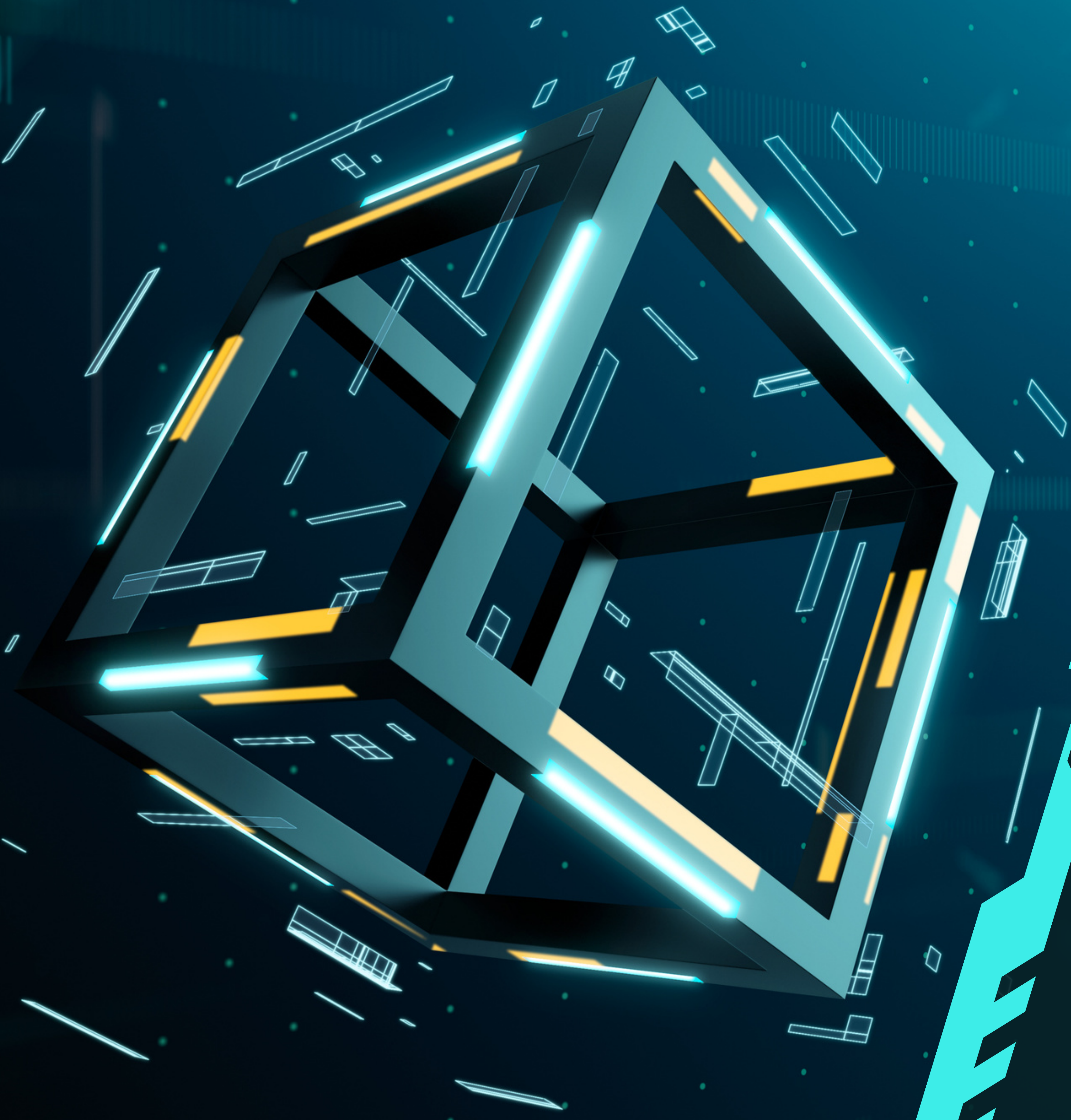


UNDER THE HOOD

MID-TERM EVALUATION

Mentors and Mentees

Aditya Tanwar

Bornadhya Abhir Rajbongshi
Survat Pal
Aastha Sitpal
Sajal Jain
Nelluru Mourya Reddy
Lavesh Gupta

Rajat Gattani
Ramtej Penta
Devang Kumawat
Madhur Bansal
Hisham Hadi T
Dasari Charithambika

Akhil Agrawal

Dhruv
Ujjawal Dubey
Vishant Bhadana
Vaishnavi Singh
Rishi Poonia
Anuj

Table Of Content

Project Timeline

Phase 1

Basics of Binary numbers, Logic Gates, 1s complement, 2s complement, boolean operators, funcition, K-Maps, Making an Adder/Subtractor using gates

Phase 2

Starting with the basics of Verilog and its syntax, we implemented the previous topics in Verilog, like Adder/Subtractor. Used gtkwave for plotting the graph. Learnt about FSM and applied it.

