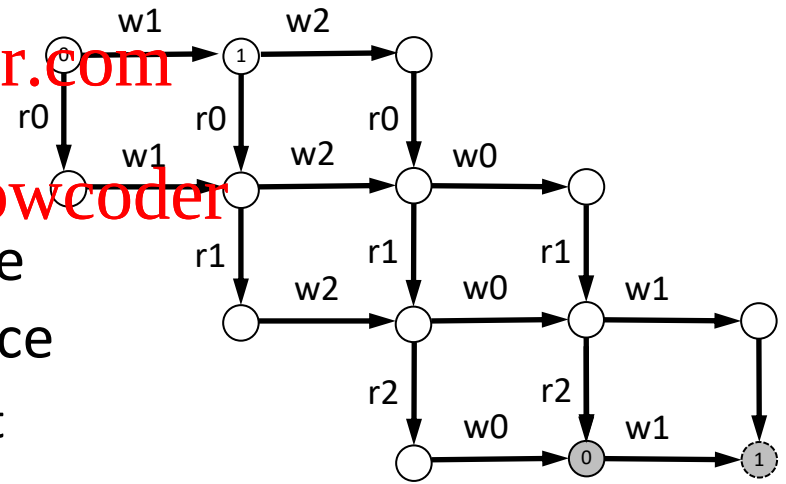
Computer architecture topics

EEE8087

Dr Fei Xia and Dr Alex Bystrov

Quiz from last seion

- Basically remove the overwriting and re-reading arcs from the pool model
- A fundamental difference is that these memory units are no longer slots, they are spaces
- In the other types of ACMs slots are functionally the same memory space
  - esp. when data freshness is important



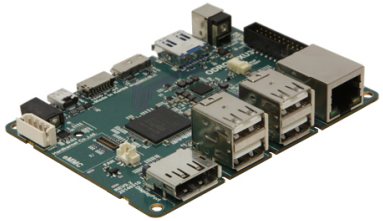
Architecture topics

- Computer architecture is an extensive topic covering aspects of both software and hardware
  - Many large books are available
- We will cover only a few selected topics most relevant to embedded and real time systems
  - Memory: multi-level structures (cache etc.)
  - Interconnects: bus vs networks on chip
  - Processors: multi-core structures
  - Execution: instruction pipelining
  - Special topics: parallelization, Harvard vs Von Neumann, CISC vs RISC

