Nang Thiri Wutyi

20113

10/21/2024

**Verilog Lab #4**

1. **1-bit NOT gate**

**Design module**

`timescale 1ns / 1ps // Specify time unit of 1 nanosecond and time precision of 1 picosecond

// 1-bit NOT gate using continuous assignment

module not\_gate\_cont (

input wire a, // 1-bit input

output wire y // 1-bit output

);

assign y = ~a; // Continuous assignment of NOT operation

endmodule

// 1-bit NOT gate using always block

module not\_gate\_always (

input wire a, // 1-bit input

output reg y // 1-bit output (reg type for always block)

);

always @(\*) begin

y = ~a; // Assigning NOT operation inside always block

end

endmodule

**Testbench module**

`timescale 1ns / 1ps

module tb\_not\_gate();

// Declare inputs for the testbench

reg a;

// Declare outputs for both modules

wire y\_cont;

wire y\_always;

// Instantiate the not\_gate\_cont module

not\_gate\_cont u1 (

.a(a),

.y(y\_cont)

);

// Instantiate the not\_gate\_always module

not\_gate\_always u2 (

.a(a),

.y(y\_always)

);

// Testbench initial block

initial begin

// Enable VCD dumping for waveform generation

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

$dumpvars(0, tb\_not\_gate); // Dump all variables in tb\_not\_gate

// Display results in the console

$monitor("Time=%0t | a=%b | y\_cont=%b | y\_always=%b", $time, a, y\_cont, y\_always);

// Test case 1: a = 0

a = 1'b0;

#10;

// Test case 2: a = 1

a = 1'b1;

#10;

// Finish simulation

$finish;

end

endmodule

**Results**

**[2024-10-22 02:44:57 UTC] iverilog '-Wall' '-g2012' design.sv testbench.sv && unbuffer vvp a.out**

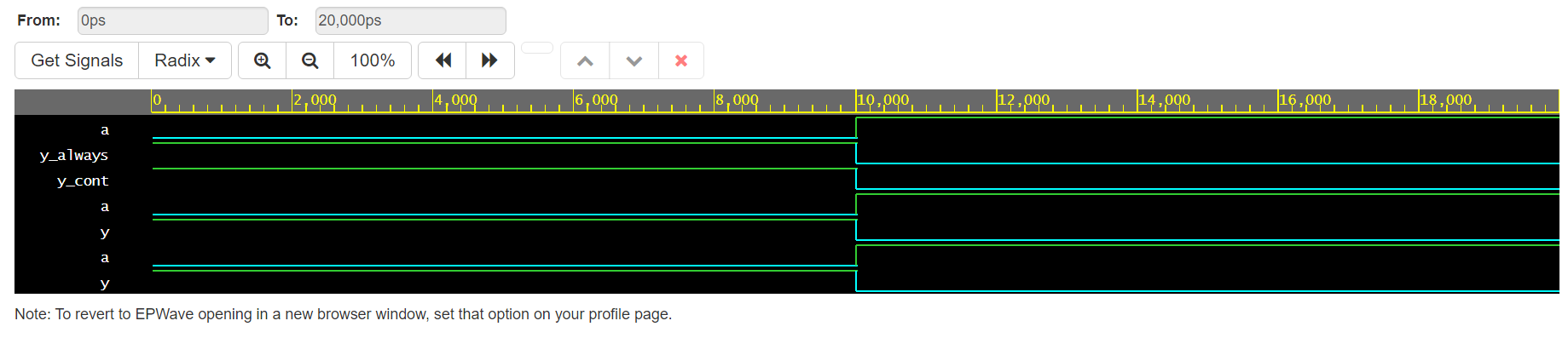
**VCD info: dumpfile dump.vcd opened for output.**

