**1. 1-bit NOT Gate**

**Design (Using Continuous Assignment and always block):**

// Code your design here

module not\_gate\_cont (output Y, input A);

assign Y = ~A;

endmodule

module not\_gate\_always (output reg Y, input A);

always @(\*) begin

Y = ~A;

end

endmodule

**Testbench:**

module not\_gate\_tb;

reg A; // Input

wire Y; // Output

// Instantiate the NOT gate (Continuous or Always)

not\_gate\_cont uut (Y, A); // Change to `not\_gate\_always` if needed

initial begin

// Set up the dump file and variables for waveform

$dumpfile("dump.vcd"); // Specify the name of the VCD file

$dumpvars(0, not\_gate\_tb); // Dump all variables in this module

// Apply test cases

A = 1'b0;

#10 A = 1'b1;

#10 $finish; // End the simulation

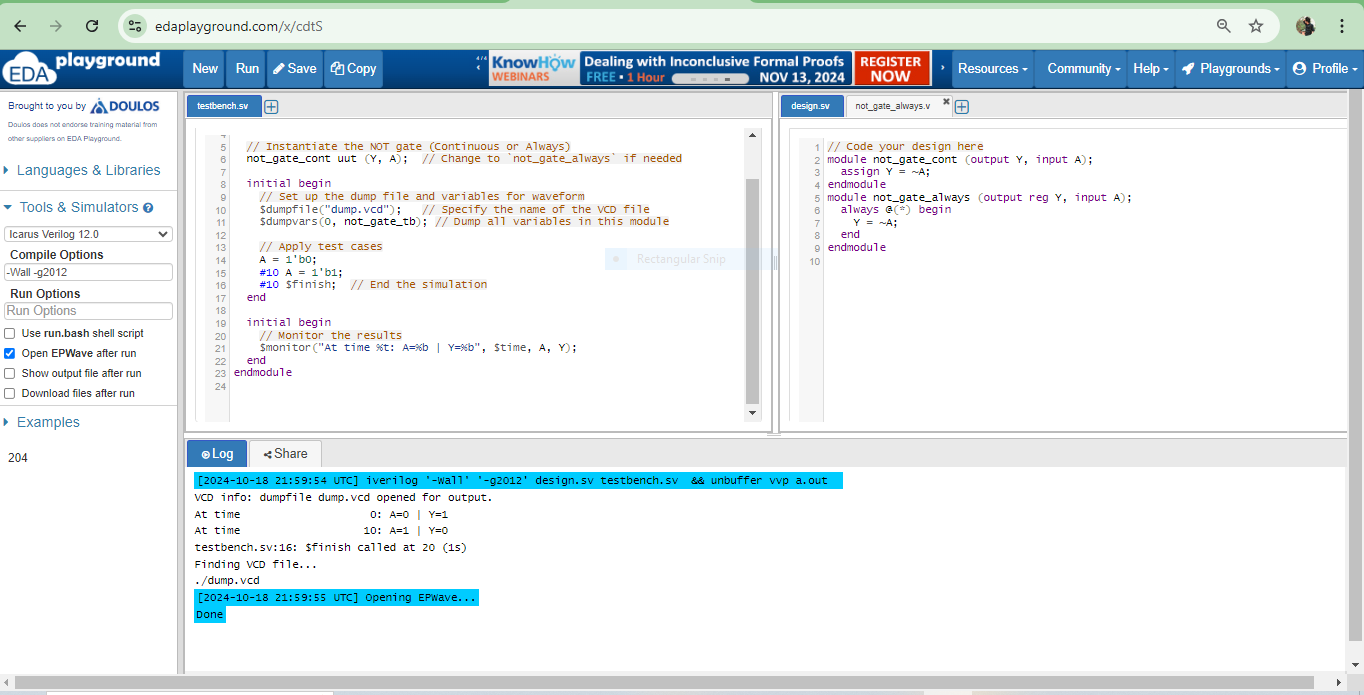
end

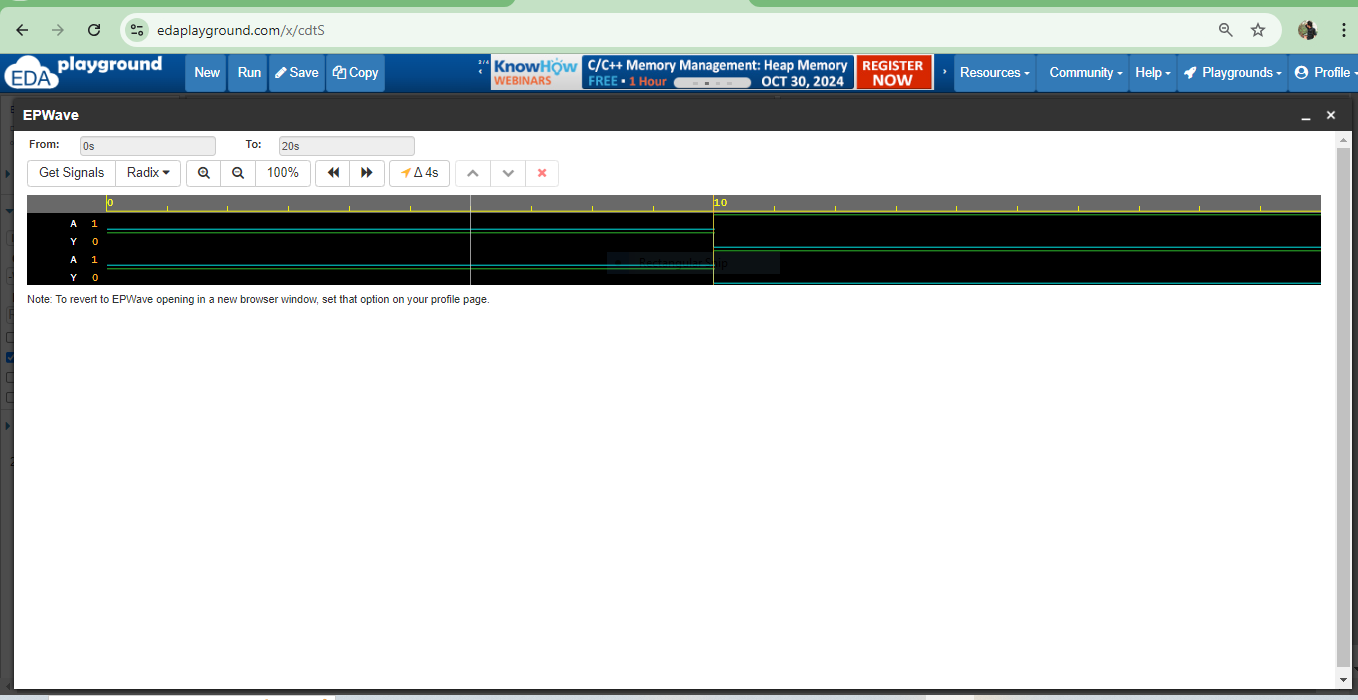
initial begin

// Monitor the results

$monitor("At time %t: A=%b | Y=%b", $time, A, Y);

end

endmodule



### 2. 2-bit AND Gate

**Design (Using Always Block and continuous assignment):**

module and\_gate\_cont (output [1:0] Y, input [1:0] A, B);

assign Y = A & B;

endmodule

module and\_gate\_always (output reg [1:0] Y, input [1:0] A, B);

always @(\*) begin

Y = A & B;

end

endmodule**Testbench:**

module and\_gate\_tb;

reg [1:0] A, B; // Inputs

wire [1:0] Y; // Output

// Instantiate the AND gate (Continuous or Always)

and\_gate\_cont uut (Y, A, B); // Change to `and\_gate\_always` if needed

initial begin

// Set up the dump file and variables for waveform

$dumpfile("and\_gate.vcd");

$dumpvars(0, and\_gate\_tb);

// Apply test cases

A = 2'b00; B = 2'b00;

#10 A = 2'b01; B = 2'b01;

#10 A = 2'b10; B = 2'b10;

#10 A = 2'b11; B = 2'b11;

#10 $finish;

