3. Hardware and Virtual Machines

15.1 Processors, Parallel Processing and Virtual Machines

RISC (Reduced Instruction Set Computers)

- Uses few instruction sets & addressing modes
- Uses fixed length, single-clock cycle instructions
- Instructions only require one clock cycle (single-cycle)
- Uses many general purpose registers and a hard-wired CPU
- Makes use of pipelining – executes instructions in parallel, where the output of one instruction is the input of the next
- Makes extensive use of RAM
- Design emphasis is on software

Pipelining

- Allows several instructions to be processed simultaneously – used to increase instruction throughput during FE cycle

Process

- Instructions are divided into subtasks
  - ➔ Instruction fetch
  - ➔ Instruction decode
  - ➔ Operand fetch
  - ➔ Opcode/Instruction execute
  - ➔ Result store
- Each subtask (instruction stage) is completed during one clock cycle
- No two instructions can execute their same stage at the same clock cycle
- The second instruction begins in the second clock cycle, while the first instruction has moved on to its second subtask etc. – while one instruction is being executed, the next one can be fetched and so on...

CISC (Complex Instruction Set Computers)

- Uses many instruction formats and has a large instruction set
- Uses variable length, multi-clock cycle instructions
- Makes use of many different addressing modes
- Uses few registers and a programmable CPU
- Design emphasis is on hardware – requires complex circuits
- Makes frequent use of cache memory

Interrupt Handling (In both RISC and CISC)

1. **The processor detects an interrupt at the start/end of the FE cycle**
2. **The current program is temporarily stopped and the status/contents of each register are stored on a stack**
3. **The appropriate ISR routine is called and executed**
4. **After the interrupt has been serviced the registers are restored to their original status (data is restored from the stack)**

Effect of Pipelining (only in RISC)

- **Adds additional complexity – there could be a number of instructions still in the pipelining when interrupt is received**
- **All currently operating instructions are discarded except for the last one – the interrupt handling routine is applied to the remaining instruction**
- **Once interrupt has been serviced the processor can restart with the next instruction in the sequence**

Computer Architecture

SISD – single instruction, single data

- **One processor executes a single instruction using the same data set – data is taken from a single source and only a single instruction is performed on it**

SIMD – single instruction, multiple data

- **Processors execute the same instruction on multiple different data sets simultaneously**
- **Instructions can be performed sequentially (Pipelining)**
- **Parallel computers with multiple processors**

MISD – multiple instructions, single data

- **Performs different instructions/operations on the same set of data**
- **Each processor works on the same data set independently**
- **Parallel computers with multiple processors**

MIMD – multiple instructions, multiple data

- **Consists of many processors that operate asynchronously/independently**
- **Any processor can execute different instructions on different data sets**

Massively Parallel Computers

- **kind of network infrastructure**
- **a large number of computer processors or separate computers connected together – simultaneously performing a set of coordinated computations**
- **communicate using a message interface (send messages between each other)**

Virtual Machines

- **Are an emulation of a computer system (and hardware/software) using a host computer system**
- **Uses guest operating system for emulation**

Benefits

- **cost saving – new system can be tried on different virtual hardware without the need to purchase the hardware**
- **different instruction set architectures can be emulated on a single computer**
- **the system can crash without affecting the host machine – virtual machine provides protection to other software**
- **security – if a virus is downloaded on the emulated system, it only affects the VM**
- **more than one new computer system can be emulated – allows multiple new systems to co-exist on a single computer (multiple VMs can be used on the same computer)**
- **allows emulation of programs for new CS that are not compatible with the host operating system – through the use of a guest operating system**
- **can emulate old software on a newer system using a compatible guest OS**

Limitations

- **cannot emulate some hardware – new hardware might have been developed after the VM was**
- **using the machine means extra code has to be executed and more load is put on the host computer – is less efficient, increases processing time, uses more RAM (hence has poorer performance)**
- **increases the maintenance expenses as both host system and the virtual machine must be maintained (is also more complex to manage & implement)**
- **VM may be affected by weaknesses of the host machine**

Roles of Host Operating System

- **Is the normal OS used by the host machine**
- **Has control of all the resources of the host machine/computer (and can access all the physical resources)**
- **Provides a user interface to operate the virtual machine software**
- **Runs the virtual machine software**

Roles of Guest Operating System

- **OS that runs within the virtual machine**
- **Controls the virtual hardware/software during the emulation – accesses actual hardware through the virtual machine and host OS**
- **Provides a user interface for the emulated hardware/software**
- **Runs under the control of the host OS**

15.2 Boolean Algebra and Logic Circuits

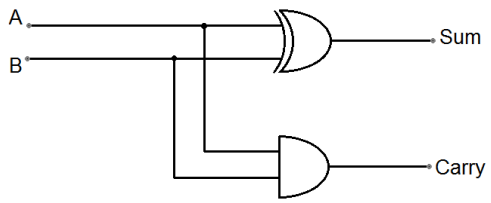
Combination Circuits – output depends entirely on the input values