**Time=0 | a=0 | y\_cont=1 | y\_always=1**

**Time=10000 | a=1 | y\_cont=0 | y\_always=0**

**testbench.sv:41: $finish called at 20000 (1ps)**

**Done**



1. **2-bits AND gate**

**Design module**

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

// 2-bit AND gate using continuous assignment

module and\_gate\_cont (

input wire [1:0] a, // 2-bit input

input wire [1:0] b, // 2-bit input

output wire [1:0] y // 2-bit output

);

assign y = a & b; // Continuous assignment for AND operation

endmodule

// 2-bit AND gate using always block

module and\_gate\_always (

input wire [1:0] a, // 2-bit input

input wire [1:0] b, // 2-bit input

output reg [1:0] y // 2-bit output (reg type for always block)

);

always @(\*) begin

y = a & b; // AND operation inside always block

end

endmodule

**Testbench module**

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

module tb\_and\_gate();

// Declare inputs for the testbench

reg [1:0] a;

reg [1:0] b;

// Declare outputs for both modules

wire [1:0] y\_cont;

wire [1:0] y\_always;

// Instantiate the and\_gate\_cont module

and\_gate\_cont u1 (

.a(a),

.b(b),

.y(y\_cont)

);

// Instantiate the and\_gate\_always module

and\_gate\_always u2 (

.a(a),

.b(b),

.y(y\_always)

);

// Testbench initial block

initial begin

// Enable VCD dumping for waveform generation

$dumpfile("dump.vcd");

$dumpvars(0, tb\_and\_gate);

// Display results in the console

$monitor("Time=%0t | a=%b | b=%b | y\_cont=%b | y\_always=%b", $time, a, b, y\_cont, y\_always);

// Test case 1: a = 00, b = 00

a = 2'b00; b = 2'b00;

#10;

// Test case 2: a = 01, b = 10

a = 2'b01; b = 2'b10;

#10;

// Test case 3: a = 11, b = 11

a = 2'b11; b = 2'b11;

#10;

// Test case 4: a = 10, b = 01

a = 2'b10; b = 2'b01;

#10;

// Finish simulation

$finish;

end

endmodule

**Results**

**[2024-10-22 02:53:33 UTC] iverilog '-Wall' '-g2012' design.sv testbench.sv && unbuffer vvp a.out**

**VCD info: dumpfile dump.vcd opened for output.**

**Time=0 | a=00 | b=00 | y\_cont=00 | y\_always=00**

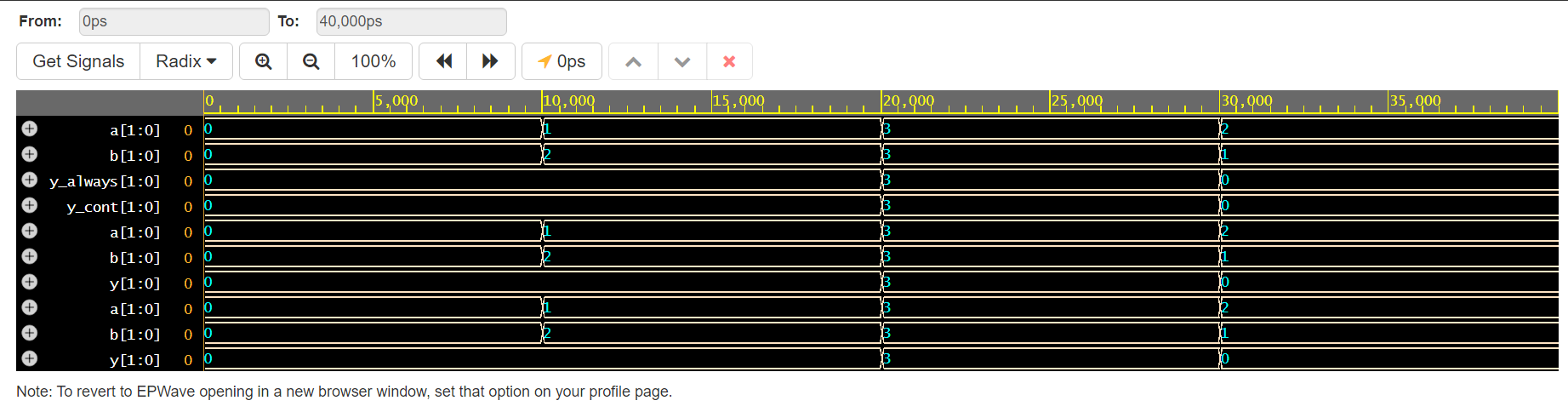
**Time=10000 | a=01 | b=10 | y\_cont=00 | y\_always=00**

