Nang Thiri Wutyi
20113
10/24/2024

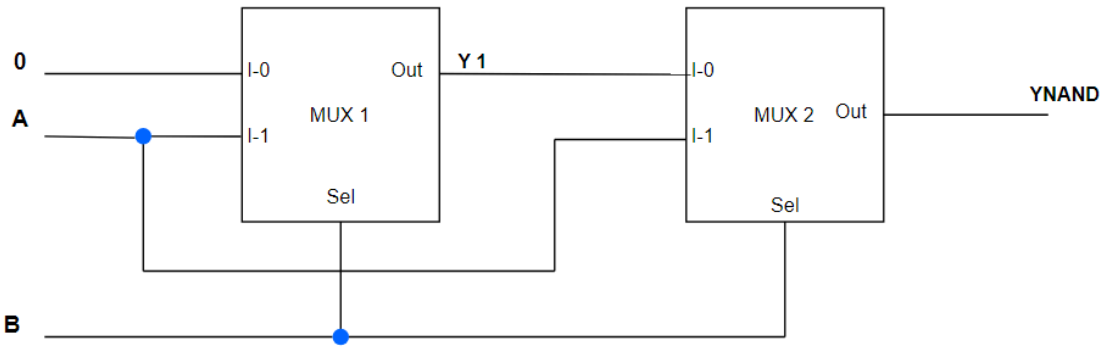
Verilog Lab Midterm Exam

1. NAND gate using 2:1 Multiplexers

Truth Table

A	B	NAND output
0	0	1
0	1	1
1	0	1
1	1	0

$$Y_{NAND} = \overline{A \cdot B} = 1$$



NAND Gate using two Muliplexers

2. RTL module

```
module swap_no_temp(
    input [7:0] A_in,
    input [7:0] B_in,
    input clk,
```

```
    input reset,
    output reg [7:0] A_out,
    output reg [7:0] B_out
);

    always @(posedge clk or posedge reset) begin
        if (reset) begin
            A_out = A_in;  // Initialize A_out with A_in
            B_out = B_in;  // Initialize B_out with B_in
        end else begin
            // Correct XOR Swap operation using blocking assignments
            A_out = A_out ^ B_out;
            B_out = A_out ^ B_out;
            A_out = A_out ^ B_out;
        end
    end
endmodule
```

Testbench

```
module tb_swap_no_temp;
  reg [7:0] A_in, B_in;
  reg clk, reset;
  wire [7:0] A_out, B_out;

  // Instantiate the swap module
  swap_no_temp uut (
    .A_in(A_in),
    .B_in(B_in),
    .clk(clk),
    .reset(reset),   // Connect reset signal
    .A_out(A_out),
    .B_out(B_out)
  );

  // Clock generation
  always #5 clk = ~clk; // Clock toggles every 5 time units

  initial begin
    // Initialize inputs
    A_in = 8'hAA;
```

```
    B_in = 8'h55;
    clk = 0;
    reset = 1;       // Assert reset at the start

    // Release reset after some time
    #10 reset = 0;

    // Print the results
    $monitor("Time: %0t | A_in: %h | B_in: %h | A_out: %h | B_out: %h", $time, A_in,
B_in, A_out, B_out);

    // Stop the simulation after some time
    #60;
    $finish;
  end

endmodule
```

Results

```
[2024-10-24 21:15:13 UTC] iverilog '-Wall' '-g2012' design.sv testbench.sv  &&
unbuffer vvp a.out
Time: 10 | A_in: aa | B_in: 55 | A_out: aa | B_out: 55
Time: 15 | A_in: aa | B_in: 55 | A_out: 55 | B_out: aa
Time: 25 | A_in: aa | B_in: 55 | A_out: aa | B_out: 55
Time: 35 | A_in: aa | B_in: 55 | A_out: 55 | B_out: aa
Time: 45 | A_in: aa | B_in: 55 | A_out: aa | B_out: 55
Time: 55 | A_in: aa | B_in: 55 | A_out: 55 | B_out: aa
Time: 65 | A_in: aa | B_in: 55 | A_out: aa | B_out: 55
testbench.sv:34: $finish called at 70 (1s)
Done
```

