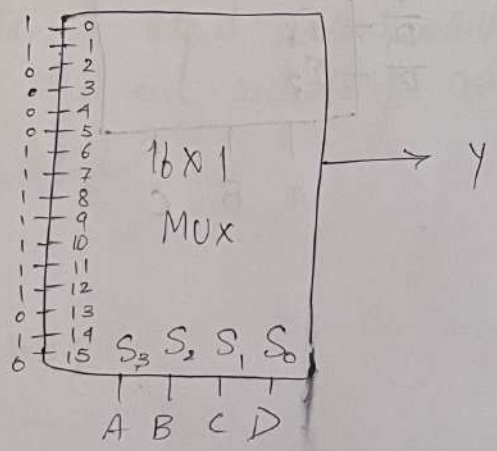
1. Implement $Y(A,B,C,D) = \Sigma m(0,1,6,7,8,9,10,11,12,14)$ using 16-to-1 multiplexer and 8-to-1 multiplexer.

→ 16 X 1 Mux :- $2^4 = 16$ ∴ 4 select lines

$A = S_3$, $B = S_2$, $C = S_1$, $D = S_0$

W.K.T,

	A	B	C	D	Y
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	0
3	0	0	1	1	1
4	0	1	0	0	1
5	0	1	0	1	0
6	0	1	1	0	0
7	0	1	1	1	0
8	1	0	0	0	0
9	1	0	0	1	0
10	1	0	1	0	0
11	1	0	1	1	1
12	1	1	0	0	1
13	1	1	0	1	1
14	1	1	1	0	1
15	1	1	1	1	0

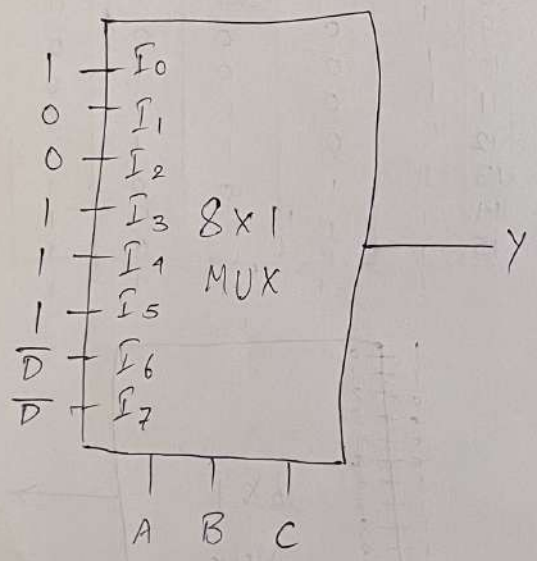


8 × 1 MUX :- $2^3 = 8$ ∴ 3 select lines

A, B, C → Select lines

D → Input

	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
$\overline{D}$	(0)	2	4	(6)	(8)	(10)	(12)	(14)
D	(1)	3	5	(7)	(9)	(11)	13	15
	1	0	0	1	1	1	$\overline{D}$	$\overline{D}$



2. Explain different modelling styles used to write the code in VERILOG with an example.

→ 1. Behavioral modelling :- Describes what the circuit does, not how it is built. The logic is described using an if-else statement inside an always block & it is suitable for algorithmic & control logics.

Ex,
```
module and_gate_behavioral(
  input A,B;
  output reg Y; );
  always @(*) begin
    if(A&&B)
      Y = 1'b1;
    else
      Y = 1'b0;
  end
endmodule
```

2. Dataflow Modelling :- Describes how data flows from inputs to outputs. The output is continuously assigned the results of A & B and suitable for combinational logic.

Ex,
```
module and_gate_dataflow(
  input A,B;
  output Y; );
  assign Y = A&B;
endmodule
```

3. Structural Modelling :- Describes the circuit as an interconnection of components using gate-level modules like AND gate as a primitive gate.

