**Q1.**

`timescale 1ns/1ps // Add time unit and time precision

module nand\_using\_mux (

input A, // First input

input B, // Second input

output Y // Output (A NAND B)

);

wire not\_B;

// Invert B (NOT B)

assign not\_B = ~B;

// 2:1 MUX logic

// If A = 0, Y = 1

// If A = 1, Y = not\_B

assign Y = (A) ? not\_B : 1'b1;

endmodule

`timescale 1ns/1ps // Add time unit and time precision

module test\_nand\_using\_mux;

// Inputs

reg A;

reg B;

// Output

wire Y;

// Instantiate the NAND gate using MUX module

nand\_using\_mux uut (

.A(A),

.B(B),

.Y(Y)

);

// Test cases

initial begin

// Monitor the inputs and output

$monitor("Time = %0t | A = %b | B = %b | Y (A NAND B) = %b", $time, A, B, Y);

// Test case 1: A = 0, B = 0 -> Y = 1

A = 0; B = 0;

#10;

// Test case 2: A = 0, B = 1 -> Y = 1

A = 0; B = 1;

#10;

// Test case 3: A = 1, B = 0 -> Y = 1

A = 1; B = 0;

#10;

// Test case 4: A = 1, B = 1 -> Y = 0

A = 1; B = 1;

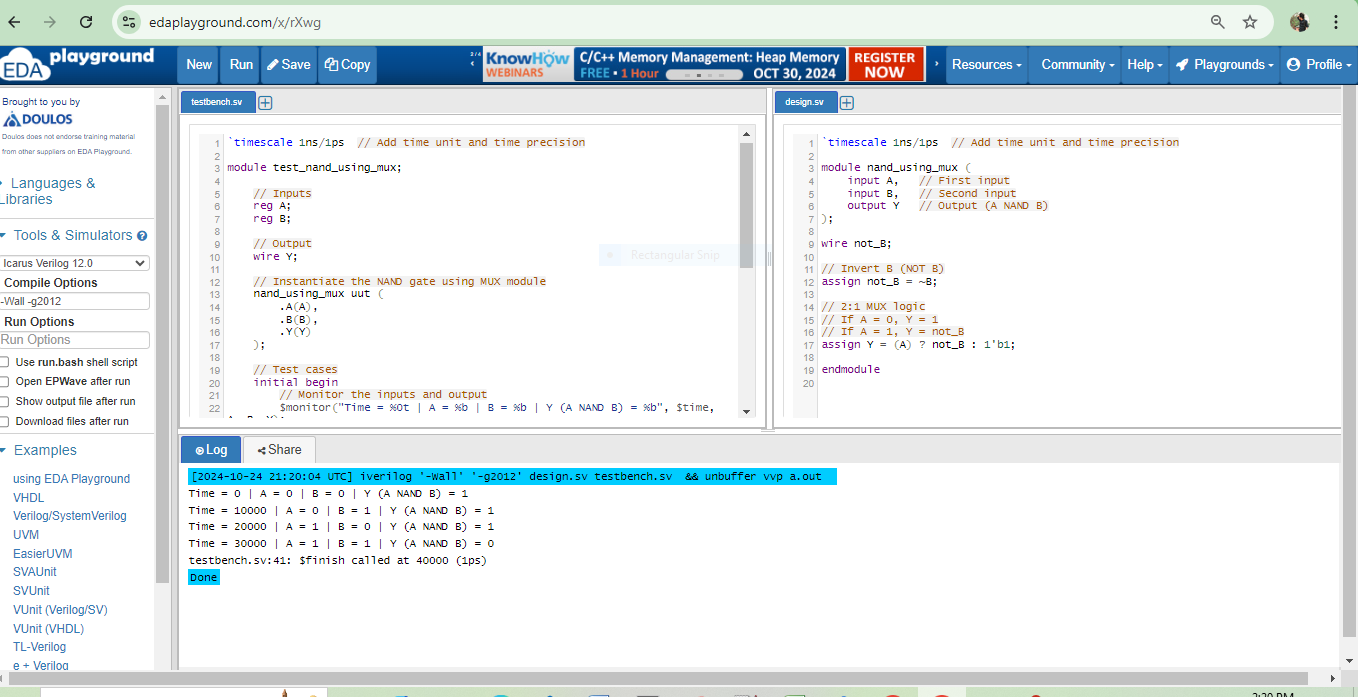
#10;

