Digital Electronics and Machine Architecture

1 Introduction

1.1 von Neumann Architecture

- Memory holds both programs and data
- Memory is addressed linearly
- Memory is addressed by location, the contents are irrelevant
- Instructions executed sequentially, unless a branch or reset occurs

1.2 Harvard Architecture

This has separate memory for instructions and data

- Quicker to execute as can access instruction and data at the same time
- Simpler to follow/analyze code
- Avoids the potential for some malware/bugs due to self modifying code

2 CPU Architecture

2.1 Components of a CPU

- Memory - RAM
- Registers - Special memory locations that can be accessed very fast
- ALU
- Buses - Wires that carry data between components
- Control Unit - Responsible for directing the flow of instructions and data within the CPU

2.2 Registers

Manipulated directly by the Control Unit
Size of registers can vary from 1 to 128 bits

Accumulator - considered part of the ALU, has general purpose registers for

- Holding data
- Holding interim and final results of arithmetic operations
- Holding data waiting to be transferred between different memory locations
- Holding data waiting to be transferred between different memory locations
- Holding data waiting to be transferred between I/O and memory

Program counter - Holds the address of the next instruction to be executed
Instruction Register - Holds the actual instruction being executed
Flags - 1 bit registers used to keep track of special conditions

Memory Address Register - Holds the address of a memory location to be accessed
Memory Data Register - Holds the value that is being stored to or retrieved from the memory location currently addressed by the MAR

2.3 Buses

Buses are used to:

- Transfer data between the different points on the CPU
- Transfer data between the CPU and main memory
- Transfer data between computer peripherals and the CPU

A bus is a group of electrical conductors (lines) used to carry signal, they have 4 general categories

- Data
- Address
- Control
- Power

Point to point - Bus carries the signal from a specific source to a specific destination
Broadcast - Bus used to carry signals to many devices
Bus interface bridges - Allow communications between the different buses

3 Binary

- Nibble - 4 Bits
- Byte - 8 Bits
- Half Word - 16 Bits
- Word - 32 bits
- Double word - 64 bits

3.1 Decimal fractions to binary

- Repeatedly multiply the number by 2, until the fractional part is 0
- If the ith result is greater than or equal to 1, places 1 in the ith position to the right of the radix, retain only fractional part
- Else, place a 0 in the ith position to the right of the radix

4 Binary Arithmetic and Floating Point

4.1 Negative numbers

One's complement

- The negative of a number is represented by flipping each bit
- Higher order bit indicates the sign of the number

Twos complement

- A negative number is obtained by flipping each bit and adding 1
- Higher order bit indicates sign of number
- One representation for 0

Add a bias

- For k bit numbers, add a bias of 2^{k+1}, then store in normal binary
- Can store numbers between $-(2^{k-1}-1)$ and 2^{k-1}
- The higher order bit does not indicate the sign of the number

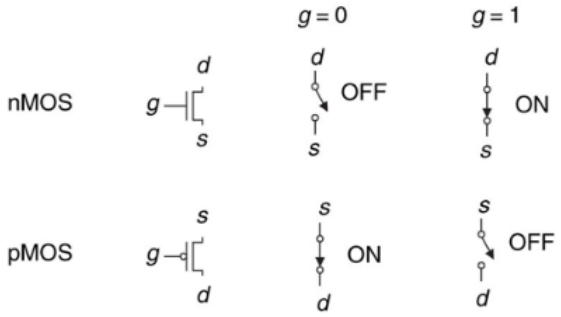
4.2 Floating point representation

Floating point representation has three fields

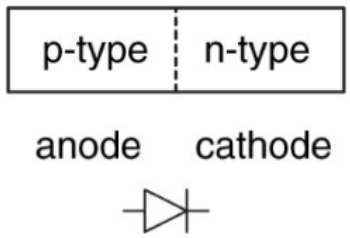
- Sign bit S
  - 0 indicates positive, 1 indicates negative
- Exponent e
  - Stored with a bias of 127
  - 0 and 255 have special meanings
    - Exponent 0 with mantissa 0 gives the number 0
    - Exponent 0 with non zero mantissa - ”subnormal numbers”
    - Exponent 255 with mantissa 0 gives + or - infinity
    - Exponent 255 with non zero mantissa: not a number
- Mantissa M
  - Binary number
  - Always scaled so that the radix point is after the leading 1

