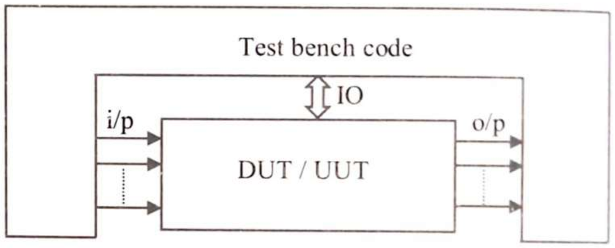
Unit-5 Verilog Program -Combinational and Sequential

03 September 2022 13:26

Verilog Program

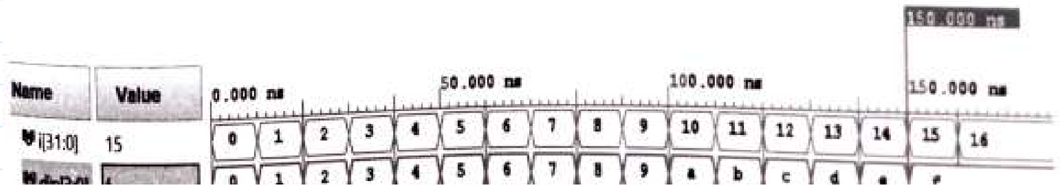


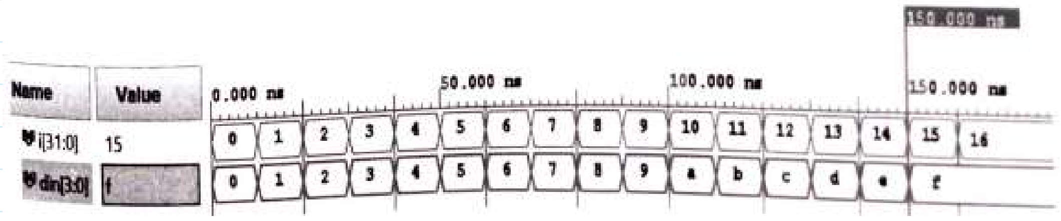
Understand the meaning of following code.

```
module test_wave();
reg A, B;

initial
begin                  //Start of initial block
A = 1'b0;              // both a and b are initialised at 0th time unit
B = 1'b1;
#5    A = 1'b1;        //A changes at 5th time units
#8    B = 1'b0;        //B changes at 5 + 8 = 13th time unit
#10{B, A} = 2'b10;     //again both B & A changes at 5 + 8 + 3 = 23rd time
                         unit. See the use of concatenation {}.
end                    //End of initial block
endmodule
```

Use of 'for' loop to generate a test pattern.

```
module for_loop();
reg [3:0] din;

integer i;

initial
begin
     for (i=0; i<= 15; i = i+1)
     begin                        //multiple statements must be
                                    encapsulated within begin…end
     din = i;
     #10;                         //increment 'i' after the delay of 10
                                    time units
     end                          //end of the for loop
end

endmodule
```



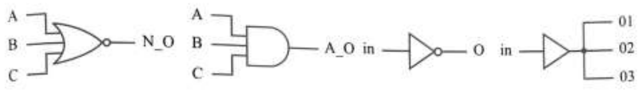


Module Instantiation
There are two types of instantiation constructs in Verilog
Positional Instantiation:
child_modu1e_name [optiona1_instance_name] (comma_seperated_list_of the signals) ;
Child module name is the name of the black box to be instantiated.

