ASSIGNMENT 5

.

NAME = BHAVYAM VERMA

ENROLLMENT NO = S24CSEU1157

Q1. What are Combinational and Sequential circuits? Discuss in brief with appropriate block diagram and example.

Solution:

Page No.

Ans-1 • Combinational Circuits

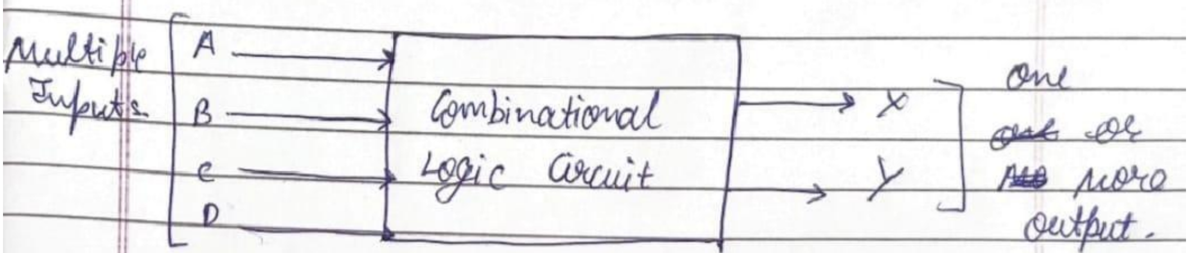
→ A type of digital Circuit where output depends only on the present input values. It does not have memory elements.

• Sequential Circuits

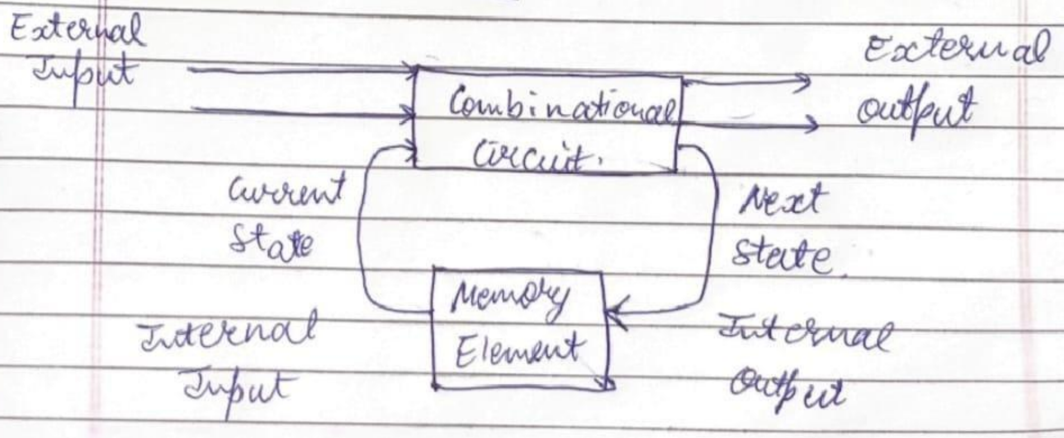
A Sequentical Circuit is a type of Circuit where output depends ~~only~~ upon present & also past inputs. It requires memory elements.

Block Diagram

1. Combinational Circuits



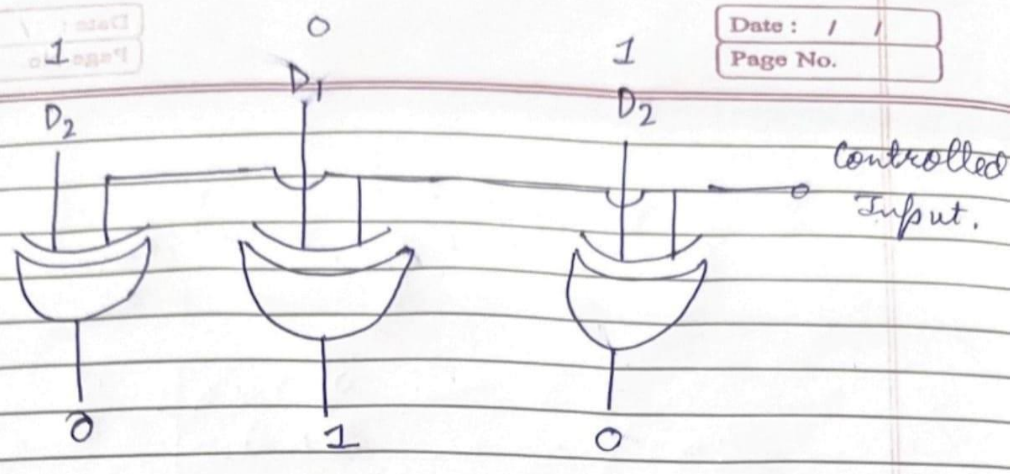
2. Sequential Circuits



Q2. Draw and explain 4-bit Controlled Inverter/Buffer using Ex-OR gates.

solution:

Ans-2



A 4-bit Controlled Inverter/Buffer using XOR gates can be implemented by connecting each bit of the 4-bit input to one input of an XOR gate.