**Time=20000 | a=11 | b=11 | y\_cont=11 | y\_always=11**

**Time=30000 | a=10 | b=01 | y\_cont=00 | y\_always=00**

**testbench.sv:52: $finish called at 40000 (1ps)**

**Done**

1. **2-bits OR gate**

**Design module**

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

module tb\_or\_gate();

// Declare inputs for the testbench

reg [1:0] a;

reg [1:0] b;

// Declare outputs for both modules

wire [1:0] y\_cont;

wire [1:0] y\_always;

// Instantiate the or\_gate\_cont module

or\_gate\_cont u1 (

.a(a),

.b(b),

.y(y\_cont)

);

// Instantiate the or\_gate\_always module

or\_gate\_always u2 (

.a(a),

.b(b),

.y(y\_always)

);

// Testbench initial block

initial begin

// Enable VCD dumping for waveform generation

$dumpfile("dump.vcd");

$dumpvars(0, tb\_or\_gate);

// Display results in the console

$monitor("Time=%0t | a=%b | b=%b | y\_cont=%b | y\_always=%b", $time, a, b, y\_cont, y\_always);

// Test case 1: a = 00, b = 00

a = 2'b00; b = 2'b00;

#10;

// Test case 2: a = 01, b = 10

a = 2'b01; b = 2'b10;

#10;

// Test case 3: a = 11, b = 11

a = 2'b11; b = 2'b11;

#10;

// Test case 4: a = 10, b = 01

a = 2'b10; b = 2'b01;

#10;

// Finish simulation

$finish;

end

endmodule

**Testbench module**

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

module tb\_or\_gate();

// Declare inputs for the testbench

reg [1:0] a;

reg [1:0] b;

// Declare outputs for both modules

wire [1:0] y\_cont;

wire [1:0] y\_always;

// Instantiate the or\_gate\_cont module

or\_gate\_cont u1 (

.a(a),

.b(b),

.y(y\_cont)

);

// Instantiate the or\_gate\_always module

or\_gate\_always u2 (

.a(a),

.b(b),

.y(y\_always)

);

// Testbench initial block

initial begin

// Enable VCD dumping for waveform generation

$dumpfile("dump.vcd");

$dumpvars(0, tb\_or\_gate);

// Display results in the console

$monitor("Time=%0t | a=%b | b=%b | y\_cont=%b | y\_always=%b", $time, a, b, y\_cont, y\_always);

// Test case 1: a = 00, b = 00

a = 2'b00; b = 2'b00;

#10;

// Test case 2: a = 01, b = 10

a = 2'b01; b = 2'b10;

#10;

// Test case 3: a = 11, b = 11

a = 2'b11; b = 2'b11;

#10;

// Test case 4: a = 10, b = 01

a = 2'b10; b = 2'b01;

#10;

// Finish simulation

$finish;