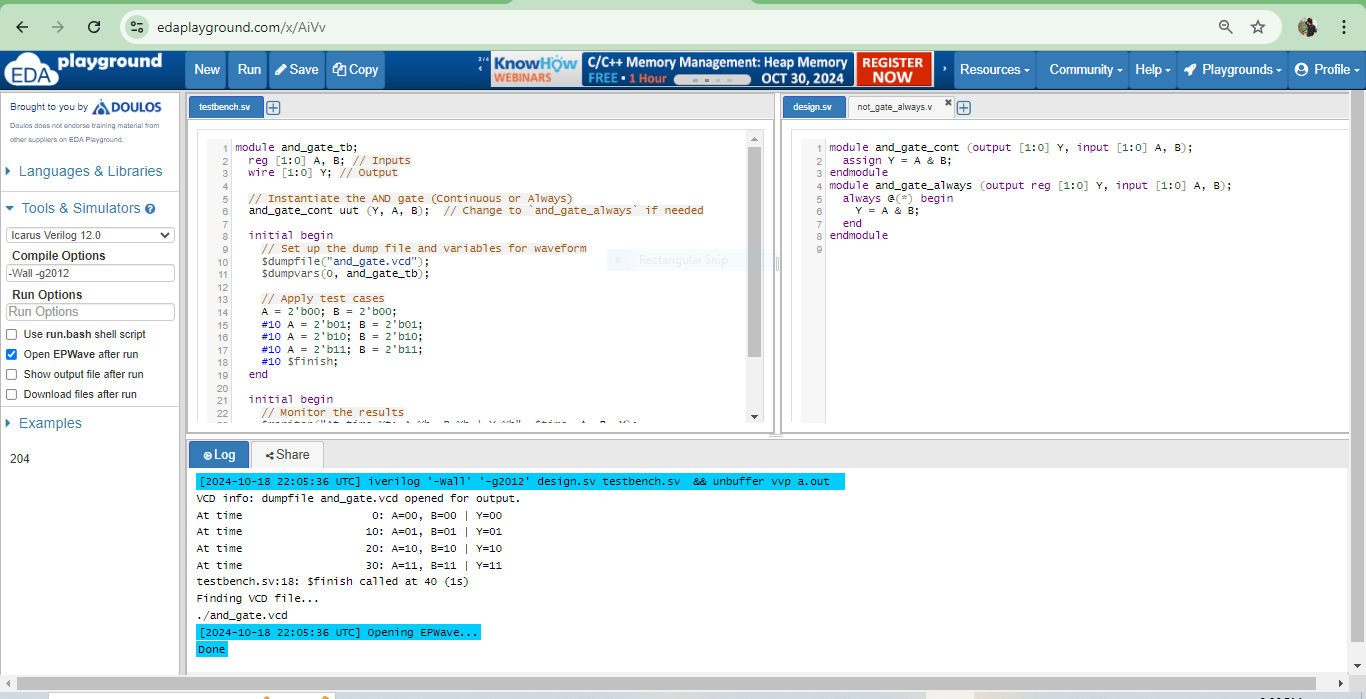
end

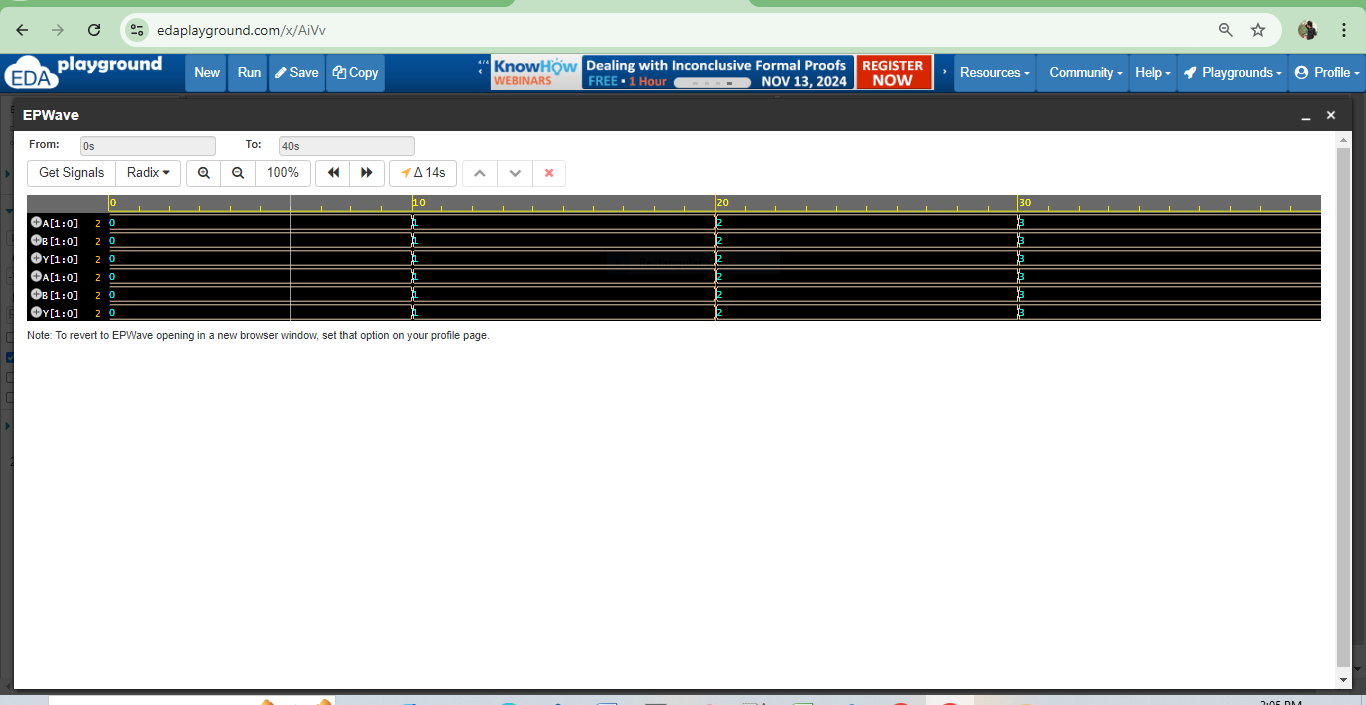
initial begin

// Monitor the results

$monitor("At time %t: A=%b, B=%b | Y=%b", $time, A, B, Y);

end

endmodule



### 3. 2-bit OR Gate

**Design (Using Continuous Assignment and always block):**

module or\_gate\_cont (output [1:0] Y, input [1:0] A, B);

assign Y = A | B;

endmodule

module or\_gate\_always (output reg [1:0] Y, input [1:0] A, B);

always @(\*) begin

Y = A | B;

end

endmodule

**Testbench:**

module or\_gate\_tb;

reg [1:0] A, B; // Inputs

wire [1:0] Y; // Output

// Instantiate the OR gate (Continuous or Always)

or\_gate\_cont uut (Y, A, B); // Change to `or\_gate\_always` if needed

initial begin

// Set up the dump file and variables for waveform

$dumpfile("or\_gate.vcd");

$dumpvars(0, or\_gate\_tb);

// Apply test cases

A = 2'b00; B = 2'b00;

#10 A = 2'b01; B = 2'b01;

#10 A = 2'b10; B = 2'b10;

#10 A = 2'b11; B = 2'b11;

#10 $finish;