Ex, module and_gate_structural(
input A, B;
output Y;);
and G1 (Y, A, B);
endmodule

3. Find the complement & simplify the boolean function and also write logic circuit $F = A'B'C' + A'BC$.

→ Given boolean function,

$$F = A'B'C' + A'BC$$

$$\Rightarrow F = A'(B'C' + BC)$$

Complement :- $F' = [A'(B'C' + BC)]'$

Apply De Morgan's theorem

$$F' = A + (B'C + BC)'$$

$$\Rightarrow (B'C' + BC)' = (B'C')'(BC)' = (B+C)(B'+C')$$

$\therefore F' = A + (B+C)(B'+C')$

But, $(B+C)(B'+C') = BC' + B'C$

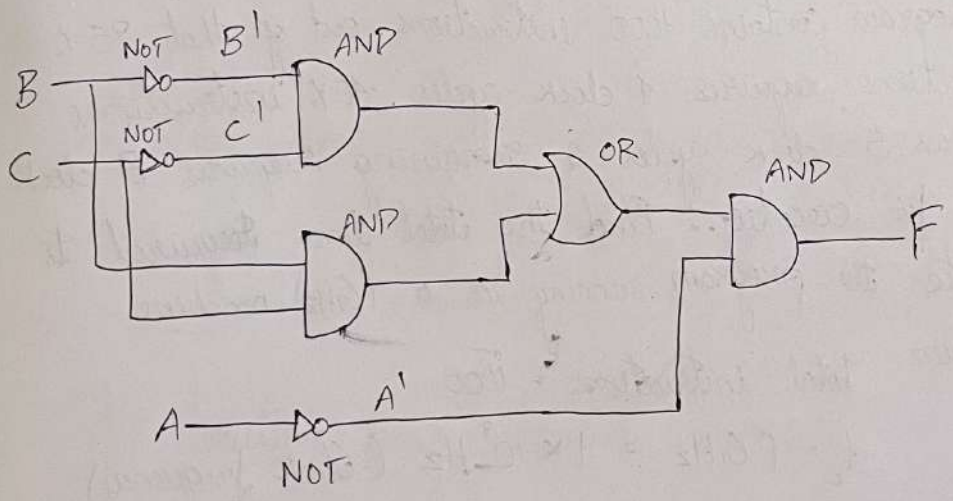
Hence, $F' = A + B'C + BC'$ //

Simplication :- The expression is XNOR of B & C

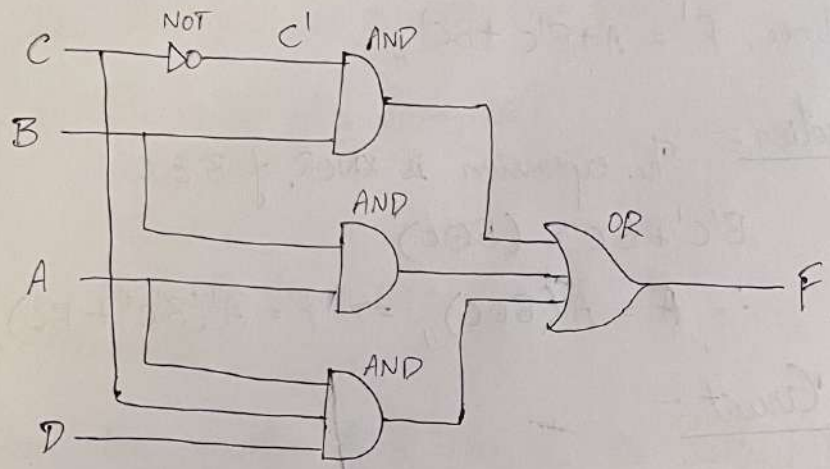
$B'C' + BC = (B \odot C)$

$\therefore F = A'(B \odot C)$ // $\Rightarrow F = A'(B'C' + BC)$ //

Logic Circuit :-



4. Draw a 2-level logic diagram to implement the boolean function $F = BC' + AB + ACD$.



5. A program contains 1000 instructions. out of that 25% instructions requires 4 clock cycles, 40% instructions requires 5 clock cycles & remaining requires 3 clock cycles for execution. Find the total time required to execute the program running in a 1GHz machine.