Computing is at the heart of ICT

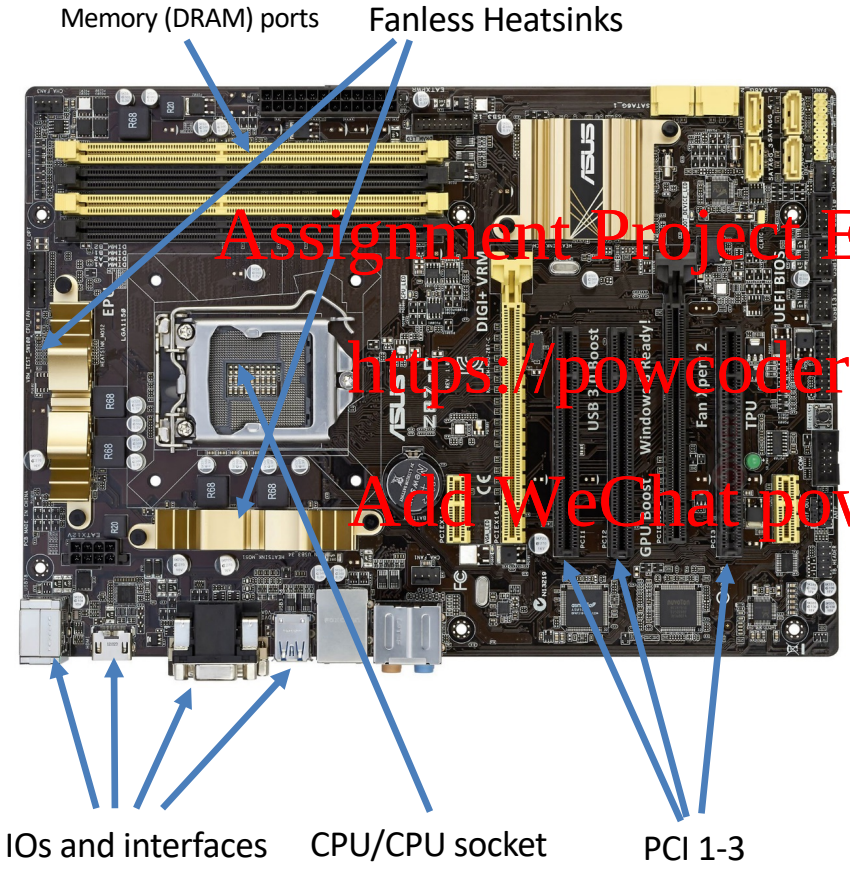
Embedded Computing

- Battery operated mostly or at least limited power supply
- Parallelism depends on performance requirements
- Modern mobile computing systems feature 4-8 cores
- Body sensor controllers have simpler single core systems
- Has to be light-weight and low cost (and sometimes highly reliable)
- Modern cars have hundreds of systems in various places



Odroid XU3, by hardkernel.com

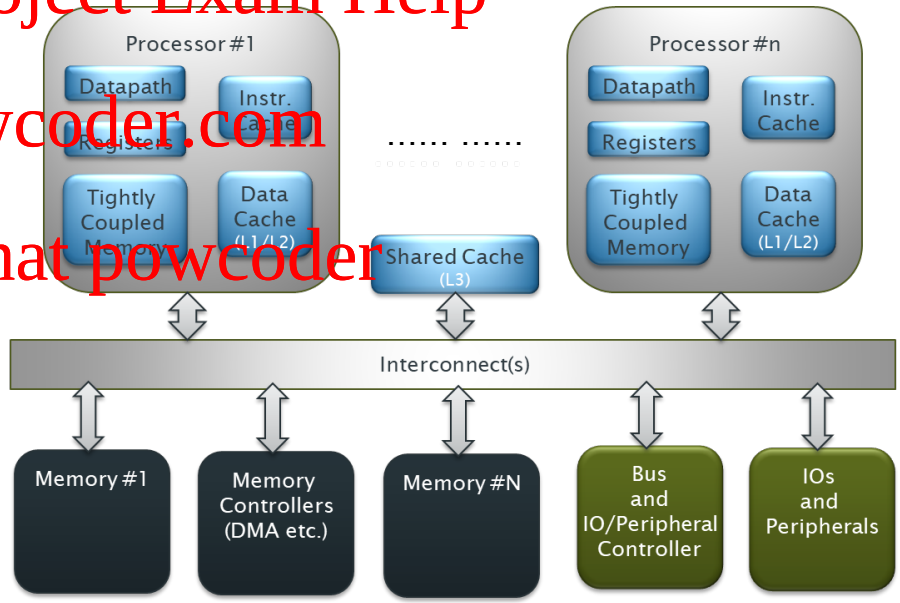
A Typical Computer Architecture (PC)



- Core to computation is the processor and memory organisation
- IO devices allow various user-interfaced functionality
- Modularisation keeps things simple