Named Instantiation: Both child- and parent- module port names are used in this method of instantiation. Therefore, the order of ports is immaterial.

child_module_name [optional_instance_name] (.chi1d_port 1(parent_port 1),
.chi1d_port_2 (parent _port_2) , ...) ;

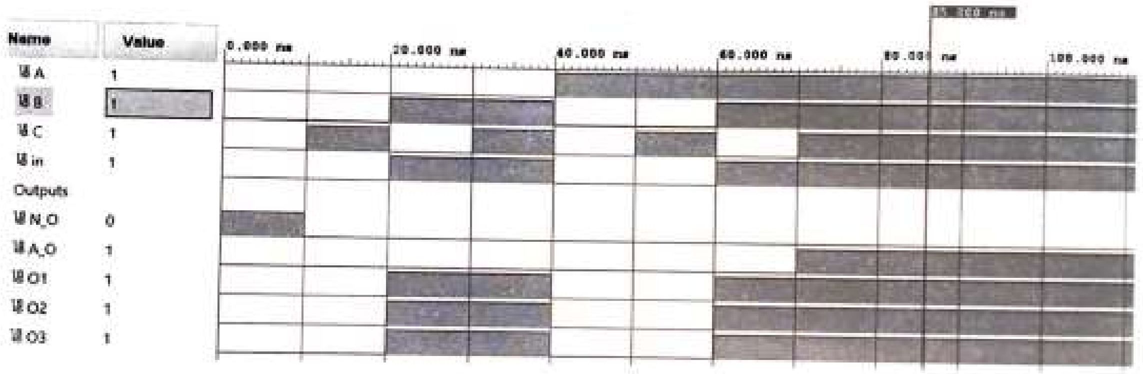
Q.1. Write a structural Verilog to implement gets Given in Figure Also write Test bench to test of the behavior of the gates.



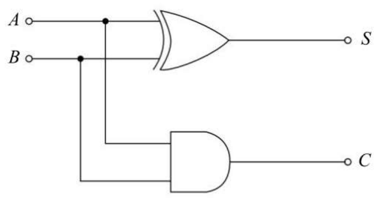
```
module basic_gates (A, B.C ,in, N_O, A_O, O, 01, 02, 03); */ include all
input and output terminals /*

input A, B, C, in;

output N_O, A_O,O,01,02,03;

nor (N_O, A,B,C);      //first terminal is always output than write inputs

and (A_O, A, B,C);

not (O,in) ;

buf (01, 02, 03,in);

Endmodule
```

```
module basic_gates_tb();

reg A, B, C, in;
wire N_O, A_O, O1, O2, O3;

basic_gates u (A, B, C, in, N_O, A_O, O1, O2, O3);  //DUT is instantiated using
                                                      positional instantiation

initial                                  //Applying stimuli to 3 input gates
begin
    {A, B, C} = 3'b000;                  //Use of concatenation {} operator.
#10 {A, B, C} = 3'b001;
#10 {A, B, C} = 3'b010;
#10 {A, B, C} = 3'b011;
#10 {A, B, C} = 3'b100;
#10 {A, B, C} = 3'b101;
#10 {A, B, C} = 3'b110;
#10 {A, B, C} = 3'b111;
end


  initial                                        //Applying stimuli to 1 input gates
  begin
        in = 1'b0;
  #20   in = 1'b1;
  #20   in = 1'b0;
  #20   in = 1'b1;
  end

  endmodule
```

Simulation webform

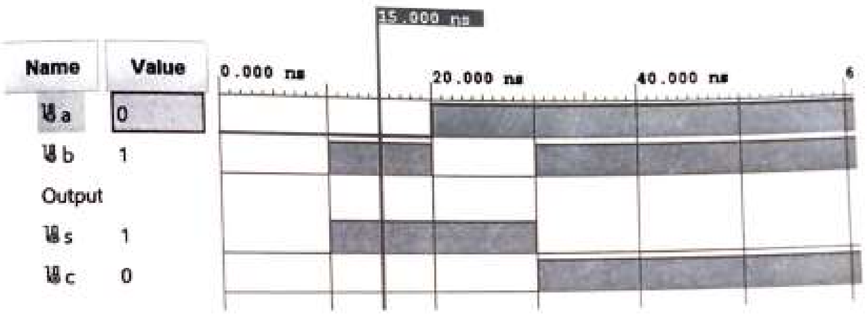


Q.2. Write a structural Verilog code to implement half adder circuit shown in Fig. write the test bench to verify the functionality of design.