3. **RTL module(TFF)**

```
`timescale 1ns/1ps  // Time unit is 1 nanosecond, precision is 1 picosecond

module T_FF (
    input wire T,         // Toggle input
    input wire RST,       // Reset input
    input wire CLK,       // Clock input
    output reg Q          // Output
);
    // Always block for T Flip-Flop behavior
    always @(posedge CLK or posedge RST) begin
        if (RST) begin
```

```
            Q <= 0;      // Reset output to 0
        end else if (T) begin
            Q <= ~Q;     // Toggle output
        end
    end
endmodule
```

Testbench

```
`timescale 1ns/1ps

module testbench;
    reg T;        // T input for the T Flip-Flop
    reg RST;      // Reset signal
    wire Q;       // Output of the T Flip-Flop

    // Instantiate the T Flip-Flop module
    T_FF tff (
        .T(T),
        .RST(RST),
        .Q(Q)
    );

    // VCD Dump to generate waveform
    initial begin
        $dumpfile("T_FF_waveform.vcd"); // Specify the VCD file name
        $dumpvars(0, testbench);         // Dump all variables in the testbench
    end

    initial begin
        // Initialize signals
        T = 0;
        RST = 1;

        // Apply reset
        #10 RST = 0; // Release reset
        #10 T = 1;   // Set T to 1
        #20 T = 0;   // Set T to 0
        #10 T = 1;   // Set T to 1 again
        #30 T = 0;   // Set T to 0
        #10 RST = 1; // Assert reset
```

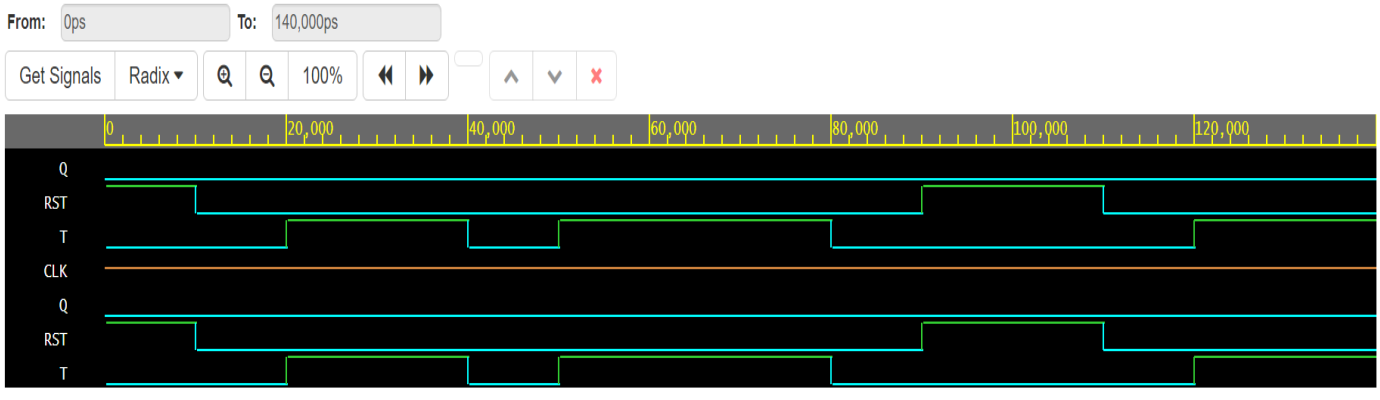
```
        #20 RST = 0; // Release reset
        #10 T = 1;   // Set T to 1 again
        #20 $finish; // End the simulation
    end

    // Display output
    initial begin
        $monitor("Time: %0t | T: %b | RST: %b | Q: %b", $time, T, RST, Q);
    end
endmodule
```

Results

Log

Share

```
[2024-10-24 21:27:32 UTC] iverilog '-Wall' '-g2012' design.sv testbench.sv  &&
unbuffer vvp a.out
Time:                     0 | T: 0 | RST: 1 | Q: 0
Time:                 10000 | T: 0 | RST: 0 | Q: 0
Time:                 20000 | T: 1 | RST: 0 | Q: 0
Time:                 25000 | T: 1 | RST: 0 | Q: 1
Time:                 35000 | T: 1 | RST: 0 | Q: 0
Time:                 45000 | T: 1 | RST: 0 | Q: 1
Time:                 55000 | T: 1 | RST: 0 | Q: 0
Time:                 60000 | T: 0 | RST: 0 | Q: 0
Time:                 90000 | T: 1 | RST: 0 | Q: 0
Time:                 95000 | T: 1 | RST: 0 | Q: 1
Time:                105000 | T: 1 | RST: 0 | Q: 0
Time:                115000 | T: 1 | RST: 0 | Q: 1
Time:                120000 | T: 0 | RST: 0 | Q: 1
Final Output: Q = 1
testbench.sv:44: $finish called at 150000 (1ps)
Done
```



Note: To revert to EPWave opening in a new browser window, set that option on your profile page.