Phase 3

We will start with the five state abstract fsm model of an ISA, and then begin assembly language using MIPS ISA.

Phase 4

In this phase we implement what we learnt throughout the project and make a small application in MIPS as a part of team

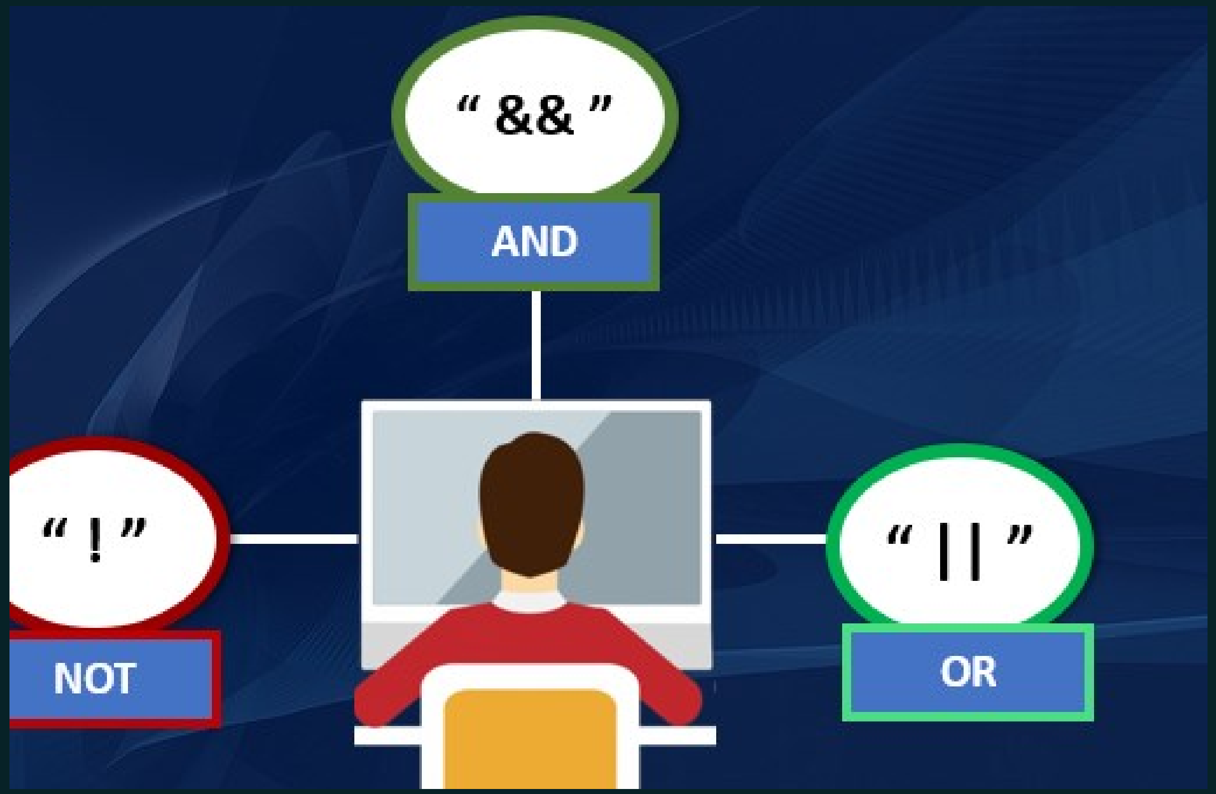
Binary Representation

Representation of Integers

We learnt how to represent the integers including both positive and negative numbers. The main topics that were covered under this heading were:

- Number of Combinations possible using n binary digits
- 1's complement and it's drawbacks and inefficiency
- 2's complement
- Converting positive numbers to their negative counterparts