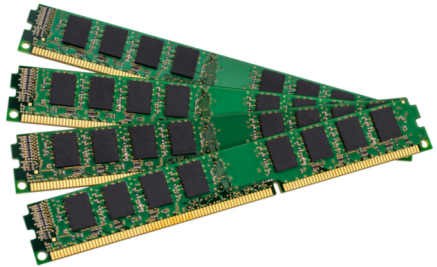
Key components of a computer

- Memories
- Busses/Interconnects
- Processors (CPU, etc.)
- IOs and controllers



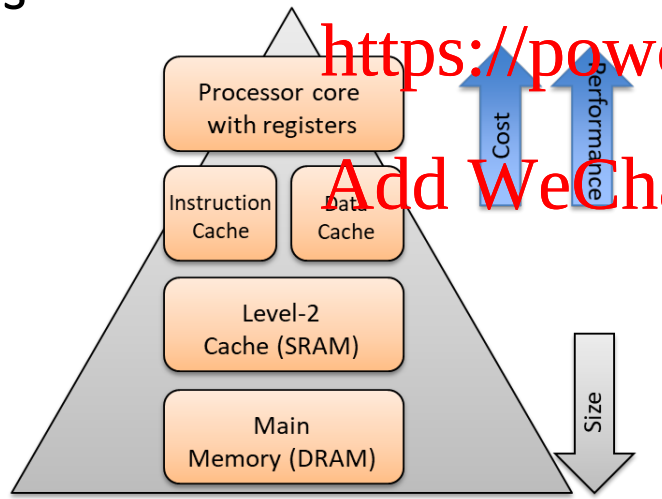
Memory

- Memory is organised in hierarchy with different types:
  - Register:
    - Small and fast memories within the CPU; consists of a number of flip-flops
    - Volatile in nature – flashed out when powered off
  - Cache Memory:
    - Larger but slower memories, usually Static Random Access Memories (SRAMs)
    - Volatile in nature – flashed out when powered off
  - Main Memory:
    - Even larger and slower; main storage; usually Dynamic Random Access Memories (DRAMs)
  - Secondary Storage:
    - Larger and even slower; usually disk drives
  - Other Memories: Offline/Flash Memories:
    - Used to store offline information, not critical to CPU or applications



Different levels and trade-offs

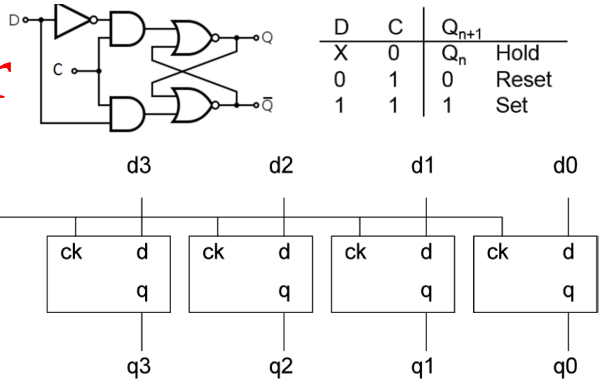
- Memories of different types, speeds and sizes are used depending on their distances from CPUs



Memory	Access time	Capacity
Registers	1 ns	1 Kb
Cache	10 ns	8 Mb
Main memory	100 ns	8 Gb
Hard drive	10000000 ns	4 Tb

Register

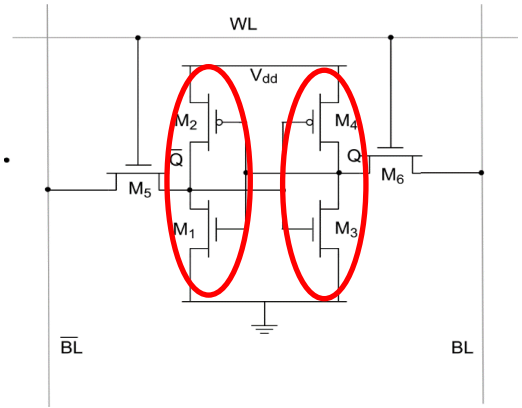
- Each register is (usually) a parallel set of *N* flip-flops with all clock lines connected together
  - *N* is likely the computer's word length (32-bit CPU has *N*=32-bit registers)
- A set of registers form a register file
- Usually on the same silicon as the CPU and considered part of the CPU
- Usually has the same clock as the CPU
- Each flip-flop usually requires well over a dozen transistors
- Usually very small in capacity
  - Difficult to have register-only compute



Cache memory

- Usually static RAMs (SRAMs)
  - Static: holds data as long as power is applied
  - Volatile: cannot hold data if power is removed
- Each SRAM cell requires multiple transistors
  - 6T SRAM is the most commonly understandable example
- Three operations
  - Hold, write and read

