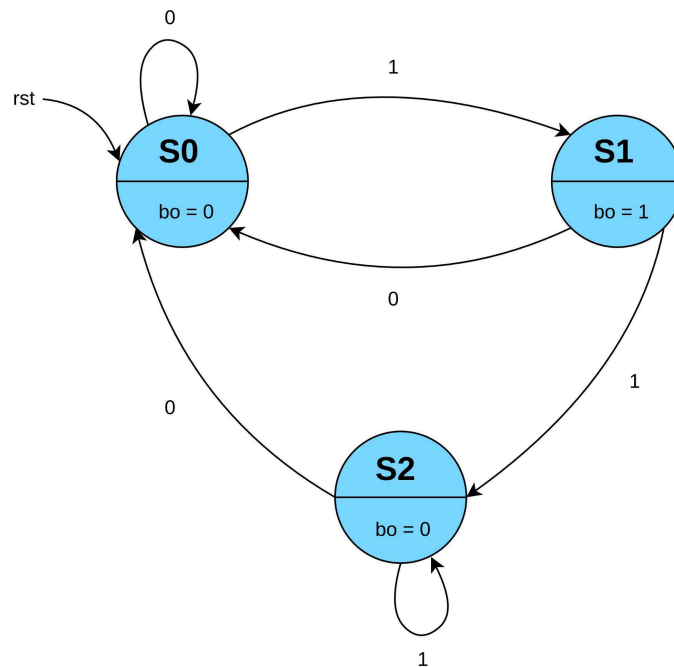


**Module: R3: DLD + DSD****Section: Sequential Circuits Task: Assignment 2****Assignment 2****Sequential Circuits****➤ Question 1: Design button press synchronizer:****a. FSM Diagram:****b. Truth Table:**

One-Hot Encoding	S0 = 001	S1 = 010	S2 = 100
------------------	----------	----------	----------

Present State			Input	Next State			Output	Inputs to Flip-Flops		
Q <sub>C</sub>	Q <sub>B</sub>	Q <sub>A</sub>	b <sub>i</sub>	Q <sub>C+1</sub>	Q <sub>B+1</sub>	Q <sub>A+1</sub>	b <sub>o</sub>	D <sub>C</sub>	D <sub>B</sub>	D <sub>A</sub>
0	0	1	0	0	0	1	0	0	0	1
0	0	1	1	0	1	0	0	0	1	0
0	1	0	0	0	0	1	1	0	0	1
0	1	0	1	1	0	0	1	1	0	0
1	0	0	0	0	0	1	0	0	0	1
1	0	0	1	1	0	0	0	1	0	0

### c. Circuit Diagram:

Using K-Maps:

$Q_C Q_B$		$Q_A b_i$			
		00	01	11	10
$Q_C$	00	X	X	0	0
	01	1	1	X	X
	11	X	X	X	X
	10	0	X	X	X

$b_i$

$b_0 = Q_B$

$Q_C Q_B$		$Q_A b_i$			
		00	01	11	10
$Q_C$	00	X	X		
	01		1	X	X
	11	X	X	X	X
	10		1	X	X

$b_i$

$D_C = Q_A' b_i$

$Q_C Q_B$		$Q_A b_i$			
		00	01	11	10
$Q_C$	00	X	X	1	
	01			X	X
	11	X	X	X	X
	10		1	X	X

$b_i$

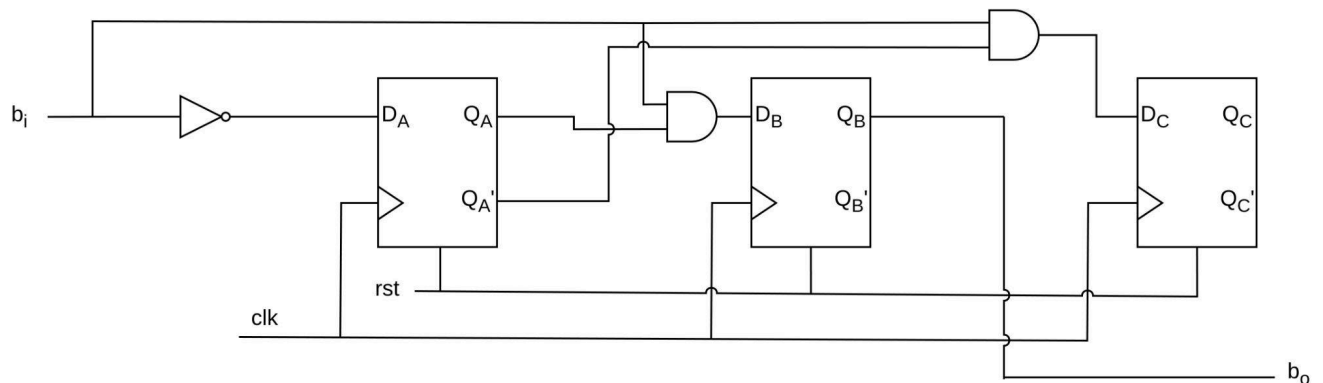
$D_B = Q_A b_i$

$Q_C Q_B$		$Q_A b_i$			
		00	01	11	10
$Q_C$	00	X	X		1
	01	1		X	X
	11	X	X	X	X
	10	1		X	X

$b_i$

$D_A = b_i'$

Here's the final circuit:



#### d. Verilog Code:

```
module d_ff(input clk, rst, d, output reg q);
```

```
    always @(posedge clk or posedge rst) begin
```

```
        if (rst)
```

```
            q <= 0;
```

```
        else
```

```
            q <= d;
```

```
    end
```

```
endmodule
```

```
module d_ff2 (input clk, rst, d, output reg q);
```

```
    always @(posedge clk or posedge rst) begin
```

```
        if (rst)
```

```
            //LSB = 1 for 1-Hot Encoding
```

```
            q <= 1;
```

```
        else
```

```
            q <= d;
```

```
    end
```

```
endmodule
```

```
module button_press_synchronizer (input clk, rst, bi, output reg bo);
```

```

        wire Da, Db, Dc, Qa, Qb, Qc;

        d_ff2 A(.d(Da), .clk(clk), .rst(rst), .q(Qa));
        d_ff B(.d(Db), .clk(clk), .rst(rst), .q(Qb));
        d_ff C(.d(Dc), .clk(clk), .rst(rst), .q(Qc));

        assign Da = ~bi;
        assign Db = bi & Qa;
        assign Dc = ~Qa & bi;
        assign bo = Qb;
    endmodule

```

#### e. Testbench:

```

    module tb_button_press_synchronizer;
        reg clk, rst, bi;
        wire bo;

        button_press_synchronizer m1 (.clk(clk), .rst(rst), .bi(bi),
        .bo(bo));

        initial begin
            clk = 0;
            rst = 0;
            bi = 0;
        end

        always #5 clk = ~clk;
        initial begin

            $dumpvars;

            repeat(2) @(posedge clk);

            rst = 1;
            repeat(3) @(posedge clk);

            rst = 0;
            @(posedge clk);
            bi = 1;