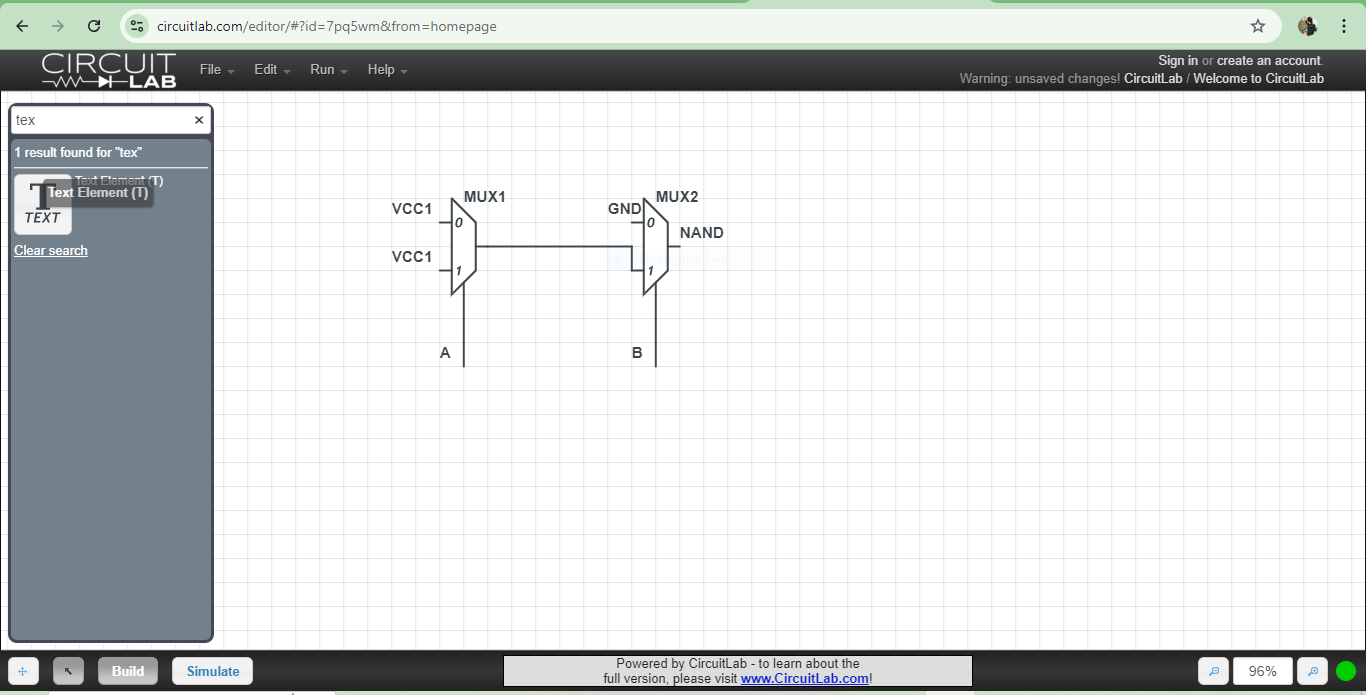
// End the simulation

$finish;

end

endmodule





**Q2.**

module swap\_registers (

input wire [7:0] A\_in, // Input to register A (8 bits)

input wire [7:0] B\_in, // Input to register B (8 bits)

input wire clk, // Clock signal

input wire rst, // Reset signal

input wire swap\_en, // Enable signal for swap operation

output reg [7:0] A\_out, // Output of register A (after swap)

output reg [7:0] B\_out // Output of register B (after swap)

);

reg [7:0] A\_temp, B\_temp;

always @(posedge clk or posedge rst) begin

if (rst) begin

// Reset both registers to initial values

A\_out <= 8'b0;

B\_out <= 8'b0;

A\_temp <= 8'b0;

B\_temp <= 8'b0;

end

else if (swap\_en) begin

// XOR swap logic in 3 steps

// Step 1: A\_out = A\_in ^ B\_in

A\_temp <= A\_in ^ B\_in;