5 Logic Gates and Transistors

5.1 Transistors

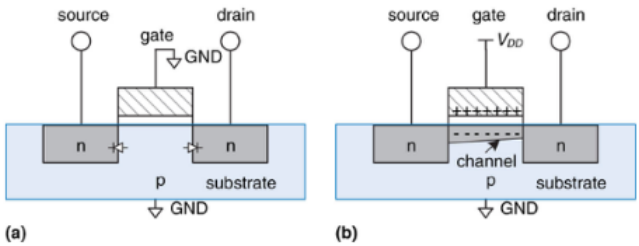


The most common transistor is the MOSFET
Silicon is a poor conductor of electricity: all the available electrons are used to form bonds with neighbouring atoms. Impurities (dopants) provide extra electrons or electron holes which increase conductivity.
Diode

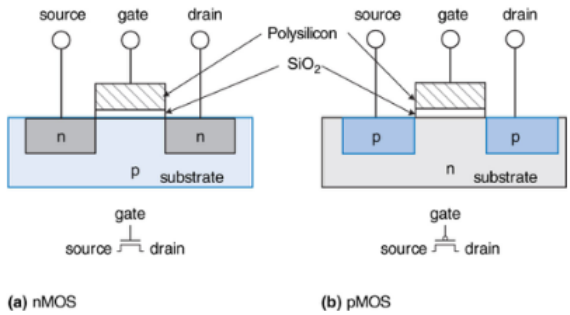


Capacitor - Two pieces of conductive material separated by an insulator

nMOS transistor



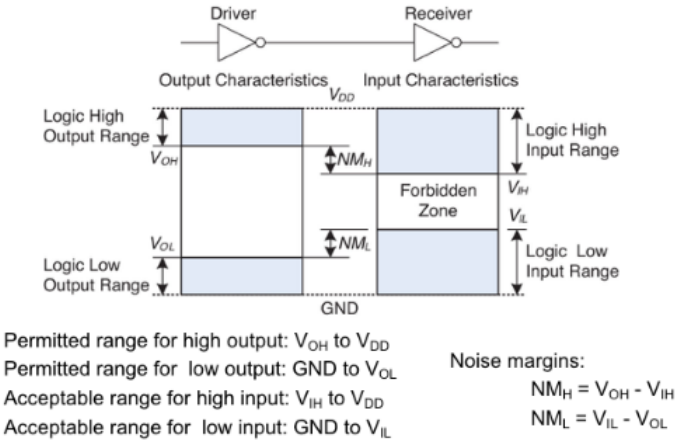
pMOS transistor



5.2 Supply voltage

The **low** voltage of the system is 0V
Historically the high voltage was 5V, called V_{DD}. However modern chips have moved to lower voltages to save power and avoid overloading transistors.
The mapping of the continuous voltage at any point between 0 and 1 is governed by defining logic levels

5.3 Logic levels



Output range

- V_{OH} - Output High
- V_{OL} - Input Low

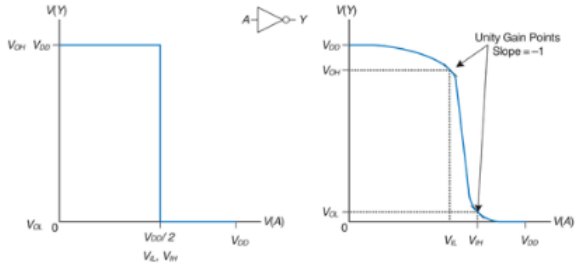
Noise Margins

- NM_H - Noise Margin High
- NM_L - Noise Margin Low

Input range

- V_{IH} - Input High
- V_{IL} - Input Low

5.4 Transfer Characteristics



An **ideal inverter** would output V_{DD} for inputs up to $V_{DD}/2$ and 0 otherwise
However real circuits are not ideal