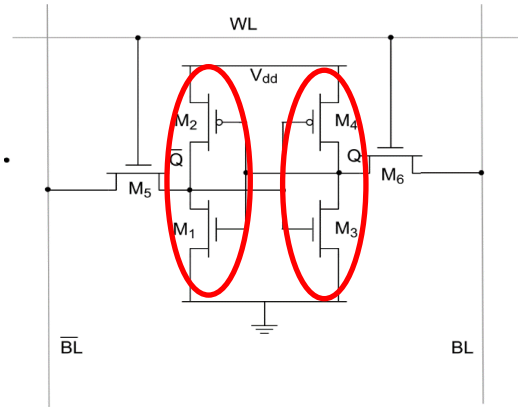
A 6T SRAM cell



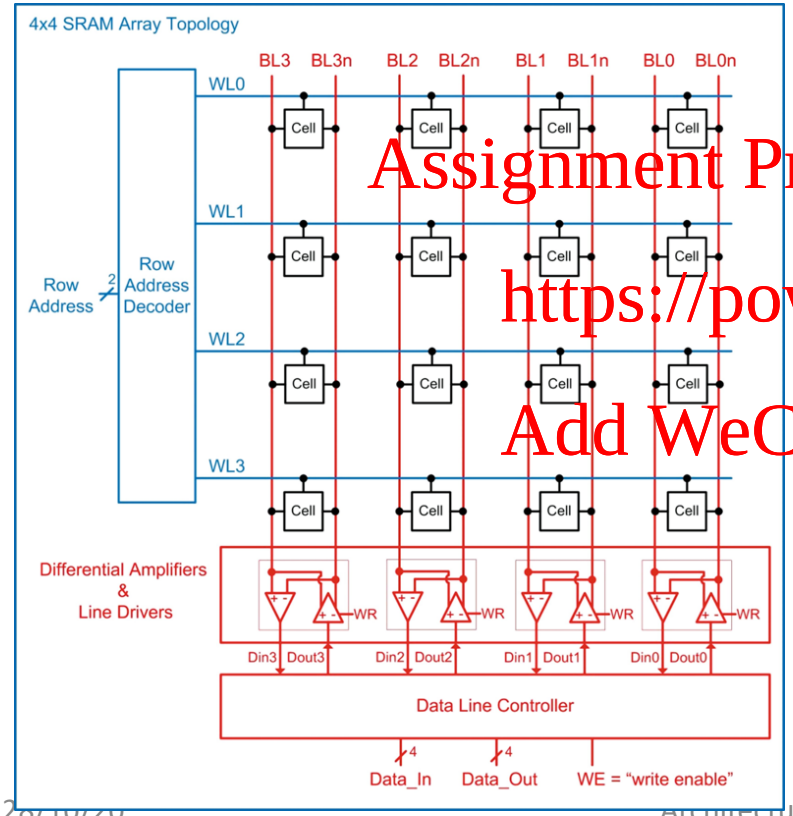
SRAM basics

- Hold (WL = 0)
  - Access transistors M_5 and M_6 are off
  - Data held in latch (the cross-coupled pair of inverters formed by the other four transistors)
- Write (WL = 1 and BL has stable data value)
  - M_5 and M_6 are on
  - Latch is written with the data held on the BLs, if this is not the same as the existing value, then there should be enough current flowing from the relevant BL to overwhelm the stability of the cross-coupled inverters and write the new value in
- Read (WL = 1 and BL precharged to half VDD)
  - M_5 and M_6 are on
  - Stored bit held in the two inverters supplies enough current to the BLs so that the BL's are no longer equal voltage
  - This (slight) difference is amplified by the sense amplifier to push the BL's apart to 0 and 1

