DAY 34 -111 DAYS VERIFICATION CHALLENGE

Topic: Basic Verilog codes

Skill: Verilog

DAY 34 CHALLENGE:

1. Explain with example different methods to code a clock in Verilog.

1.
```
initial
      Begin
      Clk = 0 ;
      end
always #5 clk = ~clk ;
```

2.
```
module clock_divider(
    input clk_in,
    output reg clk_out
);
    reg [3:0] count = 0;
    always @(posedge clk_in) begin
        count <= count + 1;
        if (count == 4) begin
            clk_out <= ~clk_out;
            count <= 0;
        end
    end
endmodule
```

2. Write a Verilog code for:

I. synchronous reset
```
module synchronous_reset(
      input clk ,
      input rst  );
always @(posedge clk)
begin
      if(rst)begin
```

```
            // give value zero to all variable
      end
      else begin
            // perform any operation
      end
end
endmodule
```

II. Asynchronous reset

```
module synchronous_reset(
      input clk ,
      input rst  );
always @(posedge clk or posedge rst)
begin
      if(rst)begin
            // give value zero to all variable
      end
      else begin
            // perform any operation
      end
end
endmodule
```

3. Write a Verilog code to swap contents of two registers:

I. with a temporary register

```
module swaping (
      input clk ,
      input rst,
      input [7:0]a_in,
      input [7:0]b_in,
      output reg [7:0]a_out,
      output reg [7:0]b_out
        );

Reg [7:0] temp ;

always @(posedge clk)
begin
      if(rst)begin
            a_out <= 0;
            b_out <= 0;
      else begin
            temp <= a_out ;
            a_out <= b_out ;
            b_out <= temp ;
      end
end
endmodule
```

II. without a temporary register

```
   module swaping (
      input clk ,
      input rst,
      input [7:0]a_in,
      input [7:0]b_in,
```

```
        output reg [7:0]a_out,
        output reg [7:0]b_out
          );

    Reg [7:0] temp ;

    always @(posedge clk)
    begin
        if(rst)begin
            a_out <= 0;
            b_out <= 0;
        else begin
            a_out <= a_out ^ b_out;
            b_out <= a_out ^ b_out;
            a_out <= a_out ^ b_out;
        end
    end
    endmodule
```

4. Design following using Verilog:

I. **XNOR Gate**
II. **D-FF**
III. **2:1 Mux**
IV. **2-bit Full adder**

5. Design a Verilog code to execute following truth table:

Input		Output	
a	**b**	**x**	**y**
0	**0**	**1**	**0**
0	**1**	**1**	**1**
1	**0**	**0**	**0**
1	**1**	**0**	**1**

```
Module given_design(
    Input a, b ;
    Output reg x, y ;
);
If (~a && ~b)begin
    X <= 1 ;
    Y <= 0 ;
end
Elseif (~a && b)begin
    X <= 0 ;
```

```
      Y <= 0 ;
End
Elseif (a && ~b)begin
      X <= 0 ;
      Y <= 1 ;
End
Else (a && b)begin
      X <= 1 ;
      Y <= 1 ;
end
```