5.5 The Statistic Discipline

You can only allow circuit elements that all satisfy the same logic levels

6 Boolean Algebra

6.1 Functionally Complete Sets

AND, OR and NOT form a functionally complete set as all propositional logic can be formed from them. It is easiest to prove that other sets are functionally complete by proving that they can form those sets.
For example, it can be shown that NOR is functionally complete

$$\text{AND:} \quad A \bullet B = \overline{\overline{(A + A)} + \overline{(B + B)}}$$

$$\text{OR:} \quad A + B = \overline{\overline{(A + B)} + \overline{(A + B)}}$$

$$\text{NOT:} \quad \overline{A} = \overline{A + A}$$

NAND can also be shown to be functionally complete and uses less silicon for the same performance so are used as universal gates

6.2 Circuits

A circuit has

- One or more discrete valued input terminals
- One or more discrete valued output terminals
- A specification of the relationship between inputs and outputs
- A specification of the delay between inputs changing and outputs responding

A circuit is made up of elements and nodes

- **Element** - A subcircuit
- **Node** - A wire joining elements

6.3 Combinational Logic

Rules:

- Individual gates are combinational circuits
- Every circuit element must be a combinational circuit
- Every node is either an input to the circuit or connecting to exactly one output of a circuit element
- The circuit has no cyclic paths - every path through the circuit visits any node at most once

6.4 Boolean Algebra

- Complement/Inverse - $\overline{A}$
- Product/Implicant - $A \cdot B$
- Minterm - Product involving all inputs to a function
- Sum/Implicant (Yeah, they're both called implicant) - $A + B$
- Maxterm - A sum involving all inputs to a function

As you know from MCS all boolean expressions can be written in sum of product or product of sum form (DNF or CNF)

6.5 Axioms of Boolean Algebra

	Axiom		Dual axiom	Name
A1	B=0 if B≠1	A1'	B=1 if B≠0	Binary field
A2	$\overline{0} = 1$	A2'	$\overline{1} = 0$	NOT
A3	0•0 = 0	A3'	1+1 = 1	AND/OR
A4	1•1 = 1	A4'	0+0 = 0	AND/OR
A5	0•1 = 1•0 = 0	A5'	1+0=0+1=1	AND/OR

	Theorem		Dual axiom	Name
T1	B•1 = B	T1'	B+0 = B	Identity
T2	B•0 = 0	T2'	B+1 = 1	Null element
T3	B•B = B	T3'	B+B = B	Idempotency
T4	$\overline{\overline{B}} = B$			Involution
T5	$B \cdot \overline{B} = 0$	T5'	$B + \overline{B} = 1$	Complements

7 Karnaugh Maps

Boolean expressions can be simplified by spotting terms of the form $PA + P\overline{A}$ (resolution)
Neighbouring 1s in the K-Map are of the form $PA + P\overline{A}$ (note that the numbers ascend with 1 bit difference rather than numerically)

		AB			
		00	01	11	10
C	0	1	0	0	0
	1	1	0	0	0

Forming a Karnaugh Map

1. Create a map so that neighbouring terms differ in the negation of one variable
2. Circle all rectangular blocks of ones in the map using as few circles as possible
3. Each circle must be a power of 2 in each dimension
4. Read off the implicants that were circled (what formula defines that circle)

8 Circuits

8.1 Key Circuits

Combinatorial circuits (Output depends only on current input)

- Adders - Add the contents of two registers
- Decoders - Use a binary number to activate a single line
- Multiplexors - Use a binary number to select an input

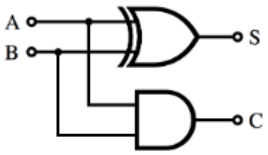
Sequential circuits (Output depends on state and input)

- Latches/Flip Flops - Basic memory element

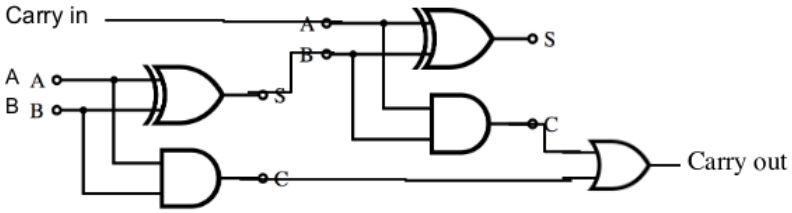
8.2 Half Adder

This has a truth table that follows the basic binary addition rules for 1 bit numbers