4. **Comparator module(Gate level)**

```
module comparator_1bit (
    input A,
    input B,
    output A_gt,  // A > B
    output A_eq,  // A = B
    output A_lt   // A < B
);
    wire not_A, not_B;  // Intermediate wires for NOT operations

    // Invert A and B
    not(not_A, A);
    not(not_B, B);

    // A = B condition
    wire eq1, eq2;  // Intermediate signals for equality
    and(eq1, not_A, not_B); // A and B both 0
    and(eq2, A, B);         // A and B both 1
    or(A_eq, eq1, eq2);     // A is equal to B

    // A > B condition
    and(A_gt, A, not_B);    // A is 1 and B is 0

    // A < B condition
    and(A_lt, not_A, B);    // A is 0 and B is 1
endmodule
```

Testbench

```
module testbench;
    reg A, B;               // Test inputs
    wire A_gt, A_eq, A_lt;    // Outputs from the comparator

    // Instantiate the comparator
    comparator_1bit comp (
        .A(A),
        .B(B),
        .A_gt(A_gt),
        .A_eq(A_eq),
        .A_lt(A_lt)
```

```
    );

    initial begin
        // Test cases
        $monitor("A=%b, B=%b | A_gt=%b, A_eq=%b, A_lt=%b", A, B, A_gt, A_eq,
A_lt);

        // Test case: A = 0, B = 0
        A = 0; B = 0; #10;
        // Test case: A = 0, B = 1
        A = 0; B = 1; #10;
        // Test case: A = 1, B = 0
        A = 1; B = 0; #10;
        // Test case: A = 1, B = 1
        A = 1; B = 1; #10;

        // Finish simulation
        $finish;
    end
endmodule
```

Results

```
[2024-10-24 21:32:55 UTC] iverilog '-Wall' '-g2012' design.sv testbench.sv  &&
unbuffer vvp a.out
A=0, B=0 | A_gt=0, A_eq=1, A_lt=0
A=0, B=1 | A_gt=0, A_eq=0, A_lt=1
A=1, B=0 | A_gt=1, A_eq=0, A_lt=0
A=1, B=1 | A_gt=0, A_eq=1, A_lt=0
testbench.sv:28: $finish called at 40 (1s)
Done
```

5. **Design Module DFF (Gate level)**

```
module DFF_gate_level(
    output Q,
    output Q_bar,
    input D,
    input CLK
);
    wire D_bar;
    wire w1, w2, w3, w4;

    // Invert D
```

```
    not (D_bar, D);

    // NAND gates for the D flip-flop
    nand (w1, D, CLK);          // First NAND gate
    nand (w2, D_bar, CLK);      // Second NAND gate
    nand (w3, w1, w4);          // Third NAND gate (feedback)
    nand (w4, w2, w3);          // Fourth NAND gate (feedback)

    assign Q = w3;              // Output Q
    assign Q_bar = w4;          // Output Q_bar (inverse of Q)

endmodule
```