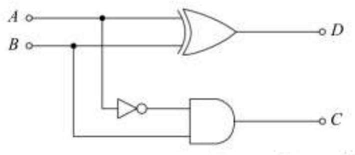


```
module half_adder (A, B, S, C);
input A, B;
output S, C;

xor (S, A, B);
and (C, A, B);

endmodule

module half_adder_tb();

reg a, b;
wire s, c;

half_adder u (.A(a), .B(b), .S(s), .C(c));   //DUT is instantiated using named
                                               instantiation


initial
begin
 {a,b} = 2'd0;       //use of decimal numbers
#10 {a,b} = 2'd1;
#10 {a,b} = 2'd2;
#10 {a,b} = 2'd3;
end

endmodule
```

Waveform



Q.3. Write a structural Verilog code to implement half subtractor circuit shown in figure. Also write test bench.



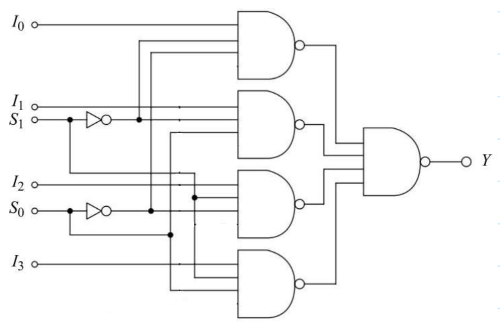
```
module half_subtractor (A, B, D, C);
input A, B;
output D, C;
```

```
module half_subtractor_tb();

reg a, b;
wire d, c;

half_subtractor u (.A(a), .B(b), .D(d), .C(c));    //DUT is instantiated using
                                                     named instantiation


initial
begin
```

```
module half_subtractor (A, B, D, C);
input A, B;
output D, C;

xor (D, A, B);
not (A_bar, A);
and (C, A_bar, B);

endmodule
```

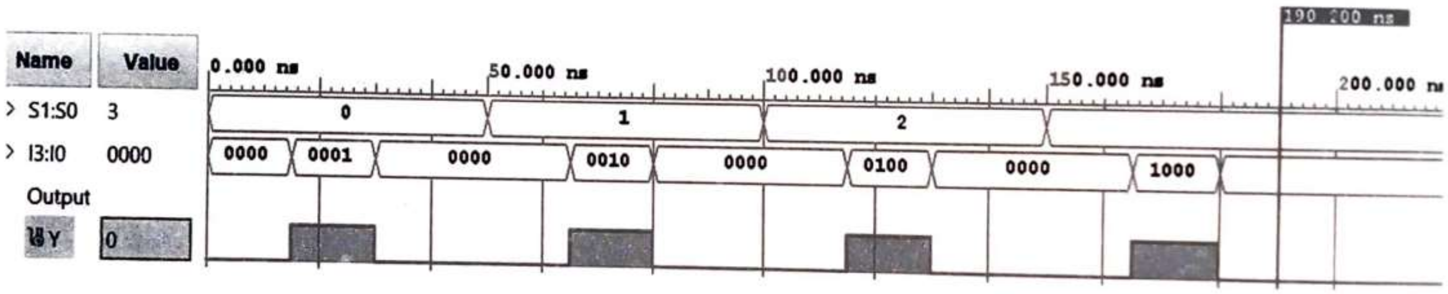
```
subtractor u (.A(a), [illegible]                    [illegible] using
                                              named instantiation

initial
begin
    {a,b} = 2'd0;  //use of decimal numbers


#10  {a,b} = 2'd1;
#10  {a,b} = 2'd2;
#10  {a,b} = 2'd3;
end

endmodule
```