A 6T SRAM cell



SRAM basics



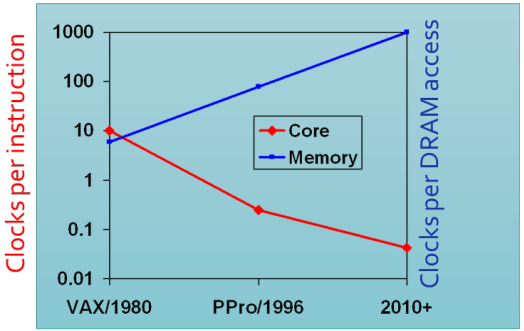
- Hold
  - Data in cells, cells not connected to bitlines
- Write
  - Bitlines have binary values
  - Wordline selects the word where these values are written to
- Read
  - Bitlines precharged BLk=BLkn=0.5
  - Wordline selects cells to read from
  - Cell sends current to bitlines making BLk≠BLKn (not full 0/1)
  - Sense amplifiers detect small differences and drive bitlines to full 0/1

Why precharge?

- Cells are small and bit lines are long
- Bit lines are much larger capacitors than the cells
- This is necessary so that the bit lines can reach every word in a block of memory (e.g. a block of 1024 words) and makes it easy to erase the cells during writing
- But during reading the cells would not be able to drive the bit lines
- Charging bit lines to half way gives cells enough opportunity to split two bit lines away from the middle
- Enough difference for sense amps to act and recreate full 0 or 1 on each bit line
- For SRAM, precharge can be to 0.5VDD (easy to understand logically) or full VDD (more likely in reality, but why and how would it work?)
- We pay all this price to reduce memory cells from ~20T for flip-flops to ~6T for SRAM and ~2T for DRAM

The memory wall

- Faster cores and more cores improving instruction speeds
- Memory latency is increased when the overall capacity keeps increasing
  - Additional communications bandwidth needed
  - Complex organization needed
- The latency gap between processors and memory is increasing
  - Making memory-intensive operations suffer
- This is called the memory wall
  - Overall performance hits a wall at the memory interface, no matter how you improve processors
  - Especially acute with regard to DRAM

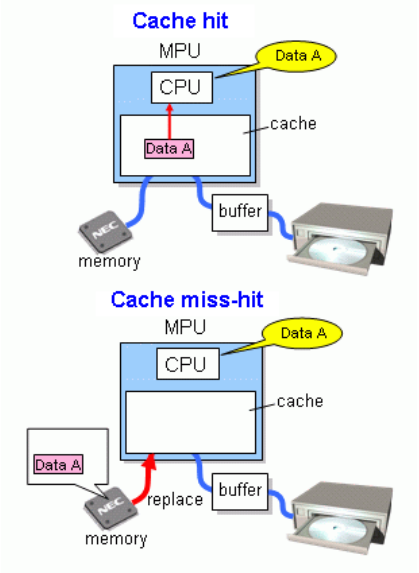


Cache memory justification

- Potentially multiple layers of memory of different speeds are used between fast processors and slow DRAMs
- At the interface between each pair of layers, the slower layer pretends to be as fast as the faster layer – the faster layer sees fast speeds most of the time
- Taking all this together, the processor hopefully sees register or close speeds but gets to use DRAM capacity of memory
  - For instance, in a current-generation smartphone, the cores may be deceived into thinking that they are running with 8GB of fast flip-flops or at least SRAM
  - But in reality they only have 8GB of very slow DRAM and less than a kB of real flip-flops and limited SRAM