// Step 2: B\_out = A\_in

B\_temp <= A\_in;

end

else begin

// Step 3: Swap the final values

A\_out <= A\_temp ^ B\_temp;

B\_out <= A\_temp ^ B\_in;

end

end

endmodule

module test\_swap\_registers;

reg [7:0] A\_in;

reg [7:0] B\_in;

reg clk;

reg rst;

reg swap\_en;

wire [7:0] A\_out;

wire [7:0] B\_out;

// Instantiate the swap\_registers module

swap\_registers uut (

.A\_in(A\_in),

.B\_in(B\_in),

.clk(clk),

.rst(rst),

.swap\_en(swap\_en),

.A\_out(A\_out),

.B\_out(B\_out)

);

// Clock generation

always #5 clk = ~clk; // 10ns clock period

initial begin

// Initialize inputs

clk = 0;

rst = 1;

swap\_en = 0;

A\_in = 8'hAA; // Example value for A (0xAA)

B\_in = 8'h55; // Example value for B (0x55)

// Release reset after 10ns

#10 rst = 0;

// Enable swapping

#10 swap\_en = 1;

#10 swap\_en = 0; // Disable after one clock cycle

// Wait and observe results

#50 $finish;

end

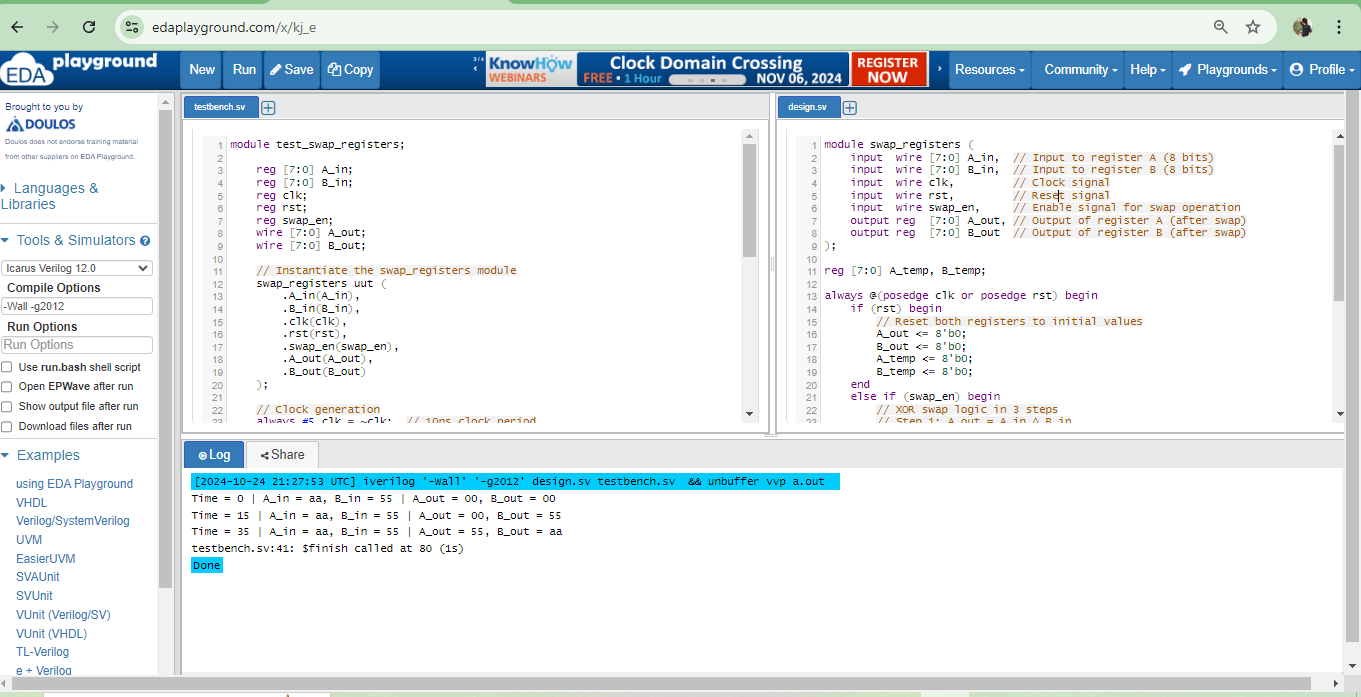
initial begin

// Monitor changes in the signals

$monitor("Time = %0t | A\_in = %h, B\_in = %h | A\_out = %h, B\_out = %h", $time, A\_in, B\_in, A\_out, B\_out);