```

```

        repeat (1.5) @(posedge clk)
        $display("Checking for Output Pulse");
        if (bo != 1) begin
            $display("Output is Incorrect Error: Line: 31");
            $finish;
        end

        repeat (1.5) @(posedge clk);
        $display("Checking if Output had one cycle");
        if (bo != 0) begin
            $display("Output is Incorrect Error: Line: 38");
            $finish;
        end

        bi = 0;

        repeat(5) @(posedge clk);

        //Asserting Input again
        bi = 1;
        repeat(5) @(posedge clk);
        bi = 0;

        if (bo != 0) begin
            @(posedge clk)
            if (bo != 1) begin
                $display("Output is Incorrect Line: 49");
                $finish;
            end
        end

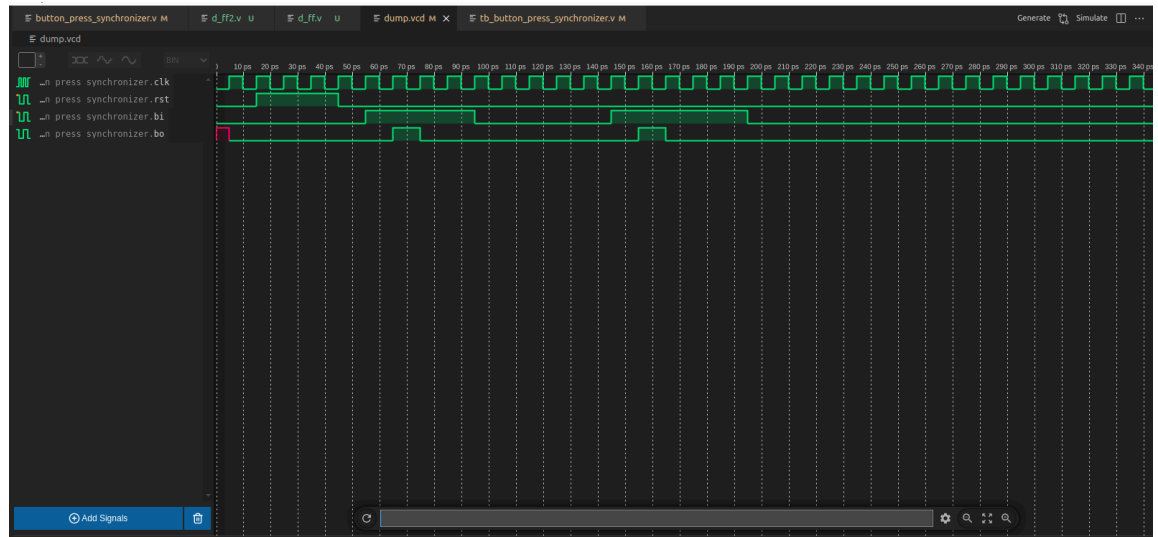
        repeat(15) @(posedge clk);

        $display("All Test cases Passed!");
        $finish;
    end

endmodule

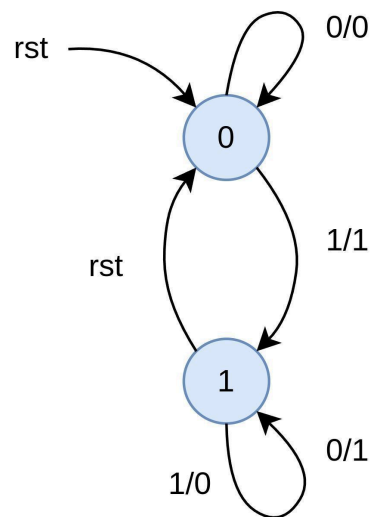
```

#### f. Output:



➤ **Question 2: Serial Input 2's Complementer:**

**g. State Diagram:**



**h. State Table:**

Present State	Input	Next State	Output	Flip-Flop Input
$Q_t$	$x$	$Q_{t+1}$	$y$	$D$
0	0	0	0	0
0	1	1	1	1
1	0	1	1	1
1	1	1	0	1

K-maps:

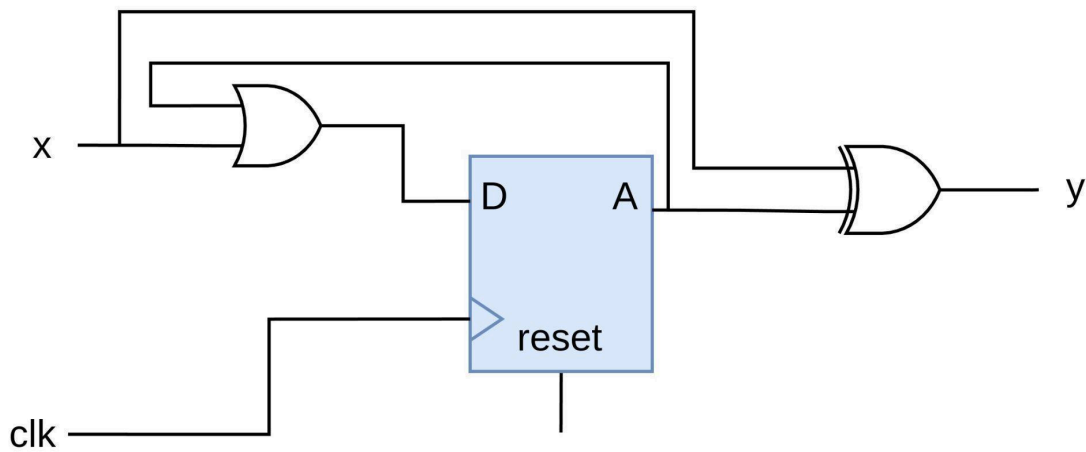
		x	
		0	1
A	0		1
	1	1	1

$$D = x + A$$

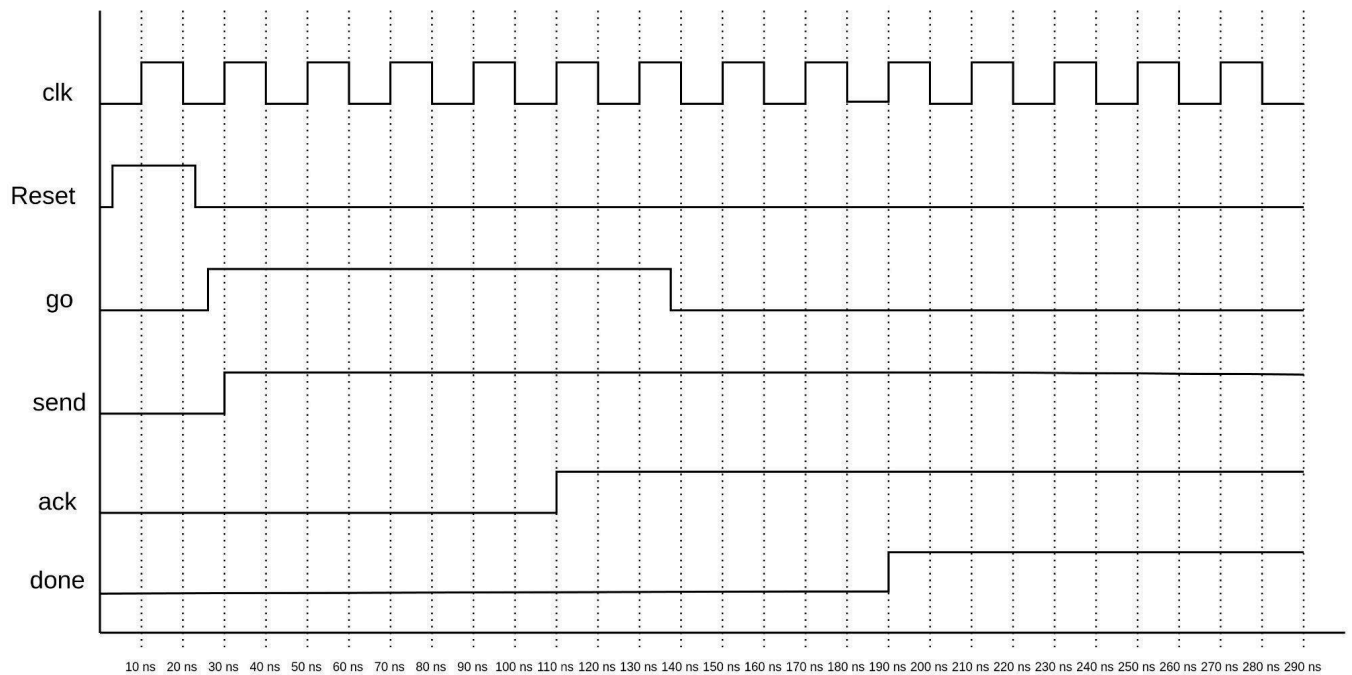
		x	
		0	1
A	0		1
	1	1	

$$y = x \oplus A$$

i. Circuit Diagram:

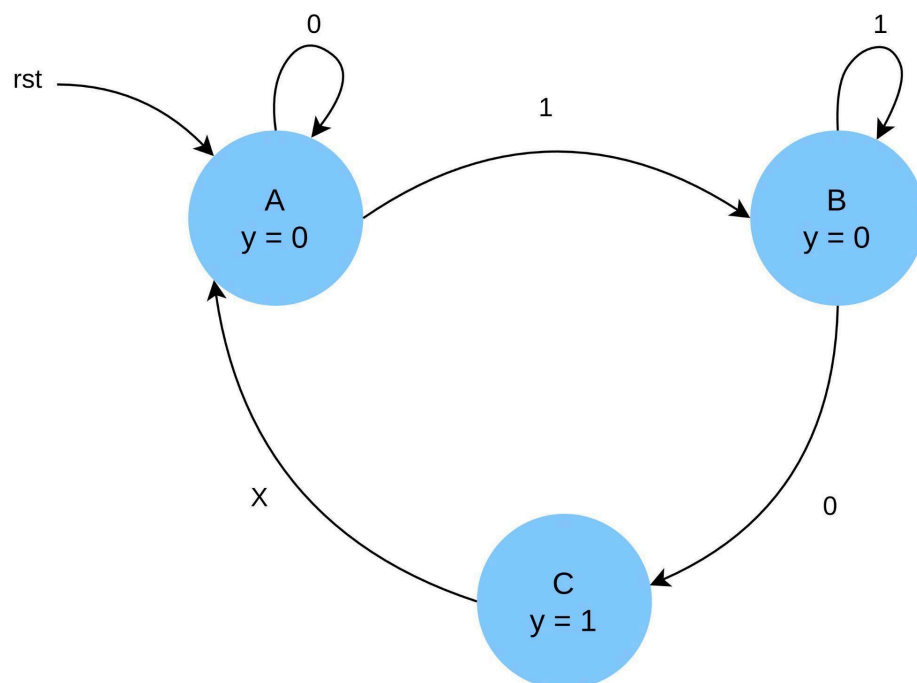


➤ Question 3: Timing Diagram:



➤ **Question 4: Pulse Detector:**

**a. FSM Diagram:**



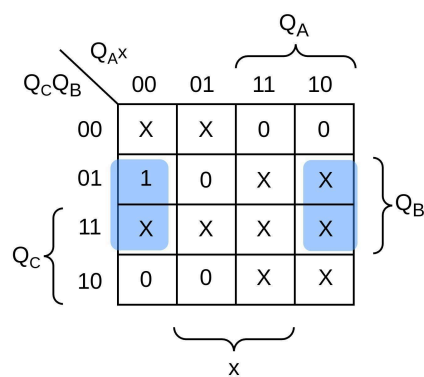
**b. Truth Table:**



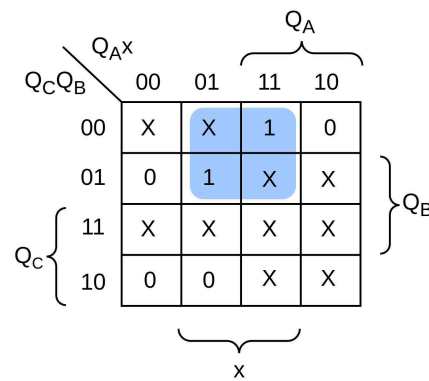
<b>One-Hot Encoding</b>	<b>A = 001</b>	<b>B = 010</b>	<b>C = 100</b>
-------------------------	----------------	----------------	----------------

