Nang Thiri Wutyi

20113

10/24/2024

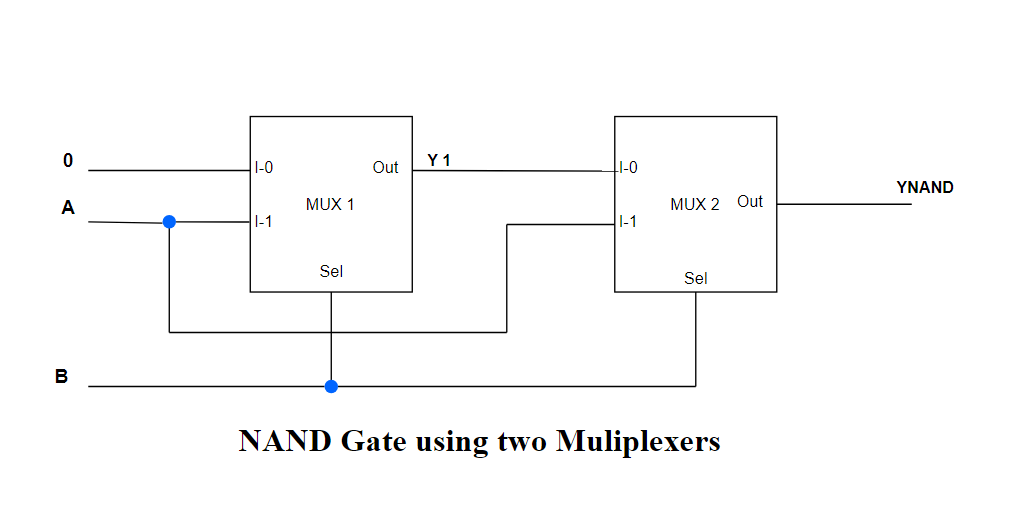
**Verilog Lab Midterm Exam**

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 |





1. **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**

1. **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](https://www.edaplayground.com/#results)
* [Share](https://www.edaplayground.com/#share-tab)

**[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**

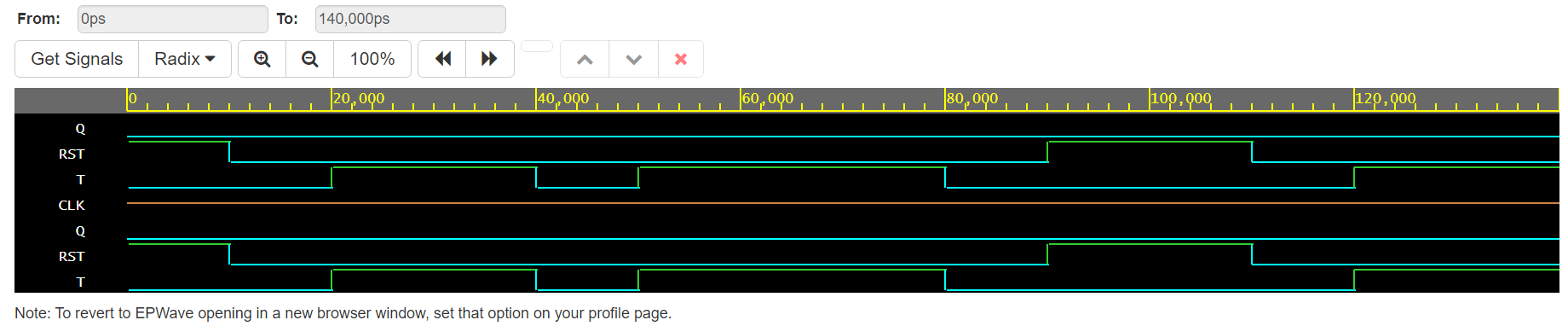
**Time: 115000 | T: 1 | RST: 0 | Q: 1**

**Time: 120000 | T: 0 | RST: 0 | Q: 1**

**Final Output: Q = 1**

**testbench.sv:44: $finish called at 150000 (1ps)**

**Done**



1. **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**

1. **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

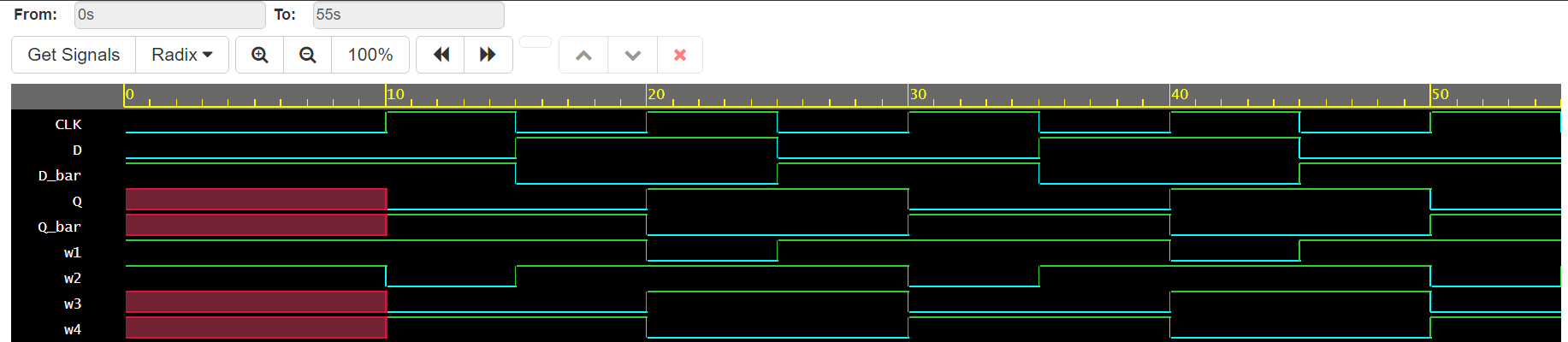
Time: 40 | D: 1 | CLK: 0 | Q: 1 | Q\_bar: 0

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



1. **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**