→ Given, total instructions = 1000

$f_c = 1\ GHz = 1 \times 10^9\ Hz$ (Clock frequency)

$T_c = 1/f = 1/10^9 = 1ns$ // (Clock Period)

* 25% requires 4 cycles :-

0.25 x 1000 = 250 instructions

250 x 4 = 1000

* 40% requires 5 cycles :-

0.40 x 1000 = 400 instructions

400 x 5 = 2000

* Remaining require 3 cycles :-

1000 − (250 + 400) = 350 instructions

350 x 3 = 1050

Total clock cycles = 1000 + 2000 + 1050 = 4050 cycles

Execution Time = Total cycles x Clock period

= 4050 x 1ns

= 4050 ns "

Total execution time = 4050 ns = 1.05 μs "

6. Design a car safety alarm circuit diagram. The system consider 4 inputs :- Door (D), Key (K), Seat pressure (P) and seat belt (B). The inputs is considered HIGH (1) if the door is closed, the key is in, the driver ~~seat~~ is on the seat, or the seat belt is fastened. The alarm (A) should sound with 2 conditions as stated below :- The door is not closed, & the key is in. The door is closed, the key is in the driver's seat & the seat belt is not closed.

(a) Construct a truth table for the system based on input arrangement D, K, P, B with A as an output.

→ Given, D = Door closed, K = Key inserted, P = Driver seat
B = Seat belt fastened, A = Alarm (output)

∴ Truth Table :-

D	K	P	B	A
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

$A = \Sigma m(4,5,6,7,14)$ //

(b) Design a Karnaugh map to verify the simplified expression.

→

DK \ PB	P'B'	PB'	PB	P'B
D'K'	0	1	3	2
DK'	1 (4)	1 (5)	1 (7)	1 (6)
DK	12	13	15	1 (14)
D'K	8	9	11	10

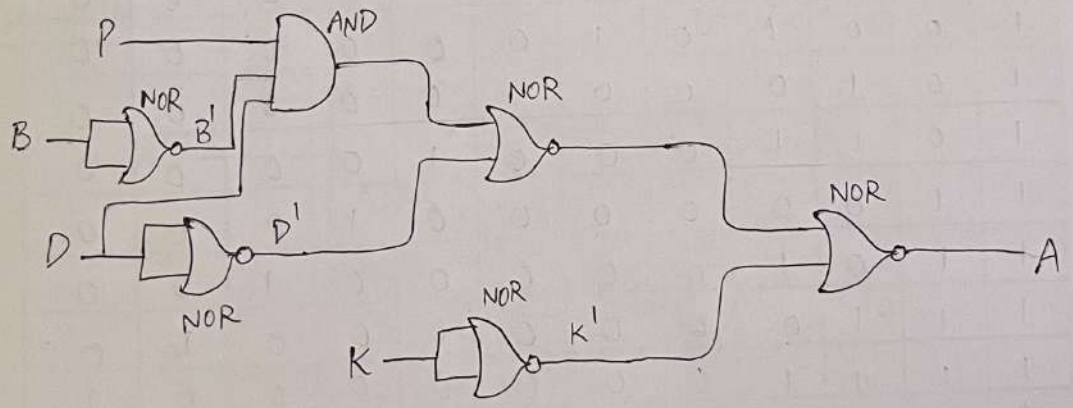
$A = D'K + DKPB'$

$\therefore A = K(D' + DPB')$ //

(c) Draw the simplified circuit using NOR gates only.

→ using De Morgan's Theorem,

$A = [(K') + (D' + DPB')]'$



7. Define decoder. Describe the working principle of a 3:8 decoder. Draw the logic ~~circuit~~ diagram of the 3:8 decoder with enabled inputs. Realize the following boolean expressions using a 3:8 decoder and multi-input OR gates :- $F1(A,B,C) = \Sigma m(1,3,7)$, $F2(A,B,C) = \Sigma m(2,3,5)$.