end

endmodule



**Q3.**  
`timescale 1ns / 1ps // Time unit is 1 nanosecond, time precision is 1 picosecond

module t\_ff (

input wire clk, // Clock input

input wire rst, // Asynchronous reset input

input wire t, // Toggle control input

output reg q // Output of the T-FF

);

// Always block triggered on the rising edge of the clock or reset

always @(posedge clk or posedge rst) begin

if (rst)

q <= 1'b0; // Reset output to 0

else if (t)

q <= ~q; // Toggle output when t is high

else

q <= q; // Hold state when t is low

end

endmodule

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

module t\_ff\_tb;

reg clk, rst, t;

wire q;

// Instantiate the T-FF module

t\_ff my\_tff(

.clk(clk),

.rst(rst),

.t(t),

.q(q)

);

// Clock generation: toggle clock every 10 ns (20 ns period)

always #10 clk = ~clk;

// Stimulus block to test the T-FF

initial begin

// Initialize signals

clk = 0;

rst = 1; t = 0;

#15 rst = 0; // Release reset after 15 ns

#20 t = 1; // Enable toggle after 35 ns

#40 t = 0; // Disable toggle after 75 ns

#20 t = 1; // Enable toggle again after 95 ns

#60 t = 0; // Disable toggle after 155 ns

#20;