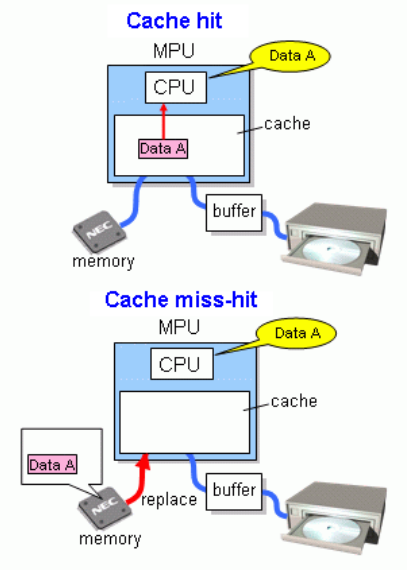
Cache memory operation

- Cache exploits the **principle of locality**:
  - Programs tend to execute the same small set of instructions and data repeatedly
- When the processor needs data for execution, it first checks the cache
  - If the data is in the cache, it is called a HIT
    - This is then fetched at cache speed and the program executes at high speed
  - If the data is not in the cache, it is called a MISS
    - This requires going to further layers of cache before a HIT happens
    - Eventually this may go to the main memory
- Size and performance trade-offs:
  - Larger cache may increase HIT ratio, but the overall access time may increase
    - Due to cache coherence protocols [i.e. update the cache with memory]
  - Smaller cache may increase MISS ratio
  - Cache management needs to be carefully done
- Caches inherently increases timing unpredictability
  - Need to be very careful about cache use when designing real-time systems



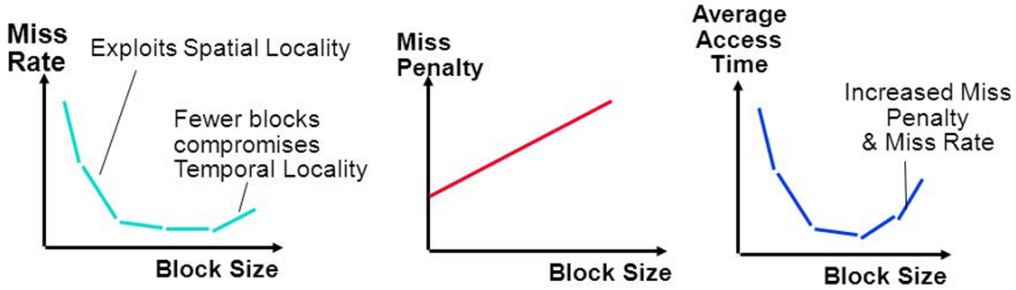
Cache memory operation

- When the CPU writes to memory, the cache copies the data, and then does the write operation.
  - 100 fetches on the same instruction will result in a slow and 99 fast read operations. The speed-up factor is almost 10.
- When the cache becomes full, "old" entries can be overwritten. They are usually "randomly" chosen.
- When cache entries are more updated than main memory, the entries are marked as dirty entries.
  - Dirty lines are updated in the main memory soon
  - Before that, you cannot use this, esp. when you have multiple cores accessing the same memory
  - The process of ensuring integrity of data is called cache coherence



Cache size tradeoff

- Caches are organized in blocks
  - e.g. 1kB of cache organized in 32-Byte blocks
  - Temporal locality refers to the reuse of specific data and/or resources within a small time duration
  - Spatial locality refers to the use of data elements within relative close storage locations – it would **help** if these are located in the same block so larger blocks help this
  - However, larger blocks lead to fewer blocks – this would reduce temporal locality
  - Larger overall size reduces the speed by requiring more complex addressing