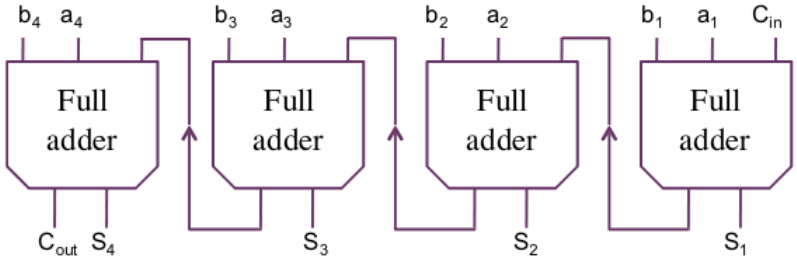
A	B	Sum	Carry
0	0	**0**	**0**
0	1	**1**	**0**
1	0	**1**	**0**
1	1	**0**	**1**



A full adder also takes into account the carry from the previous bit, allowing them to be chained to add any length of number

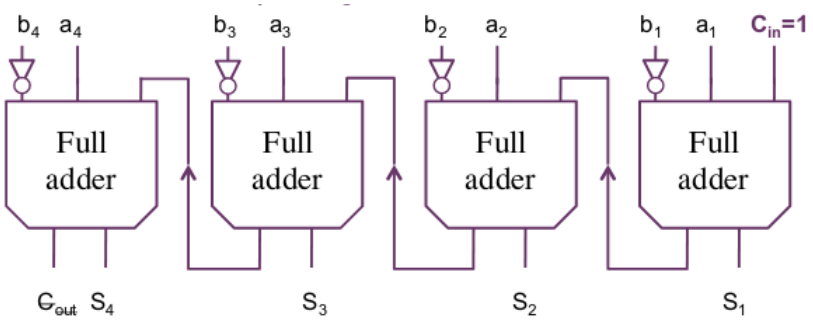


They can be chained like so:



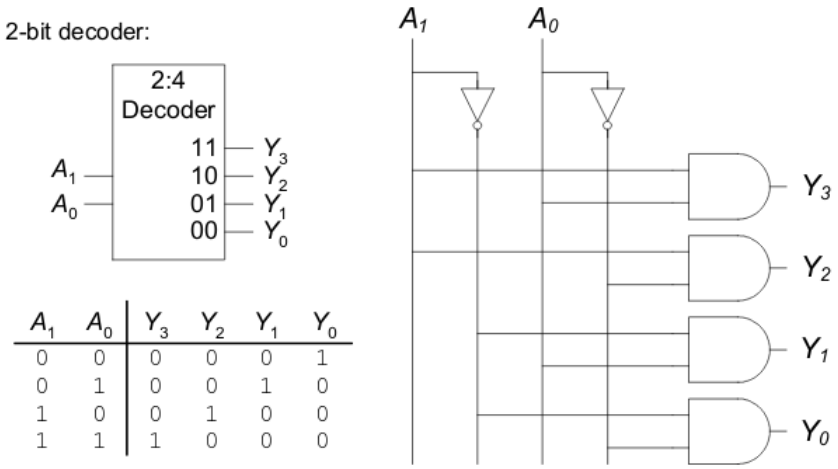
8.3 Subtractor

By creating twos complement negative we can subtract two numbers, with the circuit below



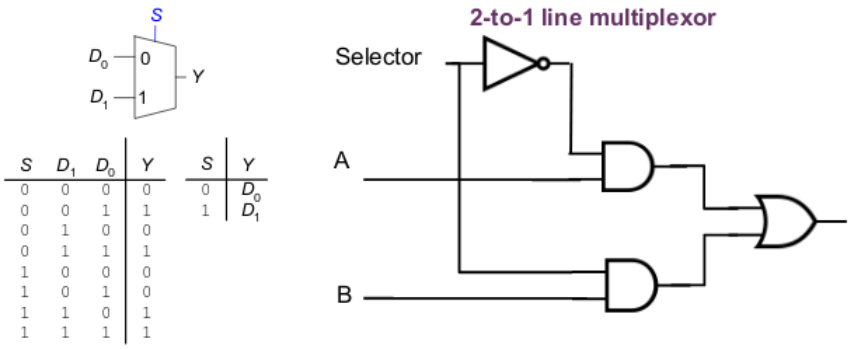
8.4 Decoder

If we want to activate a specific byte from a large amount of memory, this is how we do it. Giving the binary representation of the number as input the output will be the line corresponding to that number.