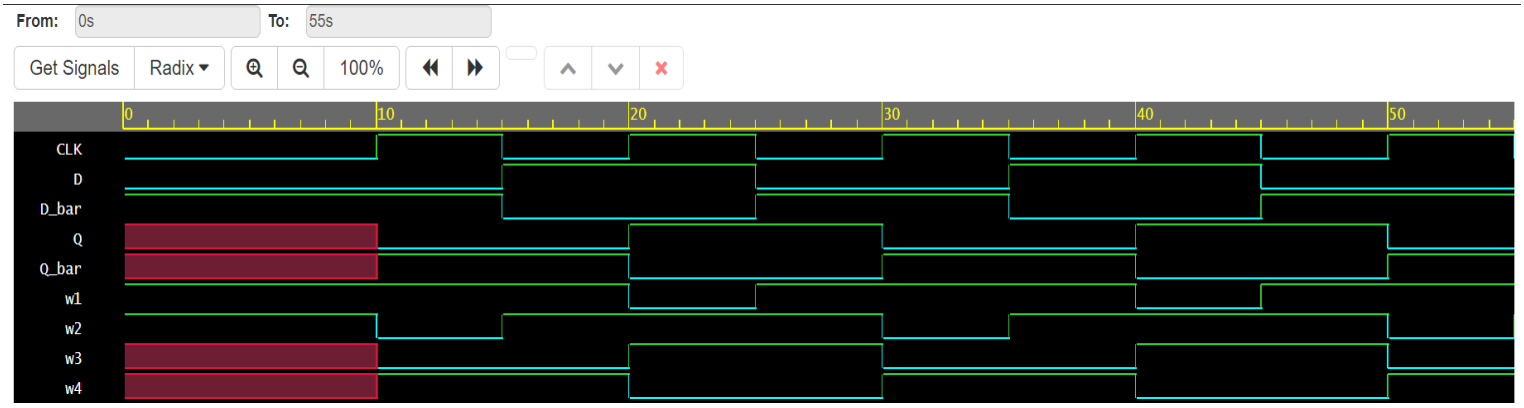
Testbench

```
module testbench();
    reg D;          // Input for the D flip-flop
    reg CLK;        // Clock signal
    wire Q;         // Output Q
    wire Q_bar;     // Output Q_bar

    // Instantiate the D flip-flop
    DFF_gate_level dff (.Q(Q), .Q_bar(Q_bar), .D(D), .CLK(CLK));

    initial begin
        // Initialize signals
        CLK = 0;
        D = 0;

        // Open a VCD file for waveform dumping
        $dumpfile("waveform.vcd");  // Name of the VCD file
        $dumpvars(0, testbench);    // Dump all variables in this module

        // Monitor changes
        $monitor("Time: %0d | D: %b | CLK: %b | Q: %b | Q_bar: %b", $time, D, CLK, Q,
Q_bar);

        // Test sequence
        #5 CLK = 1; D = 0; // Rising edge, Q should remain 0
        #5 CLK = 0;        // Falling edge
        #5 CLK = 1; D = 1; // Rising edge, Q should now be 1
```

```
        #5 CLK = 0;       // Falling edge
        #5 CLK = 1; D = 0; // Rising edge, Q should now be 0
        #5 CLK = 0;       // Falling edge
        #5 CLK = 1; D = 1; // Rising edge, Q should now be 1
        #5 CLK = 0;       // Falling edge
        #5 CLK = 1; D = 0; // Rising edge, Q should now be 0
        #5 CLK = 0;       // Final falling edge

        #5 $finish;       // End the simulation
    end

    // Generate clock signal
    always #5 CLK = ~CLK;  // Toggle CLK every 5 time units
endmodule
```

Results

```
[2024-10-24 20:50:50 UTC] iverilog '-Wall' '-g2012' design.sv testbench.sv  && unbuffer vvp a.out
Time: 0 | D: 0 | CLK: 0 | Q: x | Q_bar: x
Time: 5 | D: 0 | CLK: 1 | Q: 0 | Q_bar: 1
Time: 10 | D: 0 | CLK: 0 | Q: 0 | Q_bar: 1
Time: 15 | D: 1 | CLK: 1 | Q: 1 | Q_bar: 0
Time: 20 | D: 1 | CLK: 0 | Q: 1 | Q_bar: 0
Time: 25 | D: 0 | CLK: 1 | Q: 0 | Q_bar: 1
Time: 30 | D: 0 | CLK: 0 | Q: 0 | Q_bar: 1
Time: 35 | D: 1 | CLK: 1 | Q: 1 | Q_bar: 0
Time: 40 | D: 1 | CLK: 0 | Q: 1 | Q_bar: 0
Time: 45 | D: 0 | CLK: 1 | Q: 0 | Q_bar: 1
Time: 50 | D: 0 | CLK: 0 | Q: 0 | Q_bar: 1
testbench.sv:30: $finish called at 55 (1s)
Done
```



6. Design module

```
// UDP for AND gate
primitive and_udp (out, in1, in2);
   output out;
   input in1, in2;

   table
      // in1  in2  : out
         0    0    : 0;
         0    1    : 0;
         1    0    : 0;
         1    1    : 1;
   endtable
endprimitive

// UDP for OR gate
primitive or_udp (out, in1, in2);
   output out;
   input in1, in2;

   table
      // in1  in2  : out
         0    0    : 0;
         0    1    : 1;
         1    0    : 1;
         1    1    : 1;
   endtable
endprimitive

`timescale 1ns/1ps

module top_module (
   input A, B, C,
   output X, Y
);

   wire and1_out, and2_out; // Wires to connect intermediate signals

   // Instantiate AND gates
   and_udp U1 (and1_out, A, B); // First AND gate
   and_udp U2 (and2_out, A, C); // Second AND gate
```

```
    // Instantiate OR gates
    or_udp U3 (X, and1_out, and2_out); // First OR gate for output X
    or_udp U4 (Y, B, C);               // Second OR gate for output Y

endmodule
```