Boolean Operators

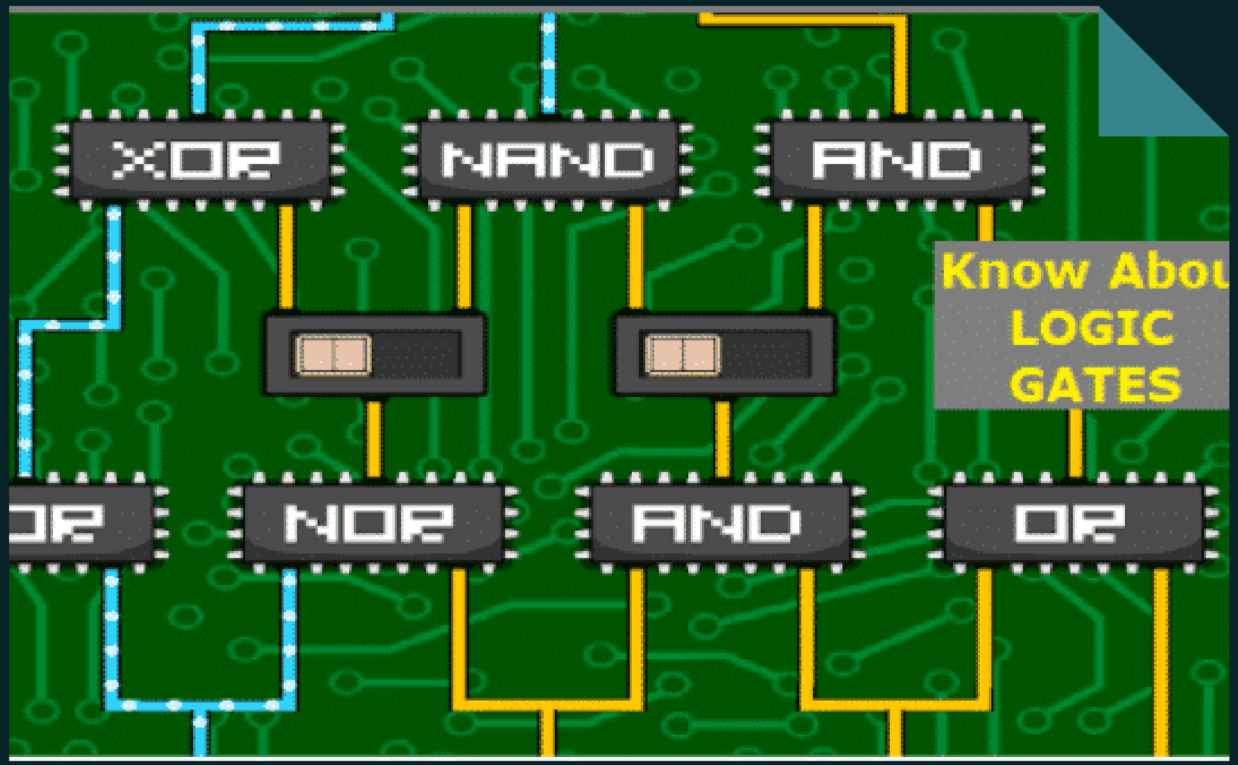


Operators and their properties

Different operators like AND, OR, NOT were taught with their properties and their use cases

- Symbols of Different operators
- Boolean Algebra using these operators
- Making more complex operators using AND, OR and NOT

Logic Gates



Various types of logic gates

A logic gate is a device that acts as a building block for digital circuits Logic gates are based on Boolean algebra. At any given moment, every terminal is in one of the two binary conditions, false or true. False represents 0, and true represents 1

- Three basic logic gates-AND (.), OR(+) , NOT(—)
- Gates-XOR,XNOR. Universal gates-NAND , NOR
- Combination of different logic gates
- Solving different logic gates using boolean algebra

Truth Tables and Functions

Making and understanding Truth Tables

Truth tables make it easier to understand and visualise different Functions made using logic gates and binary operators. We take some input variables and write all the possible combinations and we calculate the outcome of each case and plot it to make a truth table.