$finish; // Finish simulation after 175 ns

end

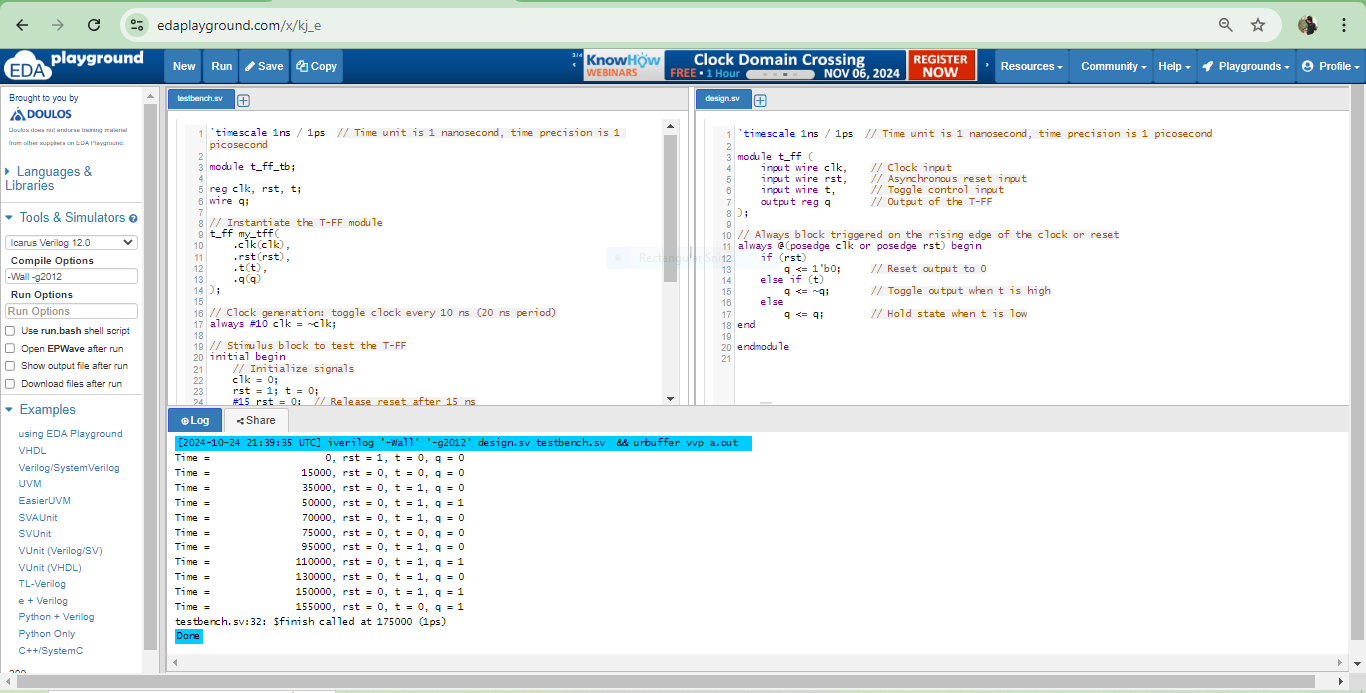
// Monitor changes during simulation

initial begin

$monitor("Time = %t, rst = %b, t = %b, q = %b", $time, rst, t, q);

end

endmodule



**Q4.**

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

module comparator\_1bit(

input wire A, // 1-bit input A

input wire B, // 1-bit input B

output wire A\_greater\_B, // Output is 1 if A > B

output wire A\_equal\_B, // Output is 1 if A == B

output wire A\_less\_B // Output is 1 if A < B

);

// A > B happens when A is 1 and B is 0

assign A\_greater\_B = A & ~B;

// A == B happens when both A and B are either 0 or both are 1

assign A\_equal\_B = ~(A ^ B); // XNOR gate (A equals B)

// A < B happens when A is 0 and B is 1

assign A\_less\_B = ~A & B;

endmodule

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

module comparator\_1bit\_tb;

reg A, B;

wire A\_greater\_B, A\_equal\_B, A\_less\_B;

// Instantiate the comparator

comparator\_1bit uut(

.A(A),

.B(B),

.A\_greater\_B(A\_greater\_B),

.A\_equal\_B(A\_equal\_B),

.A\_less\_B(A\_less\_B)

);

// Stimulus block

initial begin

// Test case 1: A = 0, B = 0

A = 0; B = 0;

#10;

// Test case 2: A = 0, B = 1

A = 0; B = 1;

#10;

// Test case 3: A = 1, B = 0

A = 1; B = 0;

#10;

// Test case 4: A = 1, B = 1

A = 1; B = 1;

#10;

$finish;