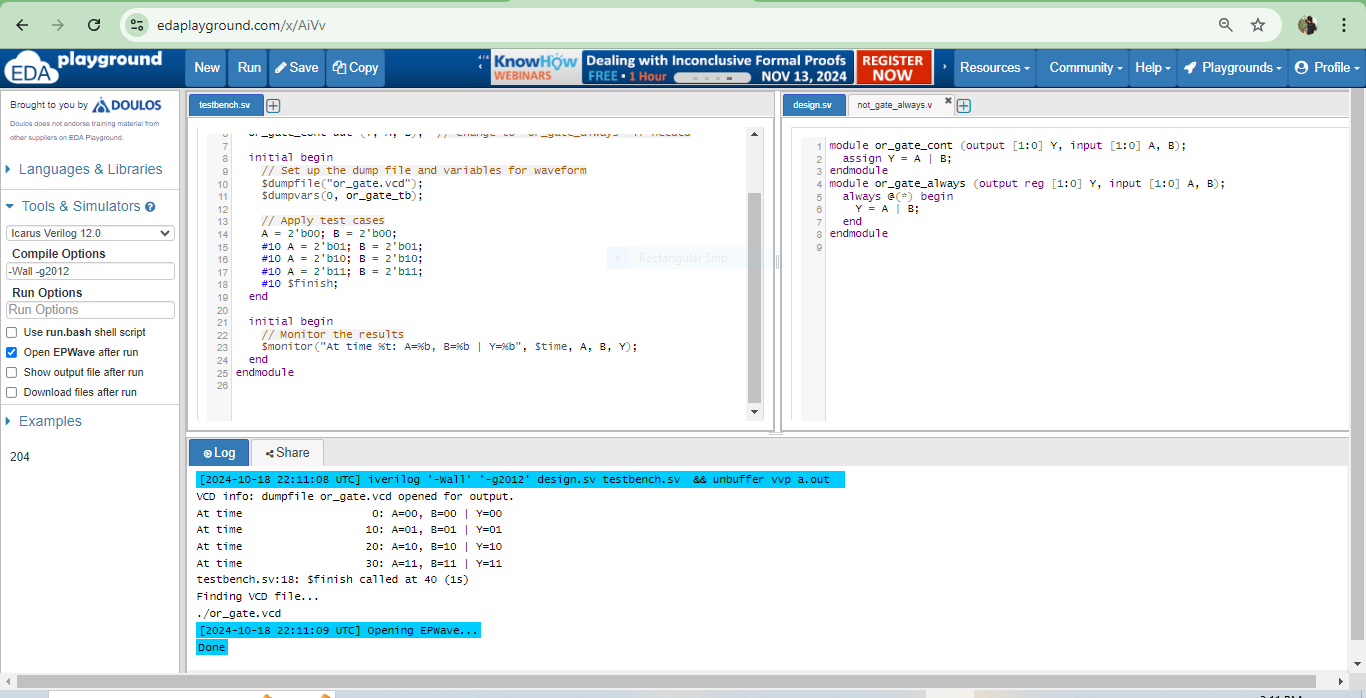
end

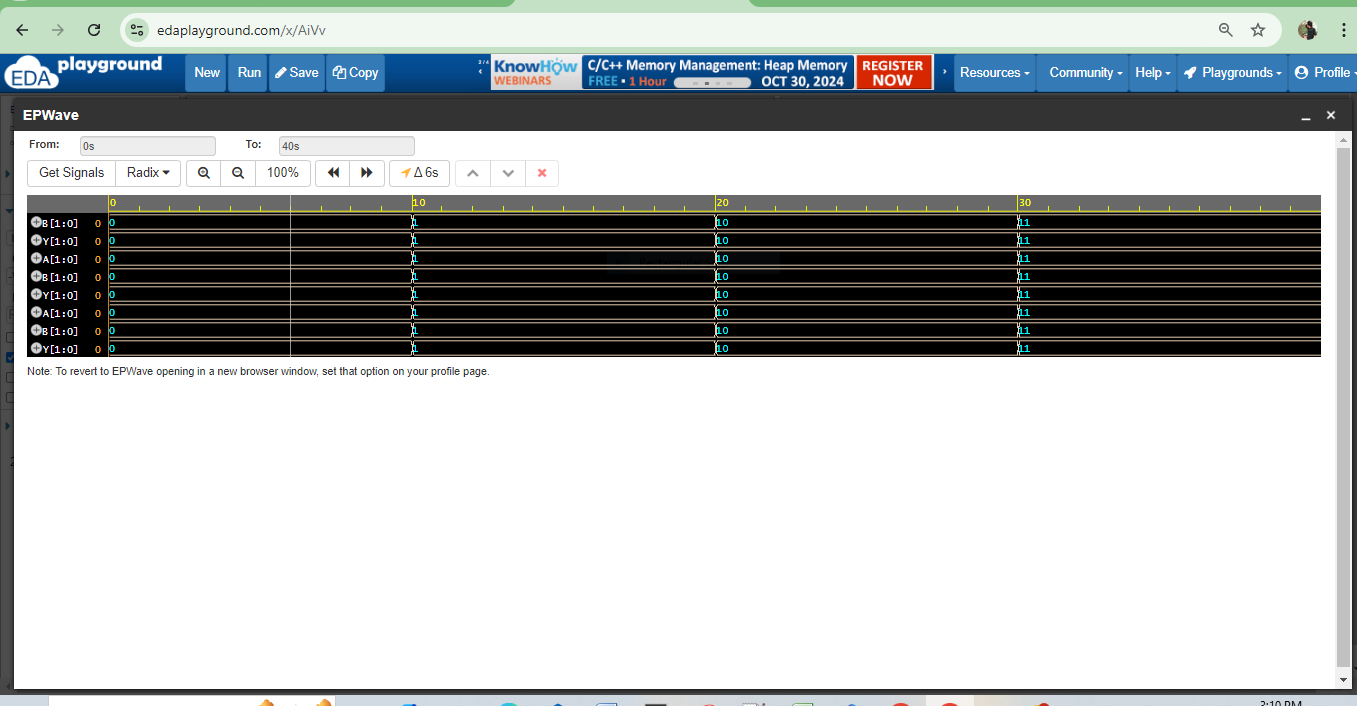
initial begin

// Monitor the results

$monitor("At time %t: A=%b, B=%b | Y=%b", $time, A, B, Y);