- Complex functions using different gates
- Making different logic gates using basic logic gates
- Making Truth Table using the given function
- Making function using the truth table and their relation

K-Maps

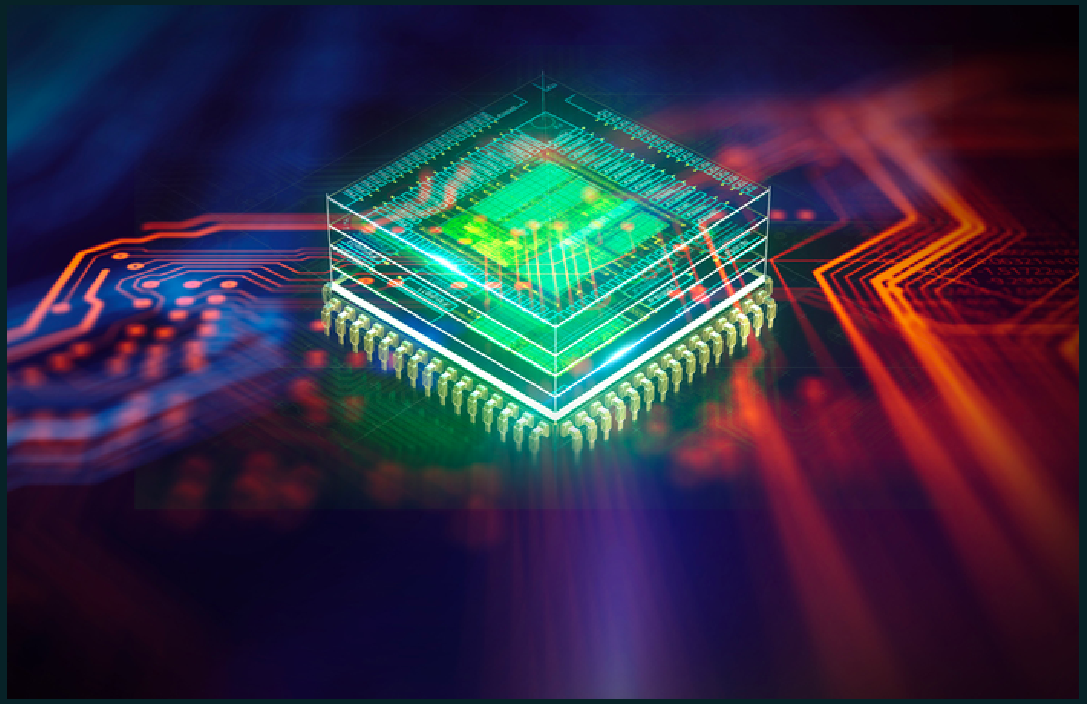
C-D↓ A-B→	00	01	11	10
00	0	0	0	1
01	0	0	0	1
11	1	1	1	1
10	0	0	0	1

K-Maps And its Applications

The Karnaugh map is a method of simplifying Boolean algebra expressions. Through this map we try to minimise the amount of gates used in a function

- We organise our Truth table in a particular way
- If two adjacent values are 1 we look for which bit is changing
- We conclude that the this input has no effect on output
- And this way we can decrease the number of gates