Testbench module

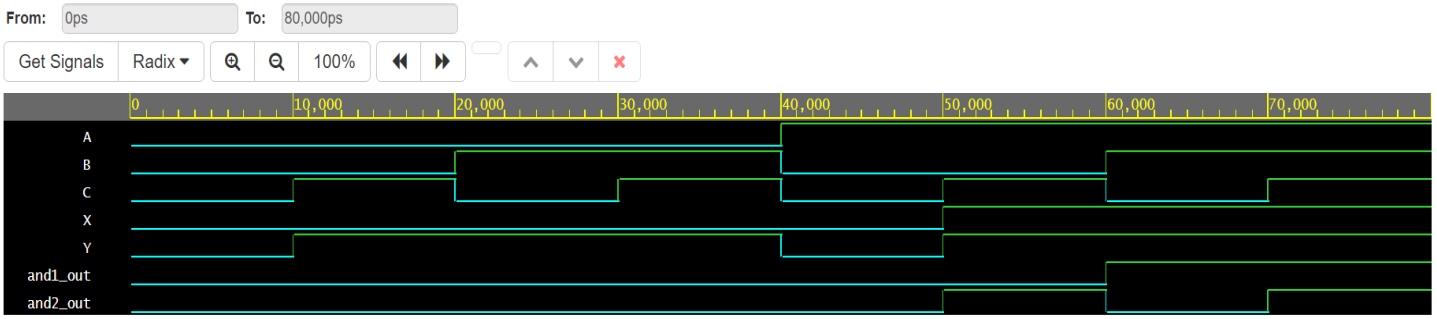
```
`timescale 1ns/1ps

module tb_top_module;

    // Inputs
    reg A;
    reg B;
    reg C;

    // Outputs
    wire X;
    wire Y;

    // Instantiate the top module
    top_module uut (
        .A(A),
        .B(B),
        .C(C),
        .X(X),
        .Y(Y)
    );

    // Test stimulus
    initial begin
        // Initialize inputs
        A = 0; B = 0; C = 0;
        #10;

        // Apply test cases
        A = 0; B = 0; C = 1;
        #10;
```

```
        A = 0; B = 1; C = 0;
        #10;

        A = 0; B = 1; C = 1;
        #10;

        A = 1; B = 0; C = 0;
        #10;

        A = 1; B = 0; C = 1;
        #10;

        A = 1; B = 1; C = 0;
        #10;

        A = 1; B = 1; C = 1;
        #10;

        // Finish simulation
        $finish;
    end

    // Monitor values for debugging
    initial begin
        $monitor("Time = %0t | A = %b, B = %b, C = %b -> X = %b, Y = %b",
                 $time, A, B, C, X, Y);
    end

    // Dump waveform data
    initial begin
        $dumpfile("waveform.vcd"); // This creates the VCD file to be used for waveform
viewing
        $dumpvars(0, tb_top_module); // Dump all variables of the testbench
    end

endmodule
```

Results

```
[2024-10-24 21:01:51 UTC] iverilog '-Wall' '-g2012' design.sv testbench.sv  &&
unbuffer vvp a.out
VCD info: dumpfile waveform.vcd opened for output.
```

```
Time = 0 | A = 0, B = 0, C = 0 -> X = 0, Y = 0
Time = 10000 | A = 0, B = 0, C = 1 -> X = 0, Y = 1
Time = 20000 | A = 0, B = 1, C = 0 -> X = 0, Y = 1
Time = 30000 | A = 0, B = 1, C = 1 -> X = 0, Y = 1
Time = 40000 | A = 1, B = 0, C = 0 -> X = 0, Y = 0
Time = 50000 | A = 1, B = 0, C = 1 -> X = 1, Y = 1
Time = 60000 | A = 1, B = 1, C = 0 -> X = 1, Y = 1
Time = 70000 | A = 1, B = 1, C = 1 -> X = 1, Y = 1
testbench.sv:52: $finish called at 80000 (1ps)
Done
```



Note: To revert to EPWave opening in a new browser window, set that option on your profile page.