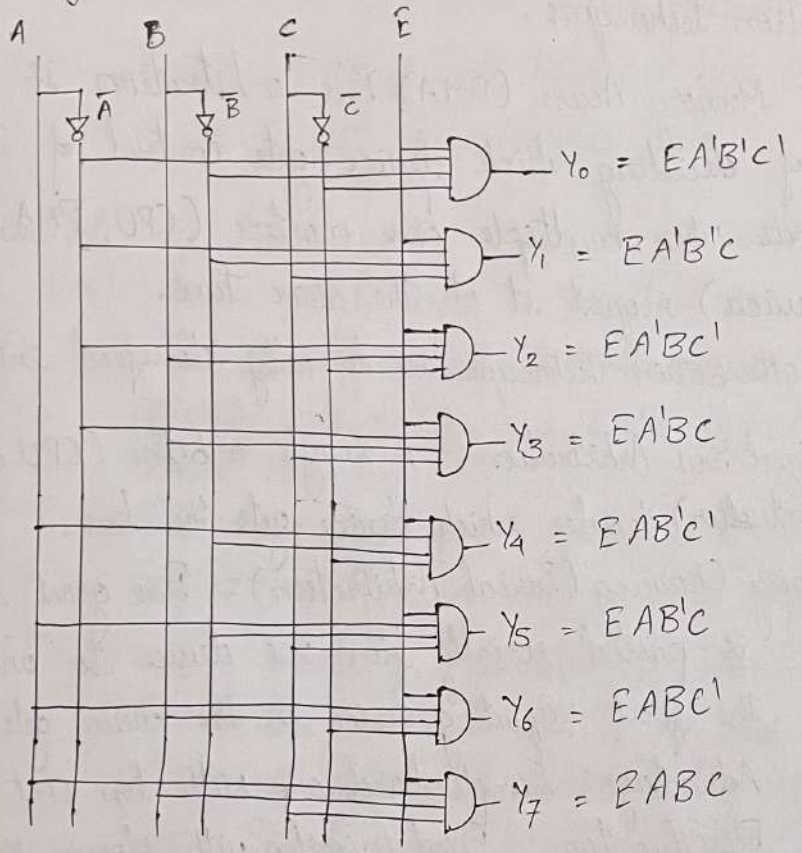
→ A decoder is a combinational logic circuit that converts any n-bit binary inputs into one of 2^n mutually exclusive outputs.

For a given input combination, only one output is active, while all others remain inactive.

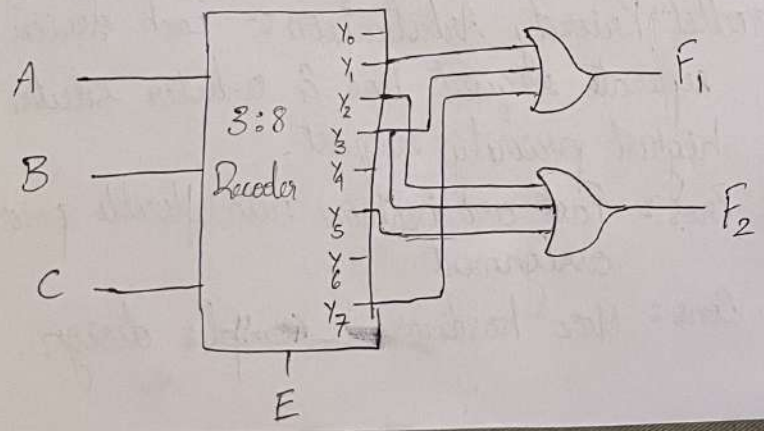
Truth Table of 3:8 decoder (with enabled)

E	A	B	C	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
0	X	X	X	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	0	0	0	0
1	0	0	1	0	1	0	0	0	0	0	0
1	0	1	0	0	0	1	0	0	0	0	0
1	0	1	1	0	0	0	1	0	0	0	0
1	1	0	0	0	0	0	0	1	0	0	0
1	1	0	1	0	0	0	0	0	1	0	0
1	1	1	0	0	0	0	0	0	0	1	0
1	1	1	1	0	0	0	0	0	0	0	1

Logic diagram :-



Realization :- $F_1 = Y_1 + Y_3 + Y_7$, $F_2 = Y_2 + Y_3 + Y_5$ (Given)



8. What is DMA Bus arbitration? Explain different bus arbitration techniques.

→ Direct Memory Access (DMA) Bus arbitration is the process of deciding which device gets control of the system bus when multiple bus masters (CPU, DMA Controller, I/O devices) request it at the same time.