Intro to Verilog

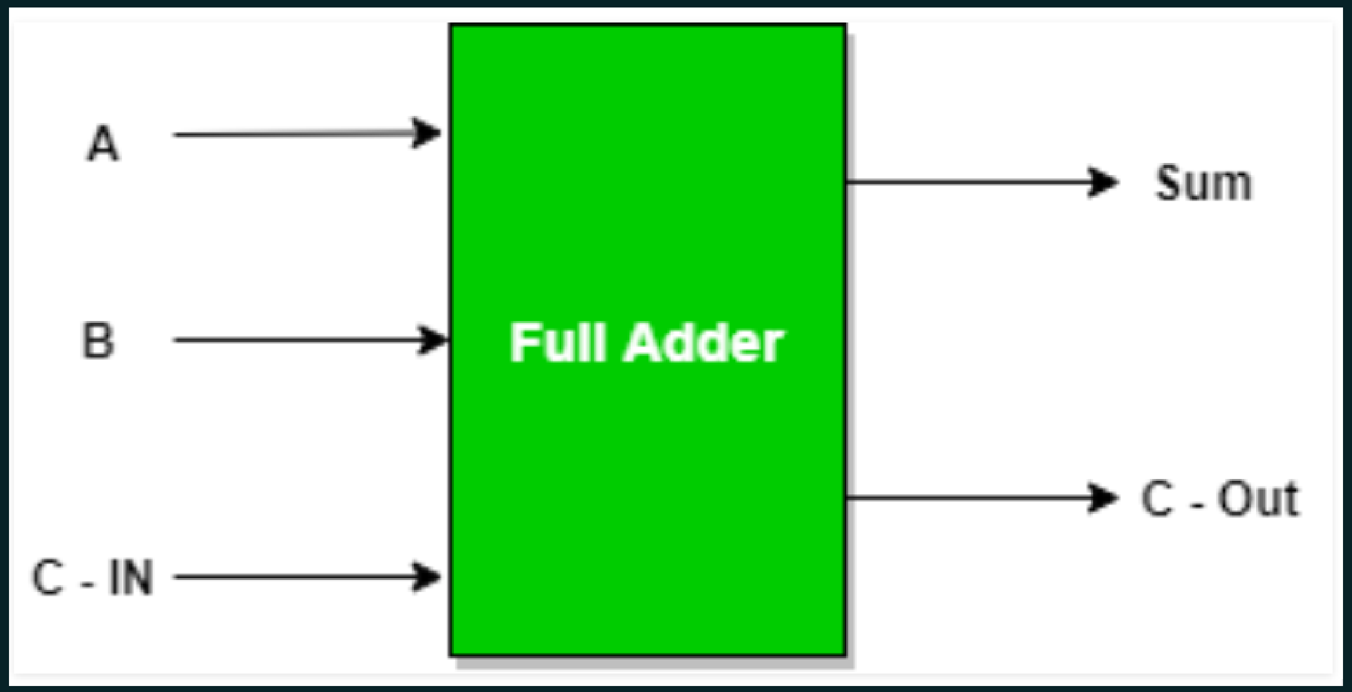


Basics of Verilog

Basic Structure of Verilog was taught using various examples and as we proceeded we went on to learn some new functions like monitor, always, if-else, etc. We also used gtkwave to visualise what really was going on while the code was running

- The variables were of 2 types - reg and wire
- An output of a function should always be stored in a wire
- Always block always initialises when its arguments change
- Monitor displays its arguments when they change
- If-else were similar to C, initial block always runs first in the code

Adder,Subtractor

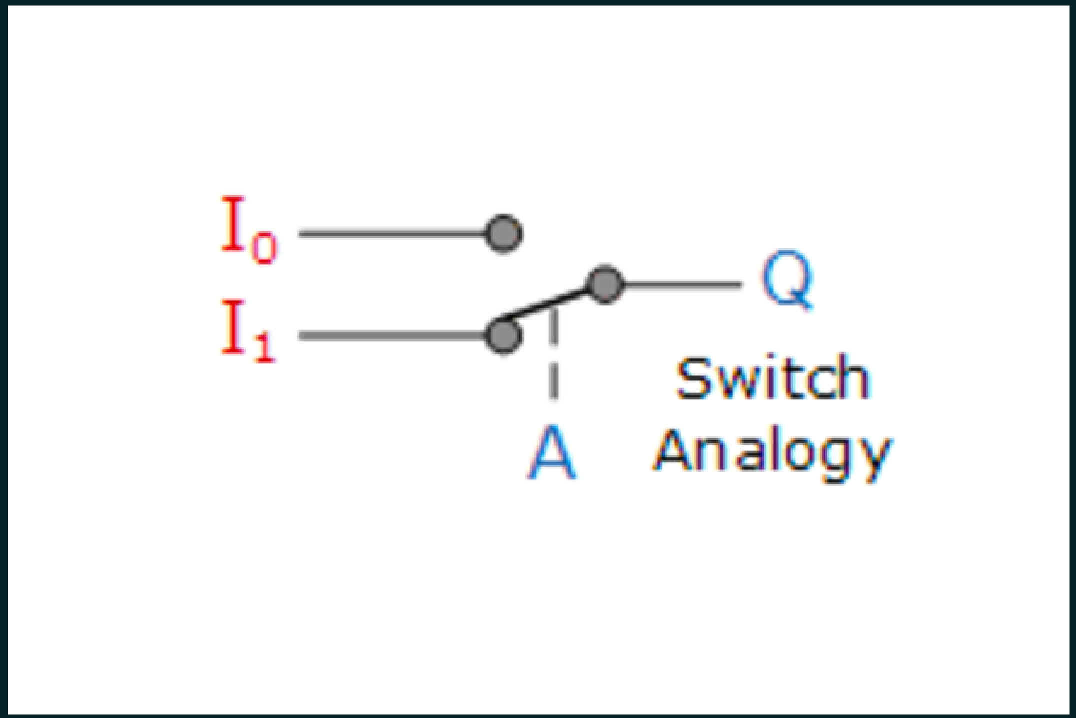


Adding,subtracting using learnt topics

Adds together two binary digits, plus a carry-in digit to produce a sum and carry-out digit. It therefore has three inputs and two outputs.The first two inputs are A and B and the third input is an input carry as C-IN. We implemented this in verilog