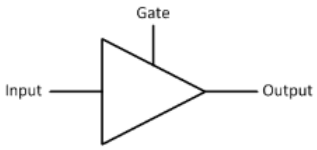


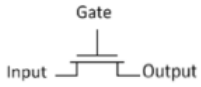
8.5 Multiplexor

This has a selector which takes a binary input and outputs one of the *D* inputs corresponding to that valie



8.6 Tristate

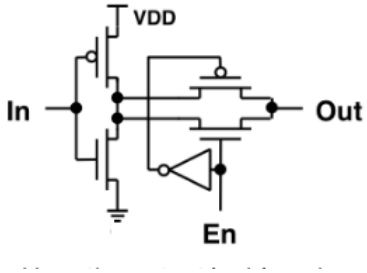




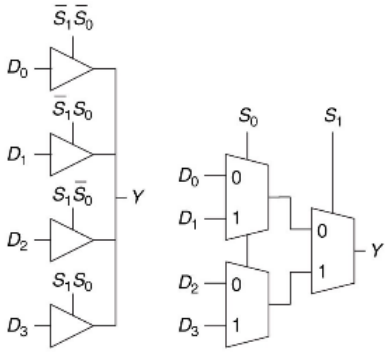
A single transistor could work, but the output is not driven.

Input	Enable	Output
0	0	**Floating**
0	1	**0**
1	0	**Floating**
1	1	**1**

There is also the inverting tristate, which has a wild circuit, but is just the tristate notted



Muxes can be simplified using tristates, also note that muxes can be chained for simplicity



9 Timing and Advanced Adders

Propagation delay (t_{pd}) - The max delay before the output is stable
Contamination delay (t_{cd}) - The min delay before the input changes

Delay is caused by:

- Capacitance and resistance in a circuit
- The speed of light limitation

Reasons why t_{pd} and t_{cd} may be different

- Different rising and falling delays
- A circuit may have multiple inputs and outputs, some of which are faster than others
- Circuit speed with relation to temperature

9.1 Critical paths

Critical path - The longest path in the circuit - this determines the propagation delay
Short path - The shortest path in the circuit - this determines the contamination delay

9.2 Glitches

Glitch - Where an output will temporarily move to an incorrect value before stabilising

9.3 Carry Lookahead Adders

Define two functions: **Generate**: G(A,B)=1 if A and B could cause C_{out} even if C_{in} = 0 (A AND B)
Propagate: P(A,B)=1 if A and B would cause C_{out} if C_{in} = 1 (A or B)

This allows carry out to be calculated as

$$C_{out} = G(A, B) + P(A, B) \cdot C_{in}$$

When expanding to larger numbers of bits, it is better to chain smaller CLAs to save gates

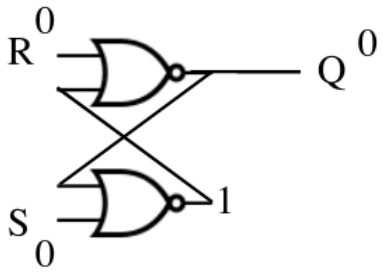
10 Sequential Circuits

- Output depends on history as well as current input (i.e. the circuit has memory)
- Can be modelled as Finite State Machines
- Fundamental components of latches and flip flops

Synchronous sequential circuits - Made from combinatorial components interleaved with banks of flip flops containing the state of the circuit

10.1 SR-Latch

Also known as a NOR latch

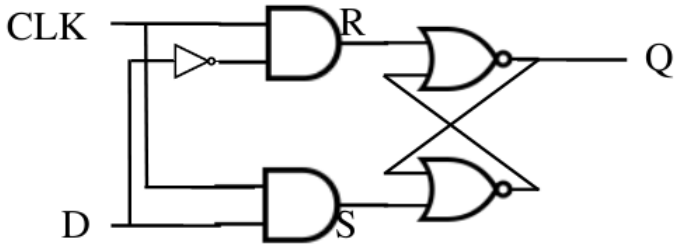


- Both inputs "usually" set to 0
- If input S (set) has a pulse of 1, the output becomes 1
- The output remains 1 even when the pulse is over
- If input R (reset) has a pulse of 1, the output becomes 0
- The output remains 0 even when the pulse is over
- If the output is already 1, a pulse on S will not change it. If it is already 0, a pulse on R will not change it
- It is called **bistable** as it has two stable states for a given input