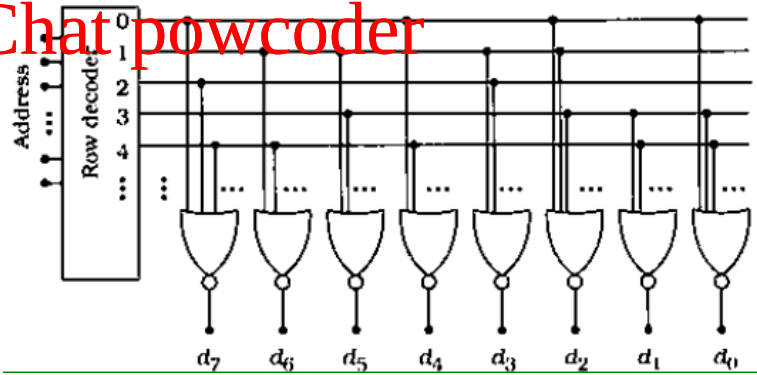


Cache recap

- Ideally the CPU should be supported by e.g. 16GB of flip-flops operating at its own clock frequency
  - Not possible, 16GB is only possible with DRAM
  - Memory wall says DRAM is becoming much slower than CPU, e.g. 2 orders of magnitude slower
  - Multiple layers of cache using SRAM are used to deceive the CPU into thinking it has faster memory than reality
  - Level 1 cache hit – the CPU thinks it has 16GB of very fast memory
  - Level 1 miss but Level 2 hit – the CPU thinks it has fast 16GB memory
  - Level 2 miss but Level 3 hit – the CPU thinks it has tolerably fast 16GB memory (the reality is that it probably only has e.g. 2MB per core of memory at this speed)
  - Level 3 miss – the reality hits the CPU hard on the head ☹

Other memories

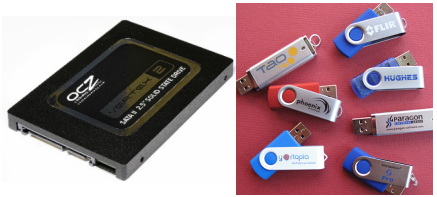
- ROM (read-only memory)
  - Cannot be written to at runtime
  - Data programmed
    - During fabrication or system updates, with high voltages, by control logic
  - Stable storage even when system is powered off
- Example: NOR-based ROM
  - 8b words
  - Address select active row
  - The row select lines program what data is stored where
  - The presence and absence of connection at the crossing points may be fuses

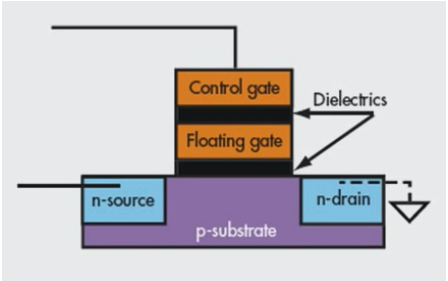


Row	Data
0	01101010
1	10010011
2	01110111
3	11011000
4	00101100

Other memories

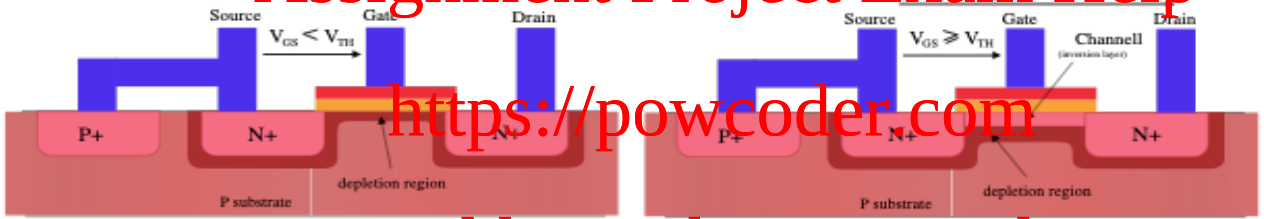
- Flash memories
  - Non-volatile; can be electronically erased and reprogrammed
    - Modern solid-state drives (SSDs) for secondary storage
    - USB flash drives for off-line storage
    - Secure Digital (SD) Cards
  - Semiconductor non-volatile, non-rotary device
    - Generally faster than HDDs (Magnetic Hard Disk Drives)
  - Flash memories are made of storage cells, e.g. single-level cell (SLC):
    - Control Gate (CG) is the traditional MOS gate
    - Floating Gate (FG) electrically isolates the channel
    - Higher threshold voltage (Vthw>Vth) is needed to charge FG for writing
    - A voltage between the two thresholds (Vthw>Vr>Vth) is applied for reading
    - For higher electron biasing hot electron injection is used