- Take eight inputs together to create a byte-wide adder and cascade the carry bit from one adder to another
- Initial carry (C-IN) for adder =0
- C-OUT for one bit will be C-IN for next bits
- Addition of single bits-
  0+0=0, 0+1=1

MUX and Comparator

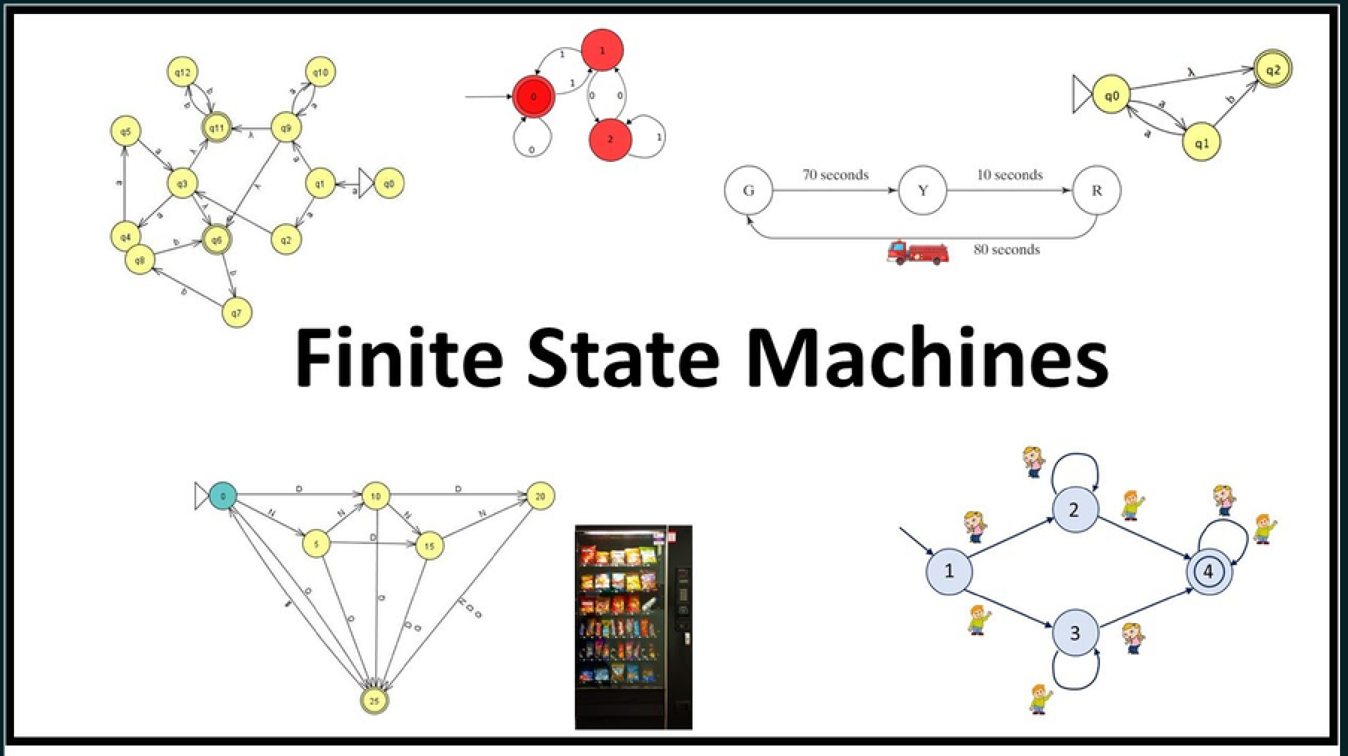


Multiplexer, Comparator and their uses

MUX or Multiplexers are best understood using a switch analogy, given 2 inputs, using Multiplexer, we can take any one input of choice as output. Comparator, as the name suggests, is used to compare two inputs, it will have three outputs and only one of them will be set to 1

- The three outputs in comparator depicts greater, equal, and smaller
- MUX can be used when we output multiple data using same wire

Finite State Machine



FSM-a computational model

A system where particular inputs cause particular changes in state can be represented using finite state machine.some daily life examples-a vending machine, a subway entrance turnstile, sequence detector

- States represent the status of a system
- Have one starting state, which is the first state that is activated upon their execution
- When a finite-state machine is running, one or many states are active in each execution step. Those active states represent the current value of the system.

THANK YOU