Half-Adder Logic Circuit

- **Used to perform binary addition**
- **Outputs a sum bit and a carry bit**

INPUT		OUTPUT	
X	Y	sum	carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

- **Consists of a XOR and AND gate – XOR outputs sum, AND outputs carry**

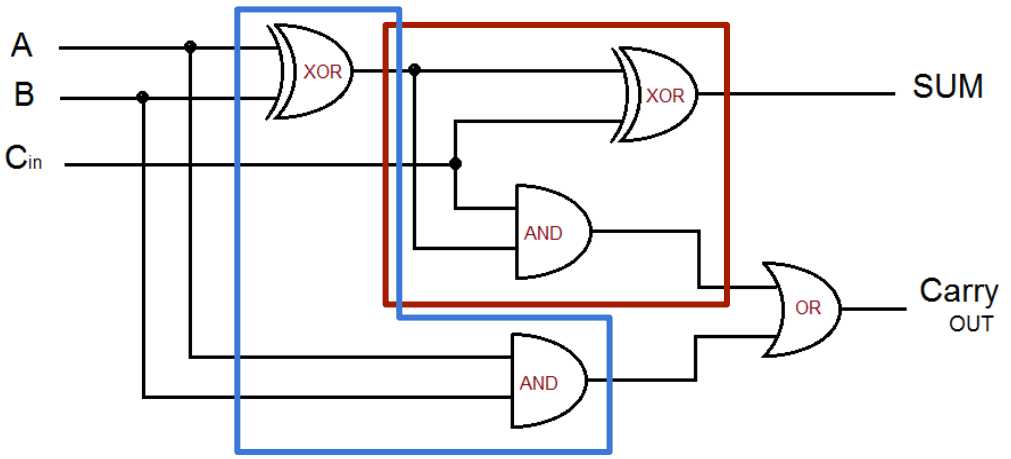


Full-Adder Logic Circuit

- **Two half-adders combined**
- **Has 3 inputs (X, Y and C_{in}) and 2 outputs (S and C_{out})**

INPUT			OUTPUT	
X	Y	C_{in}	sum	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

- **Two half-adder circuits whose outputs are joined through an OR gate**



Sequential Circuits – the output depends on the input value produced from a previous output value

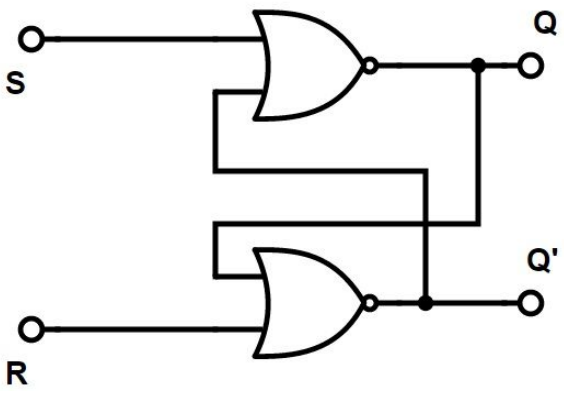
SR Flip-Flop

- stores a single bit of data – 0 or 1
- used in memory to create storage cells within RAM – each flip-flop stores a data bit (memory is made up of many SR flip-flops)

Problems of SR Flip-Flops

- potential for circuit to arrive in an uncertain state
- circuit can become unstable if inputs do not arrive simultaneously
- Consists of 2 inputs & outputs
  - ➔ "SET" labelled S
  - ➔ "RESET" labelled R
  - ➔ 'Q' and 'NOT Q' as outputs
- Made up of 2 cross-coupled NOR or NAND gates

1. Using NOR Gates



If both S and R = 1, an invalid

occurs as Q and NOT Q will be equal.

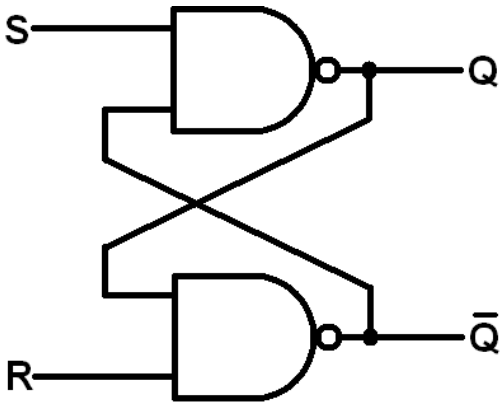
(both SET and RESET cannot be done at once)

Any other combination is valid. If both S and R = 0, there is no change.

** when assigning values, S and NOT Q will be the same, and R and Q will be the same (aim is to get the gate inputs to be either 00 or 11)*

INPUT		OUTPUT	
S	R	Q	NOT Q
1	0	0	1
0	1	1	0

2. Using NAND Gates



If both S and R = 0, an invalid condition occurs as Q and NOT Q will be equal.

(both SET and RESET cannot be done at once)

Any other combination is valid. If both S and R = 1, there is no change.