Review normal MOSFET

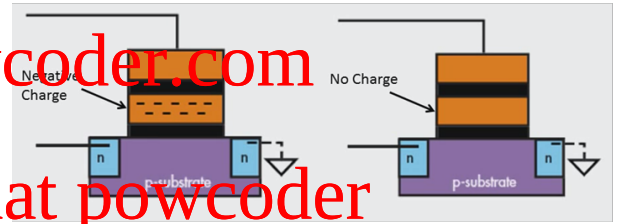
- NMOS structure and operations

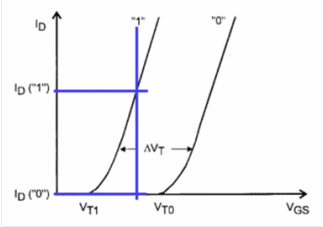


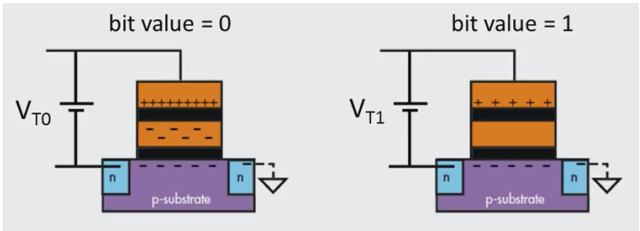
Gate and substrate separated by a thin layer of oxide, voltage applied to gate attracts enough carriers of the opposite type (in this case electrons) to form a channel between source and drain, eliminating the PN junctions and allowing conduction. A floating gate increases the distance between the gate and potential channel, so direct channel-forming by gate voltage is different.

Flash memories

- Two charge states
  - By default no charge, i.e. bit value = 1
- Operations of NOR-based flash
  - Reading:
    - For bit value = 0, higher CG bias (V_{T0} >5V) required to allow current through the channel
    - For bit value =1, lower CG bias (V_{T1}: < V_{T0}) required to allow current through the channel
    - Apply intermediate voltage (V_{T0} > WL > V_{T1}) and sense the current

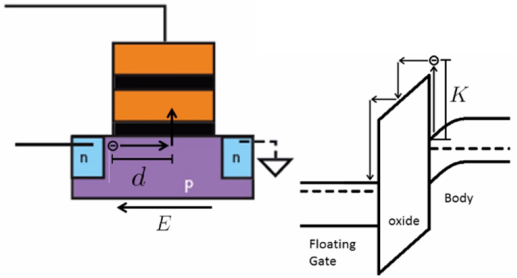






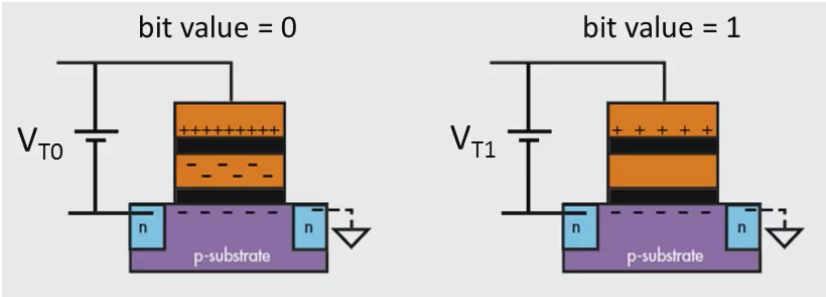
Flash memory operations

- Writing:
  - Challenging as no electrical contact to FG; needs hot electron injection (HEI)
  - HEI follows the principle of quantum tunnelling – i.e. how charge propagation decays with variation in the depletion region
  - HEI applies high negative voltage to channel to allow FG to be charged negatively (write 0)
  - HEI applies high positive voltage to channel to allow FG data to be erased (overwrite with 1)



Flash memory operations

- Hold:
  - The FG charge dictates the hold operation
  - Negative FG charge for bit value = 0
  - No (or positive) FG charge for bit value = 1 (default)
- HEI causes oxide layer damage and this limits/defines the lifetime of flash memories
  - SLCs have typically ~100k write cycles
  - Many modern flashes have redundant memory to allow for higher reliability



Intel cache example (a Core i7)

- Can you reason about why they chose such a cache architecture?
- We will revisit this later