Q.4.Write a structural Verilog code to implement 4:1 multiplexer circuit. Also write test bench code.



```
module mux_4_1 (S0, S1, I0, I1, I2, I3, Y);
input S0, S1, I0, I1, I2, I3;
output Y;

not (s1_bar, S1);
not (s0_bar, S0);
nand (y0, I0, s1_bar, s0_bar);
nand (y1, I1, s1_bar, S0);
nand (y2, I2, S1, s0_bar);
nand (y3, I3, S1, S0);
nand (Y, y0, y1, y2, y3);

endmodule
```

```
module mux_4_1_tb ();
reg S0, S1, I0, I1, I2, I3;
wire Y;
mux_4_1 u (S0, S1, I0, I1, I2, I3, Y);

initial
begin
{S1, S0} = 2'd0;
{I3, I2, I1, I0} = 4'h0;


#15 I0 = ~I0;
#15 I0 = ~I0;

#20{S1, S0} = 2'd1;
#15 I1 = ~I1;
#15 I1 = ~I1;

#20{S1, S0} = 2'd2;
#15 I2 = ~I2;
#15 I2 = ~I2;

#20{S1, S0} = 2'd3;
#15 I3 = ~I3;
#15 I3 = ~I3;
end
endmodule
```



/

02 September 2022
21:48

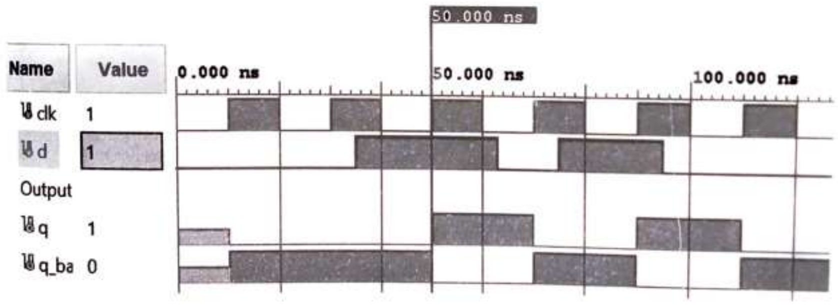
Behavioral (Algorithmic) Coding Examples

Write behavioral Verilog code to implement positive edge triggered D flip-flop.

```
module d_FF (input clk, d, output reg q, output q_bar);

assign q_bar = !q;

always @(posedge clk)
q <= d;

endmodule
```

Test bench

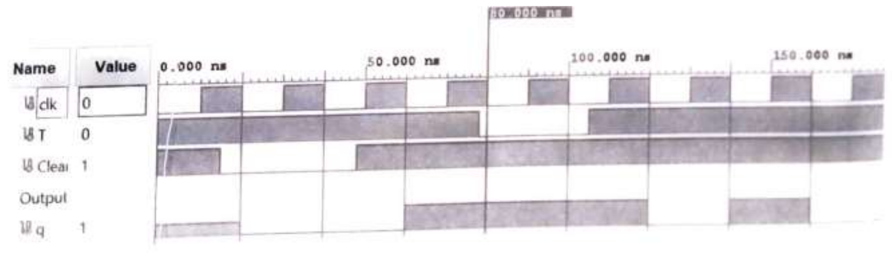
```
module d_FF_tb ();

reg clk, d;
wire q, q_bar;

d_FF u (clk, d, q, q_bar);

always
#10 clk = ! clk;

initial
begin
   clk = 1'b0;
   d = 1'b0;
#35 d = !d;
#28 d = !d;
#12 d = !d;
#20 d = !d;
end
endmodule
```



write behavioral code to implement negative edge triggered T Flip flop with synchronous active low clear input.

```
module T_ff(input clk, T, Clear, output reg q);

always @ (negedge clk)
begin
     if (~Clear)
            q = 1'b0;
     else if (T)
            q = ~q;
end
endmodule
```

```
module T_ff_tb();

reg clk, T, Clear;
wire q;

T_ff u (clk, T, Clear, q);

always
#10 clk = ! clk;

initial
begin
     clk = 1'b0;
     Clear = 1'b1;
     T = 1'b1;
#15 Clear = 1'b0;
#33 Clear = 1'b1;
#30 T = !T;
#27 T = !T;
end
endmodule
```


80.000 ns
Name
Value
0.000 ns
50.000 ns
100.000 ns
150.000 ns
clk 0
T 0
Clear 1
Output
q 1