end

endmodule



### 4. 2-bit NAND Gate

**Design (Using Always Block and continuous block):**

module nand\_gate\_cont (output [1:0] Y, input [1:0] A, B);

assign Y = ~(A & B);

endmodule

module nand\_gate\_always (output reg [1:0] Y, input [1:0] A, B);

always @(\*) begin

Y = ~(A & B);

end

endmodule**Testbench:**

module nand\_gate\_tb;

reg [1:0] A, B; // Inputs

wire [1:0] Y; // Output

// Instantiate the NAND gate (Continuous or Always)

nand\_gate\_cont uut (Y, A, B); // Change to `nand\_gate\_always` if needed

initial begin

// Set up the dump file and variables for waveform

$dumpfile("nand\_gate.vcd");

$dumpvars(0, nand\_gate\_tb);

// Apply test cases

A = 2'b00; B = 2'b00;

#10 A = 2'b01; B = 2'b01;

#10 A = 2'b10; B = 2'b10;

#10 A = 2'b11; B = 2'b11;

#10 $finish;

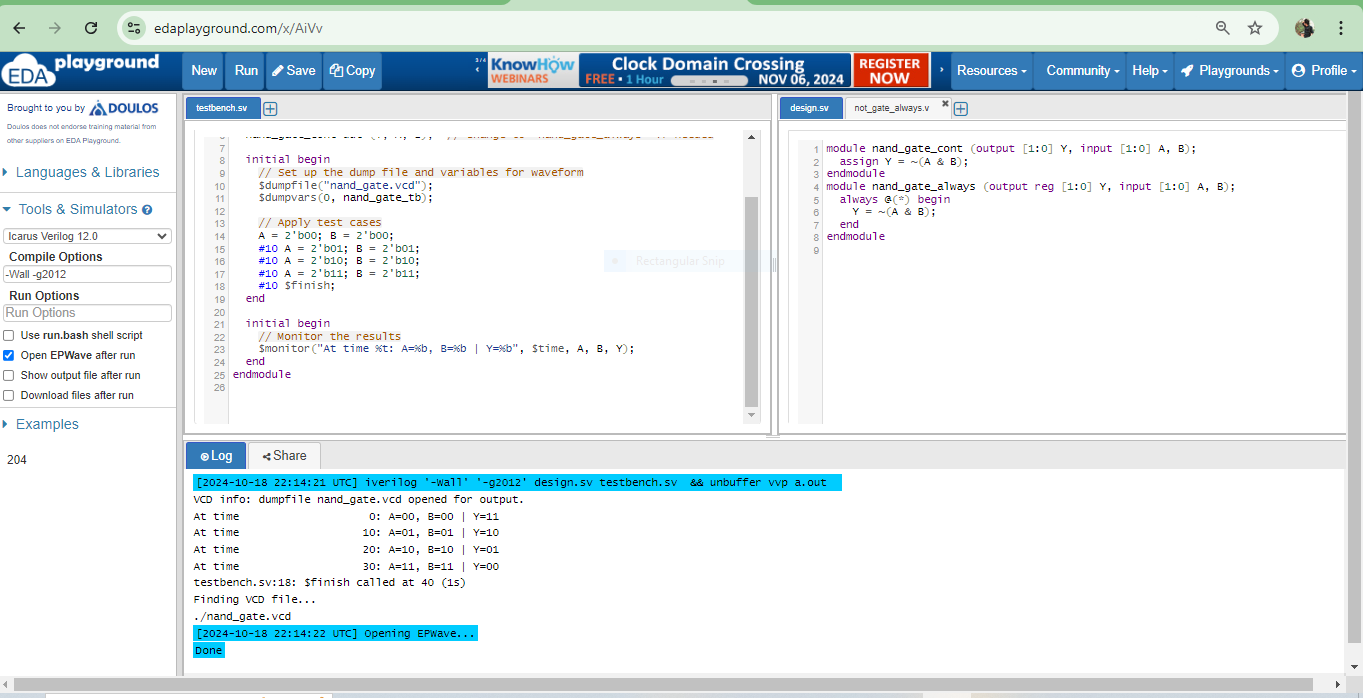
end

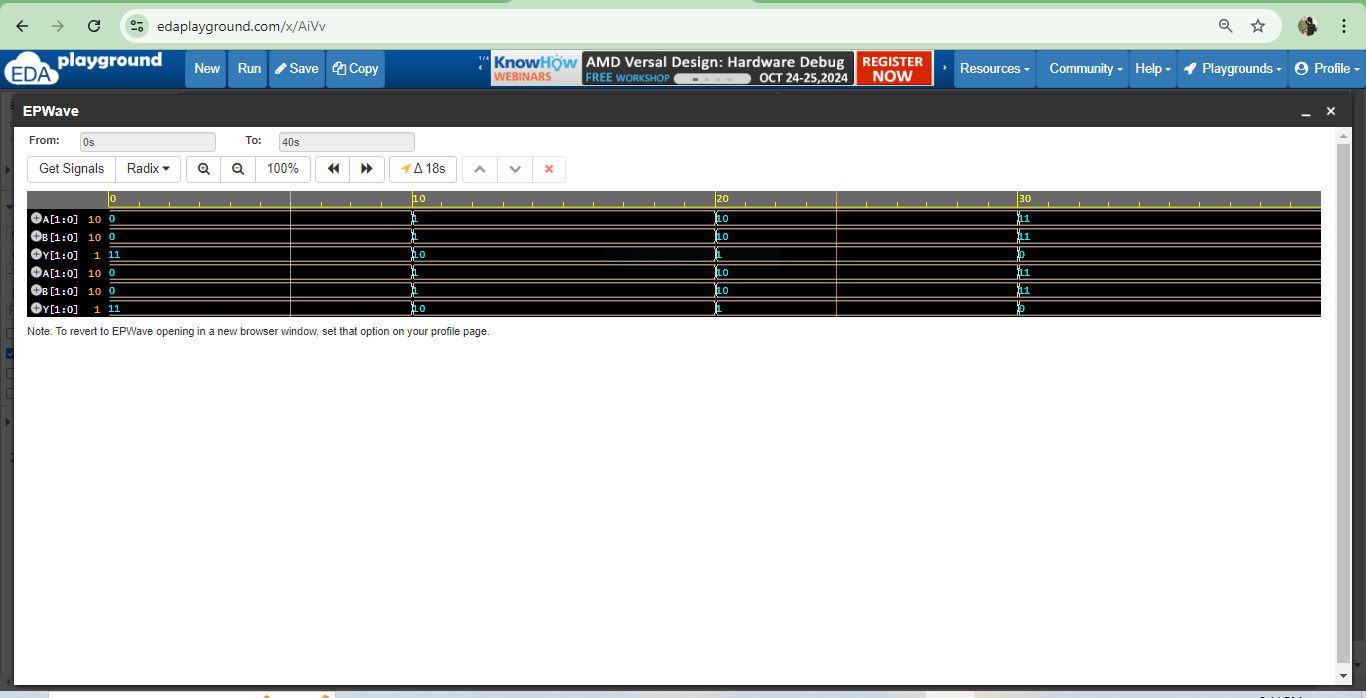
initial begin

// Monitor the results

$monitor("At time %t: A=%b, B=%b | Y=%b", $time, A, B, Y);

end

endmodule



### 5. 2-bit NOR Gate

**Design (Using Continuous Assignment and always block):**

module nor\_gate\_cont (output [1:0] Y, input [1:0] A, B);

assign Y = ~(A | B);

endmodule

module nor\_gate\_always (output reg [1:0] Y, input [1:0] A, B);

always @(\*) begin

Y = ~(A | B);

end

endmodule

**Testbench:**

module nor\_gate\_tb;

reg [1:0] A, B; // Inputs

wire [1:0] Y; // Output

// Instantiate the NOR gate (Continuous or Always)

nor\_gate\_cont uut (Y, A, B); // Change to `nor\_gate\_always` if needed

initial begin

// Set up the dump file and variables for waveform

$dumpfile("nor\_gate.vcd");

$dumpvars(0, nor\_gate\_tb);

// Apply test cases

A = 2'b00; B = 2'b00;

#10 A = 2'b01; B = 2'b01;

#10 A = 2'b10; B = 2'b10;

#10 A = 2'b11; B = 2'b11;

#10 $finish;