Present State			Input	Next State			Output	Inputs to Flip-Flops		
$Q_C$	$Q_B$	$Q_A$	$x$	$Q_{C+1}$	$Q_{B+1}$	$Q_{A+1}$	$y$	$D_C$	$D_B$	$D_A$
0	0	1	0	0	0	1	0	0	0	1
0	0	1	1	0	1	0	0	0	1	0
0	1	0	0	1	0	0	0	1	0	0
0	1	0	1	0	1	0	0	0	1	0
1	0	0	0	0	0	1	1	0	0	1
1	0	0	1	0	0	1	1	0	0	1

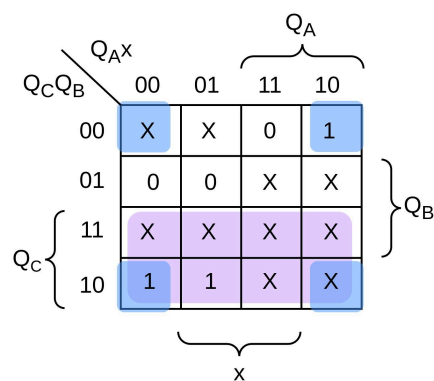
**c. Circuit Diagram:**



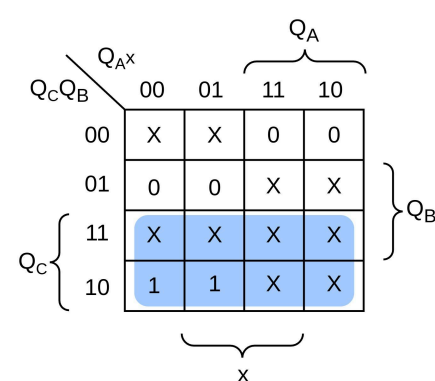
$$D_C = x'Q_B$$



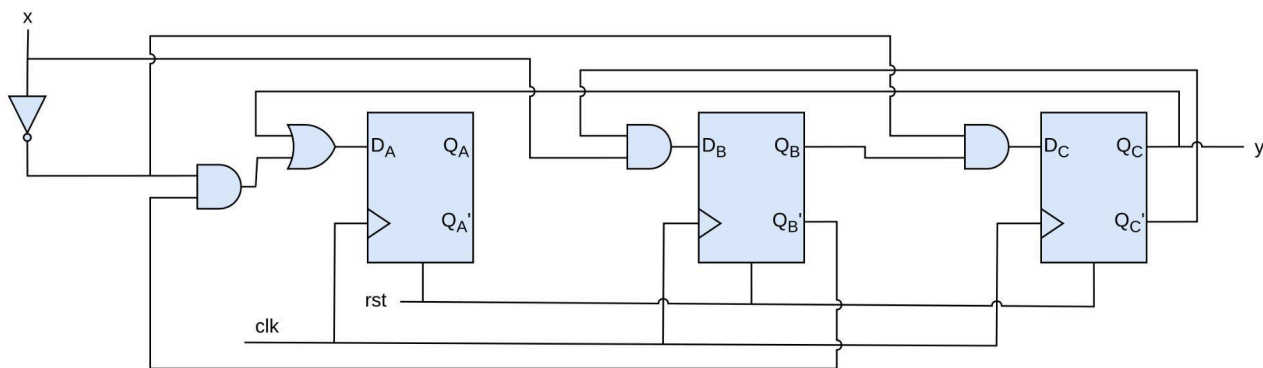
$$D_B = xQ_C'$$



$$D_A = Q_B'x' + Q_C$$



$$y = Q_C$$



#### d. Verilog Code:

```
module d_ff (input clk, rst, d, output reg q);
```

```
always @(posedge clk or posedge rst) begin
```

```
    if (rst)
```

```
        q <= 1'b0;
```

```
    else
```

```
        q <= d;
```

```
end
```

```
endmodule
```

```
module d_ff2 (input clk, rst, d, output reg q);
```

```
always @(posedge clk or posedge rst) begin
```

```
    if (rst)
```

```
        //LSB = 1 for 1-Hot Encoding
```

```
        q <= 1'b1;
```

```
    else
```

```
        q <= d;
```

```
end
```

```
endmodule
```

```
module pulse_detector (input clk, rst, x, output reg y);
```

```

    wire Qa, Qb, Qc, Da, Db, Dc;

    d_ff2 A (.d(Da), .clk(clk), .rst(rst), .q(Qa));
    d_ff  B (.d(Db), .clk(clk), .rst(rst), .q(Qb));
    d_ff  C (.d(Dc), .clk(clk), .rst(rst), .q(Qc));

    assign Da = ((~Qb & ~x) | Qc);