The bus arbitration techniques are broadly classified into :-

* Centralized Bus Arbitration :- A single arbiter (CPU or Bus controller) decides which device gets the bus.

→ Daisy Chaining (Serial Arbitration) :- Bus grant signal is passed serially from one device to another & the first requesting device in the chain gets priority.

Advantage :- Simple hardware with low cost.

Disadvantage :- Fixed priority with slower response for low priority devices.

→ Parallel Priority Arbitration :- Each device has a seperate request line & arbiter selects the highest priority request.

Pros :- Fast arbitration with flexible priority assignment.

Cons :- More hardware & complex design.

→ Rotating Priority (Round Robin) :- Priority rotates after each bus grant which prevents starvation.
Pros:- Fair access with balanced bus stage.
Cons :- Slightly more complex logic

* Distributed Bus Arbitration :- There is no central arbiter. Each device participates in arbitration.
→ Self-Selection Arbitration :- All devices place their priority code on the bus & the highest priority device wins.
Pros :- No single point of failure (Fast Arbitration)
Cons :- More logic in each device & higher cost.
→ Collision-Detection Arbitration :- Devices start transmitting simultaneously so that the collision is detected & devices retry after delay.
Pros :- Simple concept & no arbiter required.
Cons :- Collisions waste time & not ideal for DMA. (real-time).

9. Explain, with an example, the different types of hazards that occur during pipelining.

→ There are 3 types of hazards :-

* Structural Hazards :- This occurs when 2 or more instructions require the same hardware resource at the same time, but the hardware cannot support all requests simultaneously.

Ex, I1 : LOAD R1, 0(R2)
I2 : ADD R3, R1, R4

* Data Hazards :- This occurs when an instruction depends on the result of a previous instruction that has not yet completed execution.

Ex, (a) RAW (Read after write) :- I1 : ADD R1, R2, R3
I2 : SUB R4, R1, R5

(b) WAR (Write after read) :- I1 : SUB R3, R1, R2
I2 : ADD R1, R4, R5

(c) WAW (Write after write) :- I1 : MUL R1, R2, R3
I2 : ADD R1, R4, R5

* Control Hazards :- This occurs due to branch or jump instructions, where the next instruction depends on the outcome of a branch.

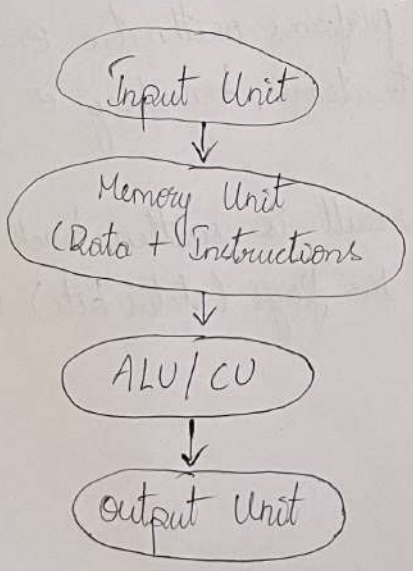
Ex, I1 : BEQ R1, R2, LABEL
I2 : ADD R3, R4, R5

10. Explain the basic operation concepts of the computer with a neat diagram.

→ A Computer performs it's operations based on the stored-program concepts, where both data & instructions are stored in memory and the processor executes instructions one by one.

Main functional units of a Computer :-

* Input Unit
* Memory Unit
* Central Processing unit (CPU) :- Arithmetic Logic Unit (ALU)
  Control Unit (CU)
  Registers.
* output Unit
* System Buses (Data, Address).



There are 5 basic operations :- (Instruction Execution Cycle)

* Instruction Fetch :- The Program Counter (PC) holds the address of the next instruction. This address is sent to memory where the instruction is fetched & stored in the Instruction Register (IR). PC is then incremented to the next instruction.
* Instruction Decode :- The Control Unit (CU) decodes the instructions in IR. It identifies the operation to be performed, operands required & addressing mode.
* Operand Fetch :- If required, operands are fetched from registers, memory & input devices.
* Execute :- The ALU performs arithmetic or logical operations. For control instructions, branching or jumping is performed.
* Write back :- The result is written back to register & memory where the flags (status bits) are updated.