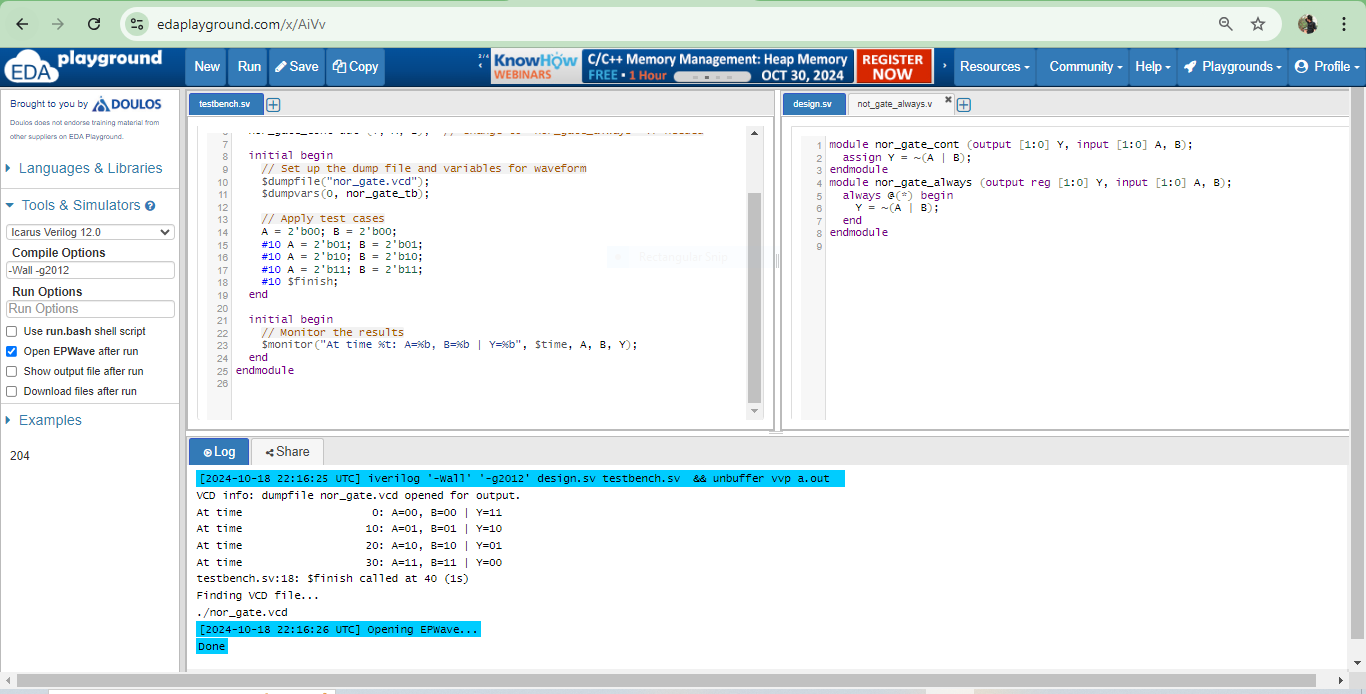
end

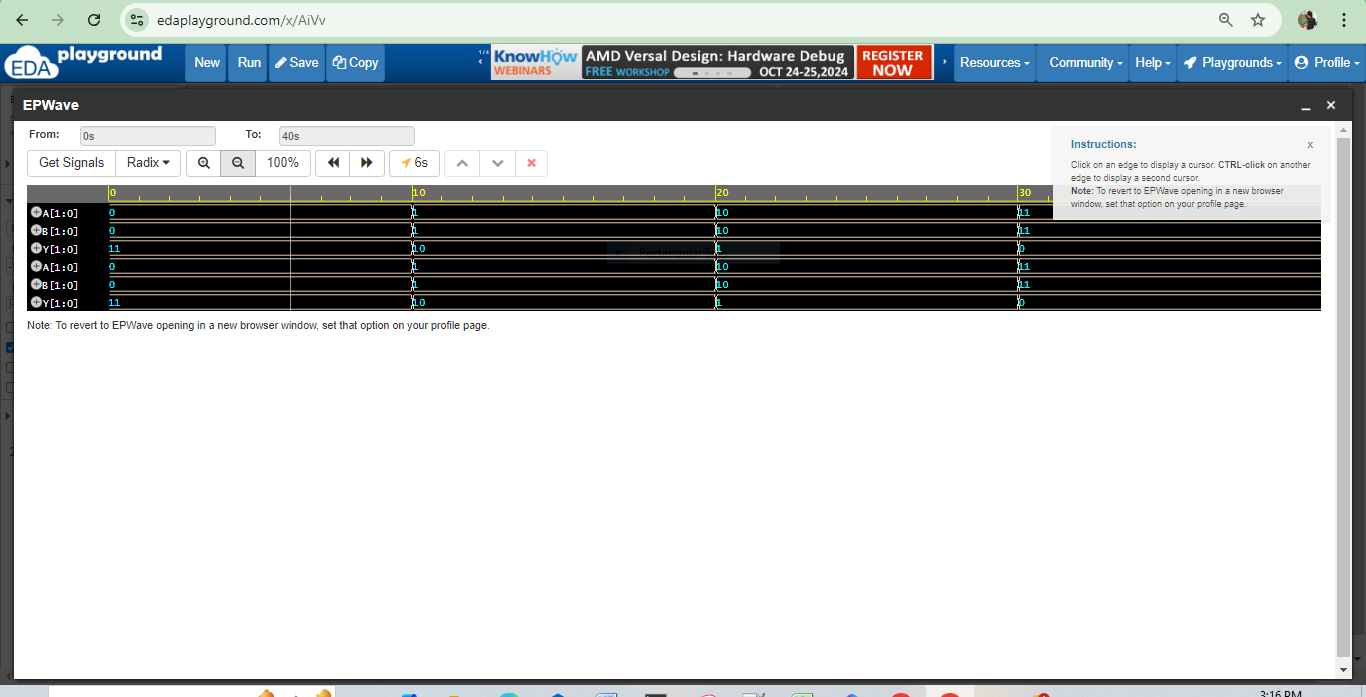
initial begin

// Monitor the results

$monitor("At time %t: A=%b, B=%b | Y=%b", $time, A, B, Y);

end

endmodule



### 6. 2-bit XOR Gate

**Design (Using Always Block and continuous block):**

module xor\_gate\_cont(output [1:0] Y, input [1:0] A, input [1:0] B);

assign Y = A ^ B; // Bitwise XOR operation

endmodule

module xor\_gate\_always(output reg [1:0] Y, input [1:0] A, input [1:0] B);

always @(\*) begin

Y = A ^ B; // Bitwise XOR operation

end

endmodule

**Testbench:**

module xor\_gate\_tb;

reg [1:0] A, B; // Inputs

wire [1:0] Y; // Output

// Instantiate the XOR gate (Continuous or Always)

xor\_gate\_cont uut (Y, A, B); // Use the correct XOR gate implementation

initial begin

// Set up the dump file and variables for waveform

$dumpfile("xor\_gate.vcd");

