Name:	Nandana P
Register No.:	727721EUEC096
Subject Code and Subject Title:	21EC504- VLSI Design
Internal Component:	Simulation using EDA Playground
Date:	11th October, 2023

1. Simulate the following gates and combinational circuits using EDA Playground:
   i) XOR
   ii) XNOR
   iii) 8 x 1 MUX
   iv) 5 input NAND

a) XOR

Testbench code:

```
module myxor_tb;
 reg a1,b1;
 wire y1;


 myxor myxor_tb(.a(a1),.b(b1),.y(y1));
 initial begin
  $dumpfile("dump.vcd");
    $dumpvars(1);


//   $monitor(a1,b1,y1);
  a1=1'b0;
  b1=1'b0;
 #1 $display("a1:%b, b1:%b, y1:%b",a1,b1,y1);
  #1
  a1=1'b0;
  b1=1'b1;
   #1 $display("a1:%b, b1:%b, y1:%b",a1,b1,y1);
  #1
  a1=1'b1;
  b1=1'b0;
```

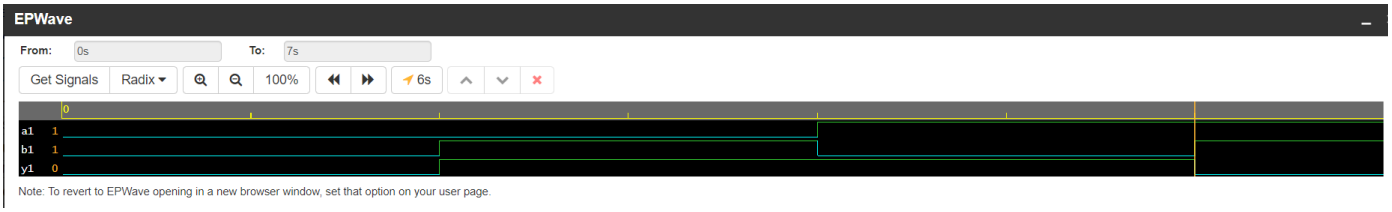
```
    #1 $display("a1:%b, b1:%b, y1:%b",a1,b1,y1);
    #1
    a1=1'b1;
    b1=1'b1;
    #1 $display("a1:%b, b1:%b, y1:%b",a1,b1,y1);
  end
endmodule
```

VHDL code:

```
module myxor (a,b,y);
  input a,b;
  output y;

  assign y = a^b; //dataflow
endmodule
```

Simulation:

```
[2023-09-24 13:40:54 UTC] iverilog '-Wall' design.sv testbench.sv  && unbuffer vvp a.out
VCD info: dumpfile dump.vcd opened for output.
a1:0, b1:0, y1:0
a1:0, b1:1, y1:1
a1:1, b1:0, y1:1
a1:1, b1:1, y1:0
Finding VCD file...
./dump.vcd
```



b) XNOR

Testbench code:

```
module XNOR_Gate_tb;
 reg A;
 reg B;
 wire Y;
 integer i;

 XNOR_Gate inst(.A(A), .B(B), .Y(Y));
 initial begin
  $dumpfile("dump.vcd");
  $dumpvars;
  #100 $finish;
 end
 initial
  begin
   A <= 0;
   B <= 0;
   $monitor ("A=%0b B=%0b Y=%0b", A, B, Y);
   for (i=0; i< 4; i= i + 1)
    begin
     {A, B} = i;
   #10;
    end
  end
endmodule
```

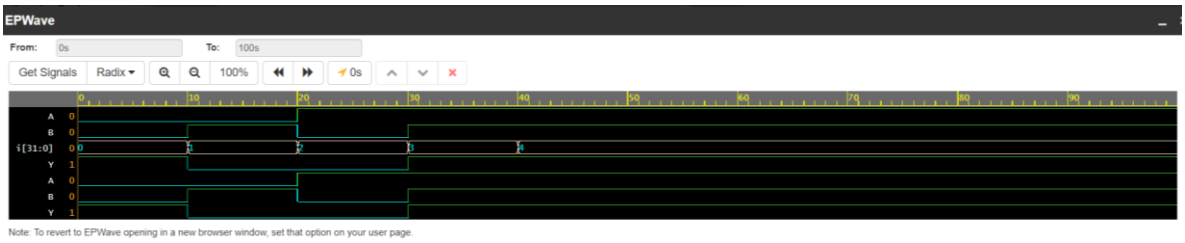
Verilog code:

```
module XNOR_Gate(
 input A,
 input B,
 output Y);
  assign Y = ~ (~A&&B || A&&~B);
endmodule
```

Simulation:

Log Share

```
[2023-09-24 16:46:44 UTC] iverilog '-Wall' design.sv testbench.sv  && unbuffer vvp a.out
VCD info: dumpfile dump.vcd opened for output.
A=0 B=0 Y=1
A=0 B=1 Y=0
A=1 B=0 Y=0
A=1 B=1 Y=1
Done
```



c) 8 x 1 Multiplexer

Testbench code:

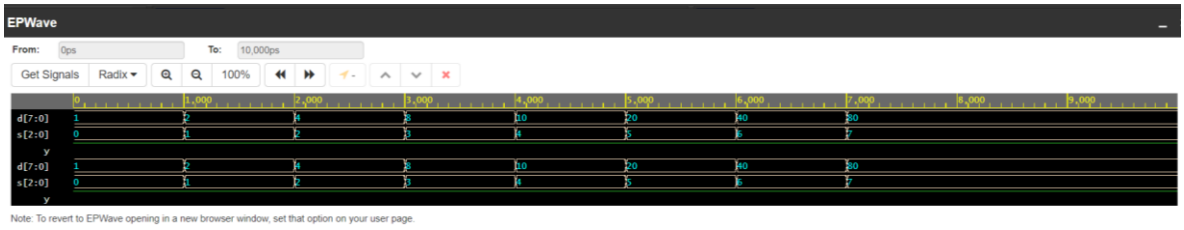
```
-- Testbench for 8 to 1 Multiplexer
entity testbench is
-- empty
end testbench;
architecture tb of testbench is
-- DUT component
component mux8to1 is
port
( s : in bit_vector (2 downto 0);
d : in bit_vector (7 downto 0);
y : out bit);
end component;
signal s : bit_vector (2 downto 0);
signal d : bit_vector (7 downto 0);
signal y : bit;
begin
  -- Connect DUT
  DUT: mux8to1 port map(
```

```
 s => s,
 d => d,
 y => y );
 process
 begin
  s <= "000";
  d <= "00000001";
  wait for 1 ns;
  s <= "001";
  d <= "00000010";
  wait for 1 ns;
  s <= "010";
  d <= "00000100";
  wait for 1 ns;
  s <= "011";
  d <= "00001000";
  wait for 1 ns;
  s <= "100";
  d <= "00010000";
  wait for 1 ns;
  s <= "101";
  d <= "00100000";
  wait for 1 ns;
  s <= "110";
  d <= "01000000";
  wait for 1 ns;
  s <= "111";
  d <= "10000000";
  wait for 1 ns;
  wait;
 end process;
end tb;
```

Verilog code:

```
-- Simple 8 to 1 Multiplexer design


entity mux8to1 is
port ( s : in bit_vector (2 downto 0);
d : in bit_vector (7 downto 0);
y : out bit);
end mux8to1;


architecture equation of mux8to1 is
begin
with s select
y <= d(0) when "000",
d(1) when "001",
d(2) when "010",
d(3) when "011",
d(4) when "100",
d(5) when "101",
d(6) when "110",
d(7) when others;
end equation;
```

Simulation:



d) 5 INPUT NAND

Testbench code:

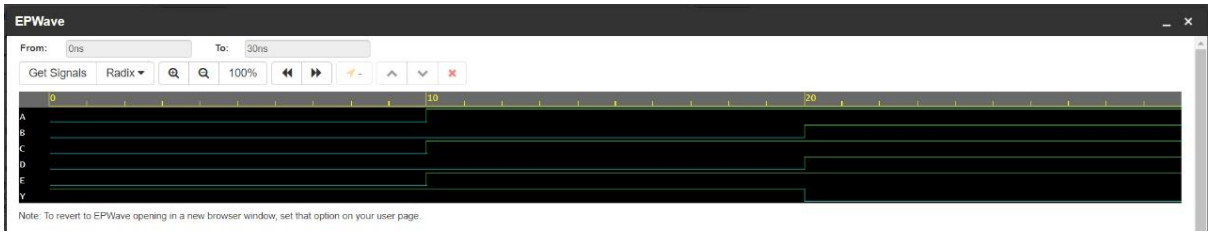
```
module testbench;

reg A, B, C, D, E;
wire Y;

// Instantiate the 5-Input NAND gate
```

```
nand5 uut (
    .A(A),
    .B(B),
    .C(C),
    .D(D),
    .E(E),
    .Y(Y)
);

// Add $dumpfile and $dumpvars
initial begin
    $dumpfile("dump.vcd"); // Specify the VCD file name
    $dumpvars; // Dump all the signals

    $display("A\tB\tC\tD\tE\t|\tY");
    $display("------------------------|---");

    A = 0; B = 0; C = 0; D = 0; E = 0;
    #10; // Wait for a moment
    A = 1; B = 0; C = 1; D = 0; E = 1;
    #10; // Wait for a moment
    A = 1; B = 1; C = 1; D = 1; E = 1;
    #10; // Wait for a moment
    $finish; // End the simulation
end

endmodule
```

Verilog code:

```
// Code your design here

module nand5 (

    input A,

    input B,

    input C,

    input D,

    input E,

    output Y

);


assign Y = ~(A & B & C & D & E);

endmodule
```

Simulation:


EPWave
From:
0ns
To:
30ns
Get Signals
Radix
100%
0
10
20
A
B
C
D
E
Y
Note: To revert to EPWave opening in a new browser window, set that option on your user page.