end

// Monitor changes

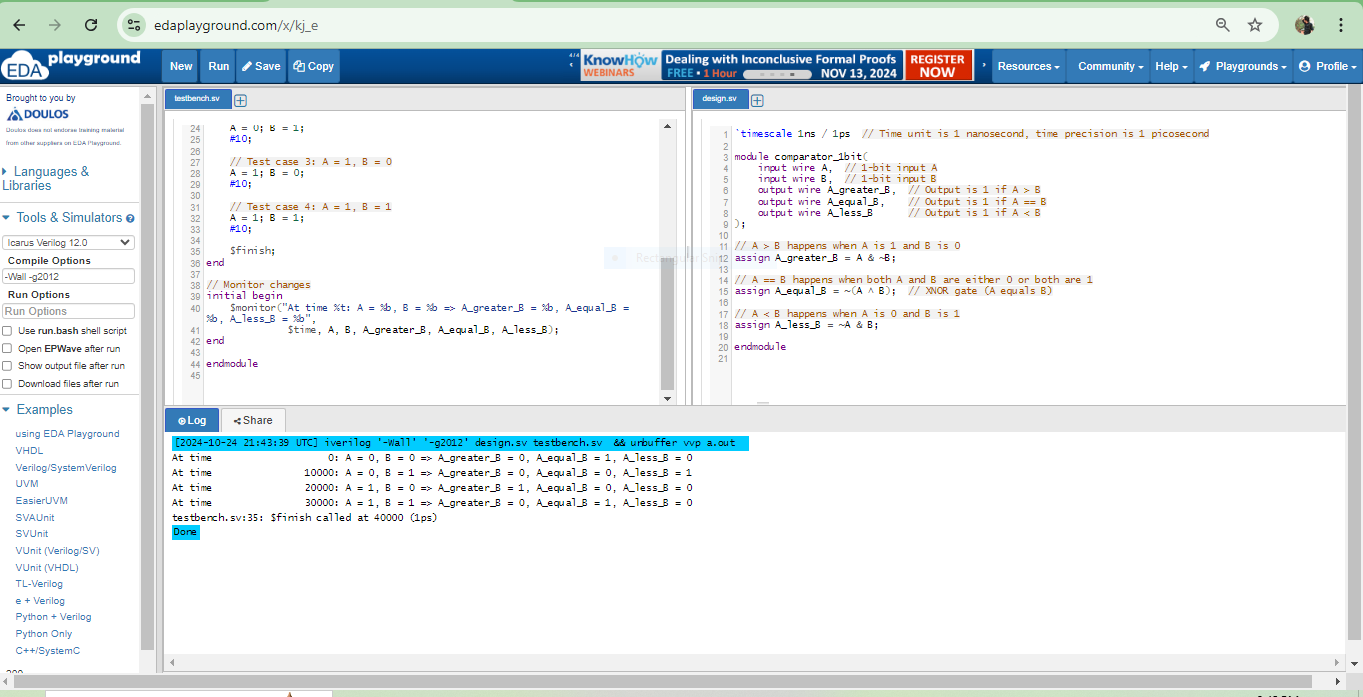
initial begin

$monitor("At time %t: A = %b, B = %b => A\_greater\_B = %b, A\_equal\_B = %b, A\_less\_B = %b",

$time, A, B, A\_greater\_B, A\_equal\_B, A\_less\_B);

end

endmodule



**Q5.**

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

module dff\_gatelevel (

input wire D, // Data input

input wire CLK, // Clock input

output wire Q, // Q output

output wire Qn // Q-bar output (negation of Q)

);

wire Dn; // Inverted D

wire S, R; // Outputs of the AND gates

// Instantiate NOT gate for D negation

not U1 (Dn, D);

// Instantiate AND gates

and U2 (S, D, CLK); // Set input: D AND CLK

and U3 (R, Dn, CLK); // Reset input: NOT D AND CLK

// Instantiate NOR gates for SR latch

nor U4 (Q, R, Qn); // NOR gate for Q

nor U5 (Qn, S, Q); // NOR gate for Q-bar

endmodule

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

module dff\_gatelevel\_tb;

reg D, CLK; // Inputs for the DFF

wire Q, Qn; // Outputs from the DFF

// Instantiate the DFF module

dff\_gatelevel uut (

.D(D),

.CLK(CLK),

.Q(Q),

.Qn(Qn)

);

// Generate clock signal with a 20 ns period (toggle every 10 ns)

always #10 CLK = ~CLK;

// Initial block to set input stimulus

initial begin

// Initialize inputs

D = 0;

CLK = 0;