$dumpvars(0, xor\_gate\_tb);

// Apply test cases

A = 2'b00; B = 2'b00; // Expect Y = 00

#10 A = 2'b01; B = 2'b00; // Expect Y = 01

#10 A = 2'b10; B = 2'b00; // Expect Y = 10

#10 A = 2'b01; B = 2'b11; // Expect Y = 10

#10 A = 2'b11; B = 2'b10; // Expect Y = 01

#10 A = 2'b11; B = 2'b11; // Expect Y = 00

#10 $finish;

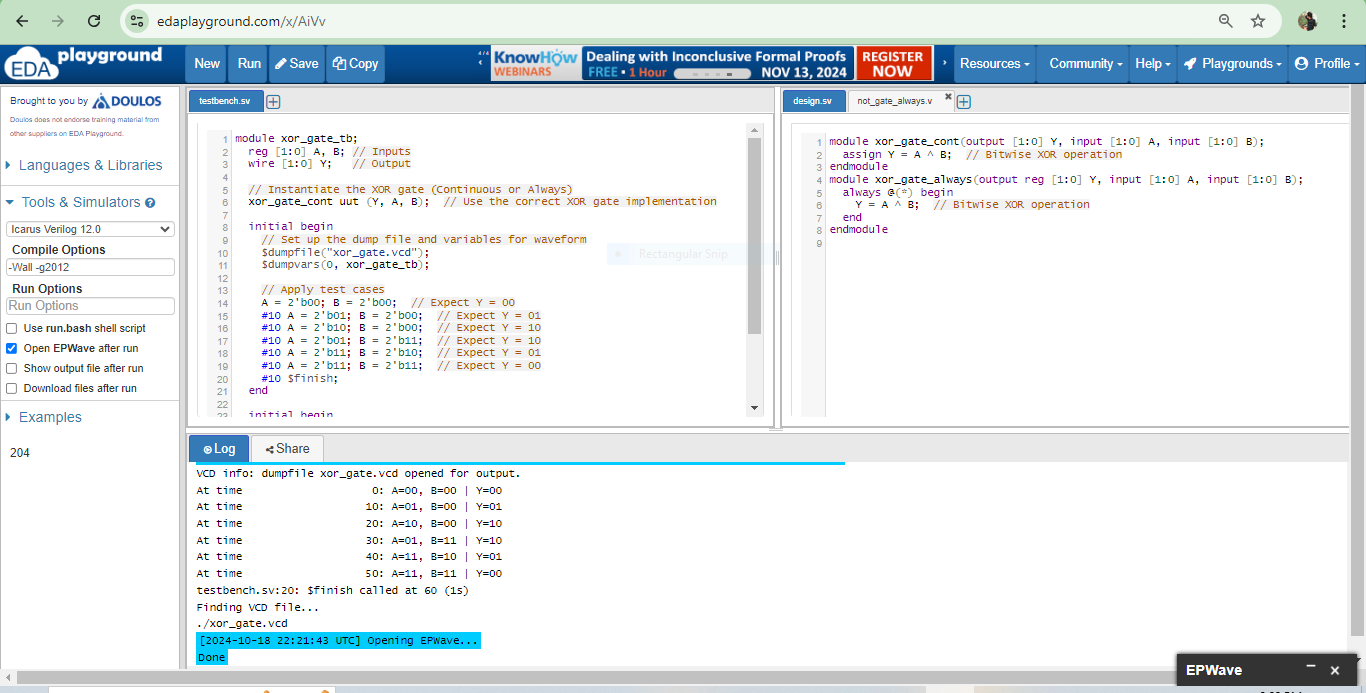
end

initial begin

// Monitor the results

$monitor("At time %t: A=%b, B=%b | Y=%b", $time, A, B, Y);

end

endmodule

### 

### 7. 2-bit XNOR Gate

### **Design (Using Continuous Assignment and always block):**

module xnor\_gate\_cont (output [1:0] Y, input [1:0] A, B);

assign Y = ~(A ^ B);

endmodule

module xnor\_gate\_always (output reg [1:0] Y, input [1:0] A, B);

always @(\*) begin

Y = ~(A ^ B);

end

endmodule

**Testbench:**

module xnor\_gate\_tb;

reg [1:0] A, B; // Inputs

wire [1:0] Y; // Output

// Instantiate the XNOR gate (Continuous or Always)

xnor\_gate\_cont uut (Y, A, B); // Change to `xnor\_gate\_always` if needed

initial begin

// Set up the dump file and variables for waveform

$dumpfile("xnor\_gate.vcd");

$dumpvars(0, xnor\_gate\_tb);

// Apply test cases

A = 2'b00; B = 2'b00;

#10 A = 2'b01; B = 2'b01;

#10 A = 2'b10; B = 2'b10;

#10 A = 2'b11; B = 2'b11;

#10 $finish;

end

initial begin

// Monitor the results

$monitor("At time %t: A=%b, B=%b | Y=%b", $time, A, B, Y);

end

endmodule