** when assigning values, the same applies; S and NOT Q will be equal, R and Q will be equal*

(truth table is the same, only difference is when no change or invalid state occurs)

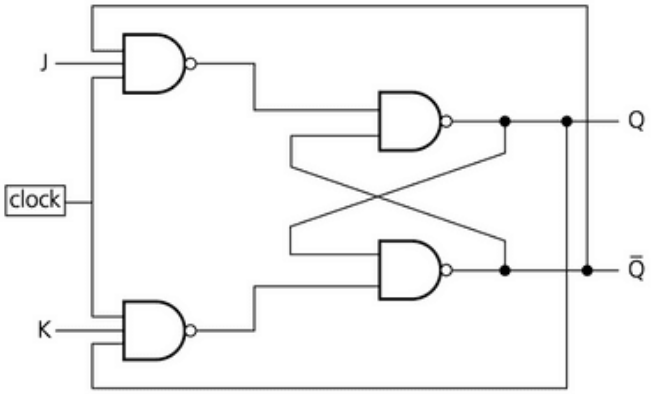
JK Flip-Flops

- improved version of SR flip-flops – eliminates invalid input combinations
- used to produce shift registers in a computer or a simple binary counter\

Advantages

- validates all input combinations
- avoidance of unstable states and increased stability

- Uses a clock pulse as an input (to synchronise inputs) and adds additional gates



When J and K = 0, there is no change to Q.

When J and K = 1, the value of Q toggles after each clock pulse.

(flip-flop changes between SET and RESET state)

** the output value of Q will always be equal to J, NOT Q will be equal to K*

J	K	Value of Q before clock pulse	Value of Q after clock pulse	OUTPUT
0	0	0	0	Q is unchanged after clock pulse
0	0	1	1	
1	0	0	1	Q = 1
1	0	1	1	
0	1	0	0	Q = 0
0	1	1	0	
1	1	0	1	Q value toggles between 0 and 1
1	1	1	0	

Boolean Algebra

- form of algebra linked to logic gates
- based on TRUE and FALSE statements

Boolean Notation

Logic Gate	Notation
AND	A.B
OR	A+B
NOT	
NAND	
NOR	

De Morgan's Laws

Commutative Laws	$A + B = B + A$	$A.B = B.A$
Associative Laws	$A + (B + C) = (A + B) + C$	$A.(B.C) = (A.B).C$
Distributive Laws	$A.(B + C) = (A.B) + (A.C)$ $(A + B).(A + C) = A + B.C$	$A + (B.C) = (A + B).(A + C)$
Tautology/Idempotent Laws	$A.A = A$	$A + A = A$
Tautology/Identity Laws	$1.A = A$	$0 + A = A$
Tautology/Null Laws	$0.A = 0$	$1 + A = 1$
Tautology/Inverse Laws	$A.\overline{A} = 0$	$A + \overline{A} = 1$
Absorption Laws	$A.(A + B) = A$ $A + A.B = A$	$A + (A.B) = A$ $A + \overline{A}.B = A + B$
De Morgan's Laws	$\overline{(A.B)} = \overline{A} + \overline{B}$	$\overline{(A + B)} = \overline{A}.\overline{B}$

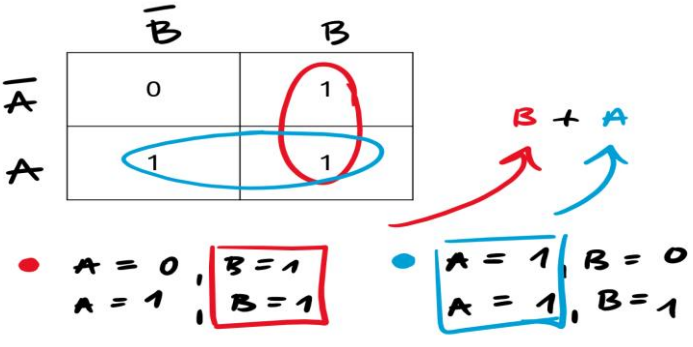
Advantages

- minimises number of boolean expressions
- minimises number of logic gates – provides a more efficient circuit

Rules of Simplification

- works by grouping together adjacent cells containing 1s
  - ➔ groups cannot be diagonal – they can wrap around
  - ➔ only 2^n cells in each group (e.g. only 2/4/8... 1s allowed in one group)
  - ➔ each 1 must be in at least one group – overlapping of groups is allowed
  - ➔ groups should be as large as possible
  - ➔ there should be the fewest amount of groups possible

Example 1:



1. Assume both A and B are TRUE (1)

2. Analyse both groups for unchanging variables

3. Combine the groups

Example 1:

A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

$A.\overline{B}$

$A=1, B=0, C=1$

$A=1, B=0, C=0$

so, $X = A.\overline{B} + A.C$

AB

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

$A.C$

$A=1, B=0, C=1$

$A=1, B=1, C=1$

1. Separate into AB and C
2. Analyse both groups for unchanging variables
3. Combine the groups