    assign Db = ~Qc & x;
    assign Dc = Qb & ~x;

    assign y = Qc;

endmodule

```

**e. Testbench:**

```

module tb_pulse_detector;

    reg clk, rst, x;
    wire y;

    pulse_detector m0 (.clk(clk), .rst(rst), .x(x), .y(y));

    always #5 clk = ~clk;

    initial begin
        clk = 0;
        rst = 0;
    end

    initial begin
        $dumpvars;

        @(posedge clk);
        rst = 1;

        repeat(4) @(posedge clk);

        rst = 0;

        $display("Check the Output after reset");
        if (y != 0 ) begin
            $display("Incorrect Output after reset");
        end
    end
endmodule

```

```

    end

    @(posedge clk);
    x = 1;

    repeat(3) @(posedge clk);
    x = 0;

    repeat(2) @(negedge clk);
    if (y != 1) begin
        @(negedge clk);
        if (y != 0) begin
            $display("Incorrect Output");
            $finish;
        end
    end
end

    @(posedge clk);
    x = 1;

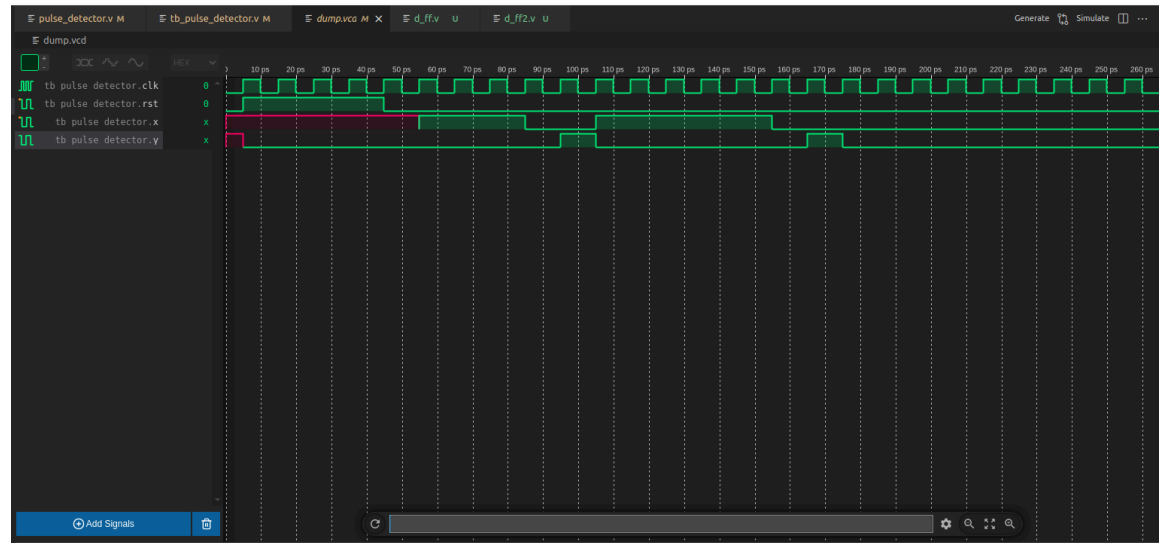
    repeat(5) @(posedge clk);
    x = 0;

    repeat(2) @(negedge clk);
    if (y != 1) begin
        @(negedge clk);
        if (y != 0) begin
            $display("Incorrect Output");
            $finish;
        end
    end
end

    repeat(10) @(posedge clk);
    $display("All Test Cases Passed!");
    $finish;
end
endmodule