This gate can also be built from AND gates, but the usual states are 1 rather than 0

10.2 D-Latch

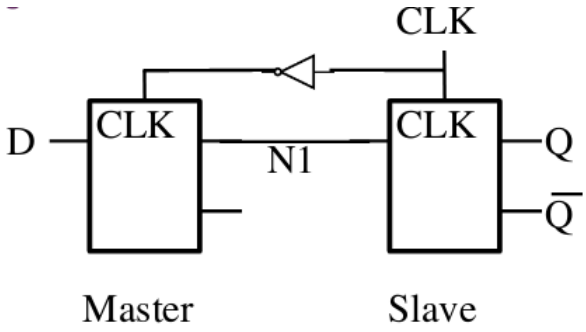
In a latch a pulse on set or reset indicates what the new state should be, and when it should change



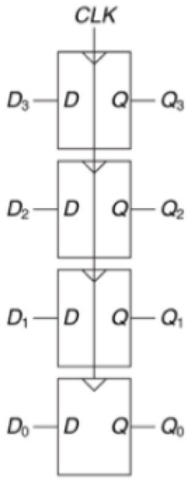
D – **data** input. Defines **what** the new value should be.
CLK – **clock** input. Define **when** the new value should arise.

10.3 D Flip-Flop

Instead of the output changing **whenever** the clock is high, the D Flip-Flop changes **only at the moment the clock goes high**

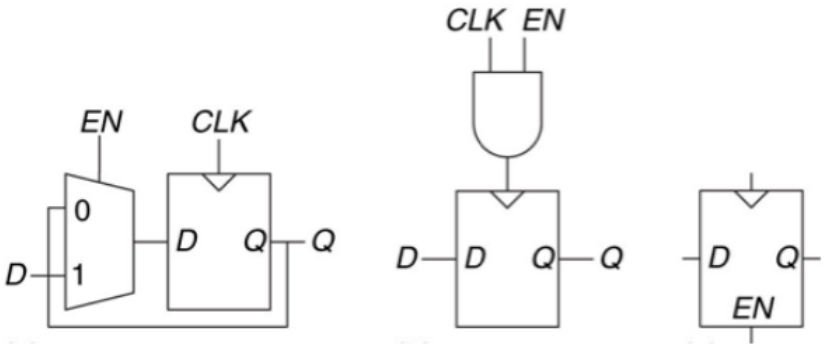


These can be combined to form a register.
Register - A bank of flip-flops driven by the same clock



10.4 Enabled Flip-Flop

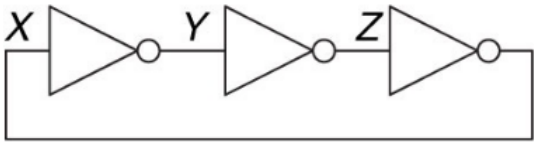
This has an additional input (enable) to control whether the data is loaded into the register or not



Gated Clock - Where there is a gate to control if the clock gets to the flip flop.
There can be timing errors and glitches with Gated Clocks

10.5 Problem Circuits

A Unstable/Astable circuit can be seen below



Race condition - Behaviour depends on which two routes through the circuits carry the signal the fastest

Loops/Cyclic Paths - Where outputs are fed back into inputs
To avoid this we insert registers into cyclic paths:

- The registers contain the 'state' of the circuit
- They break the paths
- They only update on a clock edge
- Say they are synchronised to the clock

10.6 Synchronous Circuits

A synchronous sequential circuit consists of interconnected elements such that:

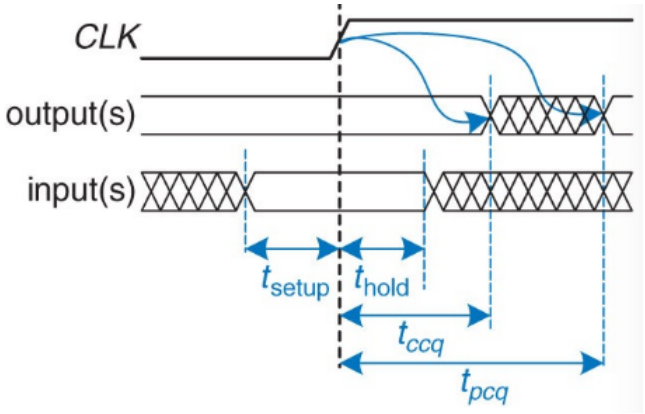
- Every circuit is either a register or combinational circuit
- At least one circuit element is a register
- All registers receive the same clock signal
- Every cyclic path contains at least one register

A synchronous sequential circuit has:

- A discrete set of states $\{S_0, ..., S_{k-1}\}$
- A clock input, whose rising edge indicates when a state change occurs
- A functional specification which details the next state and all outputs for each possible current state and set of inputs

10.7 Timing

- t_{setup} - Time before rising edge during which inputs must be stable
- t_{hold} - Time after rising edge during which inputs must be stable
- t_{ccq} - Time until output starts to change
- t_{pcq} - Time by which output has stabilized



10.8 Setting Time

Time between ticks (T_c) must be at least $t_{pcq} + t_{pd} + t_{setup}$
Sequencing Overhead - $t_{pcq} + t_{setup}$

There is also a minimum delay requirement After the clock change, there is a time of $t_{ccq} + t_{cd}$ before the output starts changing, so t_{hold} needs to be at least this

10.9 Metastable states

Metastable state - A state that will be driven to 1 or 0 eventually, but may take time

10.10 Synchronisers

A pair of flip flops can be used to synchronise the input with the clock

11 Memory

3 Common Types:

- Dynamic Random Access Memory
- Static Random Access Memory
- Read Only Memory (ROM)

Memory is constructed with N address bits and M data bits

- 2^N rows and M data bits
- Depth: Number of rows (number of words)]
- Width: Number of columns (size of word)
- Array Size: depth $\times$ width = $2^N \times M$

Wordline:

- Single row in a memory array to be read/written

- Only one wordline is HIGH at once

On memory read:

- If the wordline is active (high) the stored but should drive the bitline value
- If the wordline is not active (low) the stored bit should not affect the bitline

Memory consists of latches, each holding a single bit value, at an address

MAR holds the 'open' address
MDR holds the activated data

11.1 Lines

Wordline/Address Line - On when computer is addressing the data in that cell
Read/Write Line - Determines whether the data will be transferred to the MDR (write) or from the MDR(read) from the cell

Read occurs when:

- Address Line=1 AND R/W line=1

Write occurs when:

- Address line=1 AND R/W line=0

11.2 DRAM

Each bit is one transistor and one capacitor
Need refreshing as capacitors discharge

11.3 SRAM

6 (or more) transistor flip-flop based memory
Low density but stable and fast

11.4 ROM

In a similar way to DRAM, but replace capacitor with connection to ground or high
PROM (Programmable ROM) works in a similar way, but has fuses connecting each transistor to high, these can be blown to create the data

11.5 Memory Interleaving

Basically RAID 0

11.6 Cache hit and miss

Cache miss - Look something up in memory and it is not found
Cache hit - Look something up in memory and it is found.

12 Addressing Modes

Sometimes we would like to

- Address a large amount of memory with only a few bits
- Use indexes to loop or examine a table or array
- Relocate data or programs in the memory
- Operate on the registers rather than the actual memory

This can be achieved using alternate addressing modes

- Direct addressing - The address read in the instruction is the address of the actual data
- Immediate addressing - The address read in the instruction is the data to be used
- Indirect addressing - The address read in the instruction is the address of a memory location containing the address of the actual data
- Register indirect addressing - The address read in the instruction is the address of a register containing the address of the actual data
- Indexed addressing - The address read in the instruction should have an index value (contained in some register) added to obtain the address of the data

Alternative to absolute addressing

- Absolute Addressing - The address is the one containing the data
- Relative addressing - The address is an offset to the current instruction address
- Base offset addressing - The address read is offset by the current value in a special 'base' register

13 Microarchitecture

Architecture - Consists of instruction set and implies an architectural state
Microarchitecture - How to implement an architecture in hardware, made of

- Datapath - functional blocks and registers
- Control - Control signals