1. **Understanding XOR Gate**: The XOR (exclusive OR) gate gives a true (1) output when the number of true inputs is odd. In binary terms, if both inputs are the same, the output is 0; if different, the output is 1.
2. **4-bit Controlled Inverter/Buffer**: This circuit can either invert or buffer the input signal based on a control input. Using XOR gates, this operation can be achieved efficiently. The control input decides whether the input bits should be passed directly (buffered) or inverted (inverted buffer).

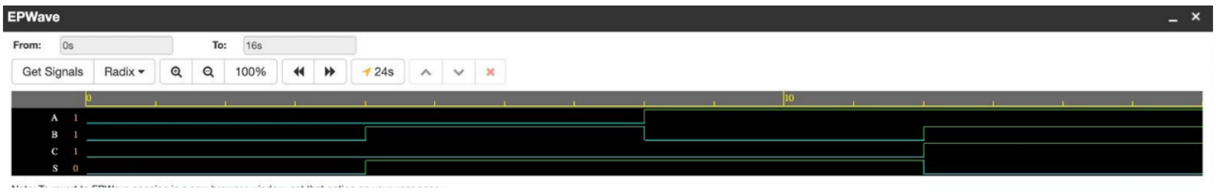
3. Write truth table and Verilog code to implement Half Adder using only NAND gates.

.Solution

:

A	B	SUM	CARRY
0	0	0	1
0	1	1	1
1	0	1	1
1	1	0	0

```
Verilog code-
module half_adder( input a, b, output s, c);
assign s = (~a&&b)||(a&&~b);
assign c = (a&&b);
 endmodule
 testbench code-
 module tb_half_adder;
reg A, B; wire S,C ;
half_adder a1(.a(A),.b(B),.s(S),.c(C));
initial begin
 A=0;B=0;#4;
 A=0;B=1;#4;
A=1;B=0;#4;
 A=1;B=1;#4;

end
initial begin
$dumpfile("dump.vcd");
$dumpvars(1);
 end
endmodule
```

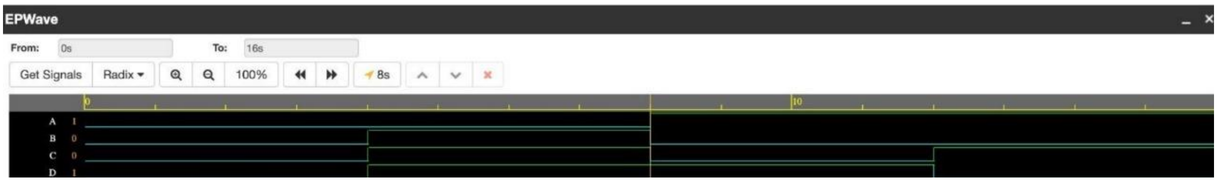


4. Write truth table and Verilog code to implement Full Adder using only NOR gates

A	**B**	**Cin**	**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

```
module full_adder_nor (
   input A, B, Cin,
   output Sum, Cout
);

   wire nota = ~(A | A);
   wire notb = ~(B | B);
   wire notcin = ~(Cin | Cin);

   wire A_nor_B = ~(A | B);
   wire t1 = ~(A | A_nor_B);
   wire t2 = ~(B | A_nor_B);
   wire AxorB = ~(t1 | t2);  // A ⊕ B


   wire AxorB_nor_Cin = ~(AxorB | Cin);
   wire t3 = ~(AxorB | AxorB_nor_Cin);
   wire t4 = ~(Cin | AxorB_nor_Cin);
   assign Sum = ~(t3 | t4);  // Final SUM = A ⊕ B ⊕ Cin


   wire ab_nor = ~(A | B);
```

```
    wire ab_and = ~(ab_nor | ab_nor); // A & B

    wire bc_nor = ~(B | Cin);
    wire bc_and = ~(bc_nor | bc_nor); // B & Cin

    wire ac_nor = ~(A | Cin);
    wire ac_and = ~(ac_nor | ac_nor); // A & Cin

    wire ab_bc_nor = ~(ab_and | bc_and);
    wire ab_or_bc = ~(ab_bc_nor | ab_bc_nor); // A&B | B&Cin

    wire final_nor = ~(ab_or_bc | ac_and);
    assign Cout = ~(final_nor | final_nor); // Final Cout

endmodule
```



5. Write truth table and Verilog code to implement Half Subtractor using XOR and AND gates .

Design:

```
module half_adder(
   input a, b,
   output sum, carry
);
   assign sum = a ^ b;  // XOR gate for sum
   assign carry = a & b; // AND gate for carry
endmodule
```

Testbench:

```
module tb_half_adder;
   reg A, B;
   wire sum, carry;
half_adder ha1 (   .a(A), .b(B), .sum(sum), .carry(carry));

   initial begin
      $monitor("At time %0t: A=%b, B=%b | Sum=%b, Carry=%b", $time, A, B, sum, carry);
   end
   initial begin
      A = 0; B = 0; #5;
      A = 0; B = 1; #5;
      A = 1; B = 0; #5;
      A = 1; B = 1; #5;
   end
   initial begin
      $dumpfile("dump.vcd");
      $dumpvars(1);
   end
endmodule
```