```

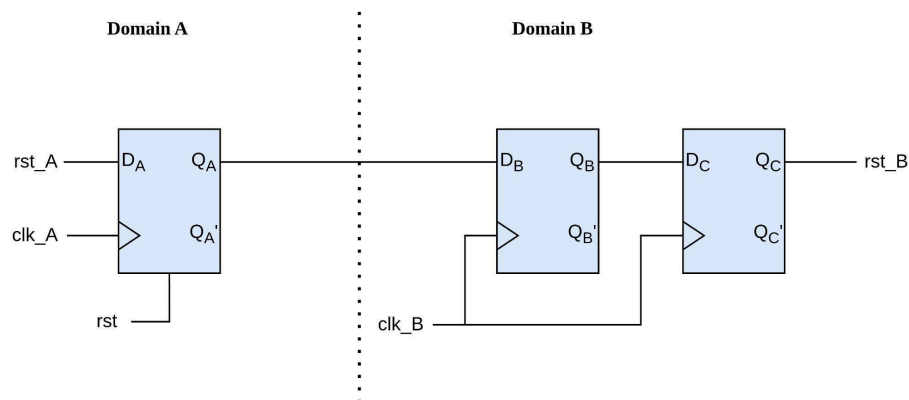
**f. Output:**



➤ **Question 5: Asynchronous reset:**

Since we are dealing with 2 different clock domains. Hence, in order to avoid any glitches or metastability, I'll be using a two-flop synchronizer circuit to map the **reset\_A** signal synchronously with clock of domain B (**clk\_B**)

**a. Circuit Diagram:**



**b. Verilog Code:**

```
module d_ff (input clk, rst, d, output reg q);

always @(posedge clk or posedge rst) begin
    if (rst)
        q <= 1'b0;
    else
        q <= d;
end
```

```

endmodule

module synchronizer (input clk_A, clk_B, rst_A, rst, output
reg rst_B);

    wire w0, w1, w2;

    d_ff m0 (.clk(clk_A), .rst(rst), .d(rst_A), .q(w0));

    //Two Flops Synchronizer
    d_ff m1 (.clk(clk_B), .rst(1'b0), .d(w0), .q(w1));
    d_ff m2 (.clk(clk_B), .rst(1'b0), .d(w1), .q(w2));

    assign rst_B = w2;

endmodule

```

### c. Testbench:

```

module tb_synchronizer;
    reg clk_A, clk_B, rst, rst_A;
    wire rst_B;

    synchronizer dut (.clk_A(clk_A), .clk_B(clk_B), .rst(rst),
.rst_A(rst_A), .rst_B(rst_B));

    always #5 clk_A = ~clk_A;
    always #20 clk_B = ~clk_B;

    initial begin
        $dumpvars;

        clk_A = 0;
        clk_B = 0;
        rst = 0;
        rst_A = 0;

        #20;

        rst = 1;
    end
endmodule

```

```
        #10;  
        rst_A = 1;  
        #40;  
  
        rst = 0;  
        #40;  
  
        rst_A = 0;  
        #80;  
  
        rst_A = 1;  
        #20;  
  
        rst_A = 0;  
        #30;  
  
        rst_A = 1;  
        #30;  
  
        rst_A = 0;  
        #100;  
  
        $finish;  
  
    end  
endmodule
```

**d. Output:**