// Apply input stimulus

#15 D = 1; // Set D to 1

#30 D = 0; // Set D to 0

#30 D = 1; // Set D to 1 again

#20 D = 0; // Set D to 0 again

#50 $finish; // End simulation

end

// Monitor to observe output changes

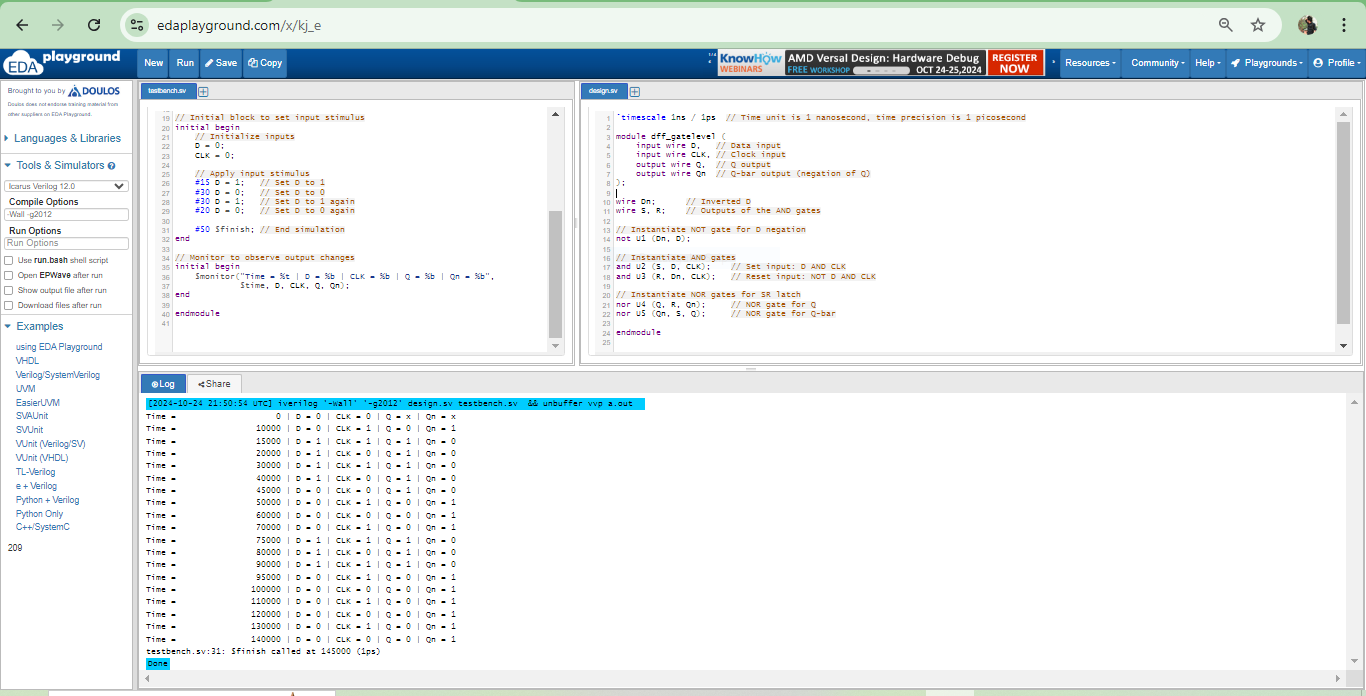
initial begin

$monitor("Time = %t | D = %b | CLK = %b | Q = %b | Qn = %b",

$time, D, CLK, Q, Qn);

end

endmodule



**Q6.**

// UDP for NAND Gate

primitive nand\_gate (out, a, b);

output out;

input a, b;

table

// a b : out

0 0 : 1;

0 1 : 1;

1 0 : 1;

1 1 : 0;

endtable

endprimitive

// UDP for AND Gate

primitive and\_gate (out, a, b);

output out;

input a, b;

table

// a b : out

0 0 : 0;

0 1 : 0;

1 0 : 0;

1 1 : 1;

endtable

endprimitive

// UDP for OR Gate

primitive or\_gate (out, a, b);

output out;

input a, b;

table

// a b : out

0 0 : 0;

0 1 : 1;

1 0 : 1;

1 1 : 1;

endtable

endprimitive

// UDP for NOT Gate

primitive not\_gate (out, a);

output out;

input a;

table

// a : out

0 : 1;

1 : 0;

endtable

endprimitive

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

`include "udp\_definitions.sv" // Include the UDP definitions

module top\_module (

input wire A, B, C, // Inputs

output wire X, Y // Outputs

);

wire nand\_out, and\_out;

// Instantiate the NAND gate (inputs A, C)

nand\_gate U1 (nand\_out, A, C);

// Instantiate the AND gate (inputs B, nand\_out)

and\_gate U2 (and\_out, B, nand\_out);

// Instantiate the OR gate (for X output)

or\_gate U3 (X, and\_out, B);

// Instantiate the NOT gate (for Y output)

not\_gate U4 (Y, nand\_out);

endmodule

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

module top\_module\_tb;

reg A, B, C;

wire X, Y;

// Instantiate the top module

top\_module uut (

.A(A),

.B(B),

.C(C),

.X(X),

.Y(Y)

);

// Apply test vectors

initial begin

// Test Case 1

A = 0; B = 0; C = 0;

#10;

// Test Case 2

A = 0; B = 1; C = 0;

#10;

// Test Case 3

A = 1; B = 0; C = 1;

#10;

// Test Case 4

A = 1; B = 1; C = 0;

#10;

$finish;

end

// Monitor outputs

initial begin

$monitor("Time = %t | A = %b | B = %b | C = %b | X = %b | Y = %b",

$time, A, B, C, X, Y);

end

endmodule