end

endmodule

**Results**

**[2024-10-22 03:09:38 UTC] iverilog '-Wall' '-g2012' design.sv testbench.sv && unbuffer vvp a.out**

**VCD info: dumpfile dump.vcd opened for output.**

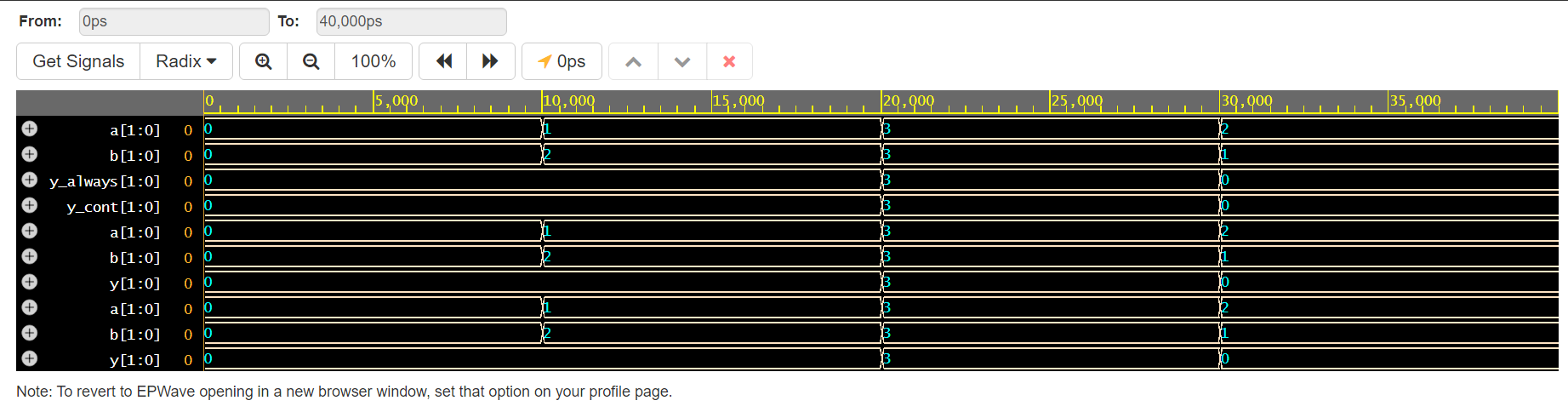
**Time=0 | a=00 | b=00 | y\_cont=00 | y\_always=00**

**Time=10000 | a=01 | b=10 | y\_cont=11 | y\_always=11**

**Time=20000 | a=11 | b=11 | y\_cont=11 | y\_always=11**

**Time=30000 | a=10 | b=01 | y\_cont=11 | y\_always=11**

**testbench.sv:52: $finish called at 40000 (1ps)**

**Done**

1. **2-bits NAND gate**

**Design module**

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

// 2-bit NAND gate using continuous assignment

module nand\_gate\_cont (

input wire [1:0] a, // 2-bit input

input wire [1:0] b, // 2-bit input

output wire [1:0] y // 2-bit output

);

assign y = ~(a & b); // Continuous assignment for NAND operation

endmodule

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

// 2-bit NAND gate using always block

module nand\_gate\_always (

input wire [1:0] a, // 2-bit input

input wire [1:0] b, // 2-bit input

output reg [1:0] y // 2-bit output (reg type for always block)

);

always @(\*) begin

y = ~(a & b); // NAND operation inside always block

end

endmodule

**Testbench module**

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

module tb\_nand\_gate();

// Declare inputs for the testbench

reg [1:0] a;

reg [1:0] b;

// Declare outputs for both modules

wire [1:0] y\_cont;

wire [1:0] y\_always;

// Instantiate the nand\_gate\_cont module

nand\_gate\_cont u1 (

.a(a),

.b(b),

.y(y\_cont)

);

// Instantiate the nand\_gate\_always module

nand\_gate\_always u2 (

.a(a),

.b(b),

.y(y\_always)

);

// Testbench initial block

initial begin

// Enable VCD dumping for waveform generation

$dumpfile("dump.vcd");

$dumpvars(0, tb\_nand\_gate);

// Display results in the console

$monitor("Time=%0t | a=%b | b=%b | y\_cont=%b | y\_always=%b", $time, a, b, y\_cont, y\_always);

// Test case 1: a = 00, b = 00

a = 2'b00; b = 2'b00;

#10;

// Test case 2: a = 01, b = 10

a = 2'b01; b = 2'b10;

#10;

// Test case 3: a = 11, b = 11

a = 2'b11; b = 2'b11;

#10;

// Test case 4: a = 10, b = 01

a = 2'b10; b = 2'b01;

#10;

// Finish simulation

$finish;

end

endmodule

**Results**

**[2024-10-22 03:18:51 UTC] iverilog '-Wall' '-g2012' design.sv testbench.sv && unbuffer vvp a.out**

**VCD info: dumpfile dump.vcd opened for output.**

**Time=0 | a=00 | b=00 | y\_cont=11 | y\_always=11**

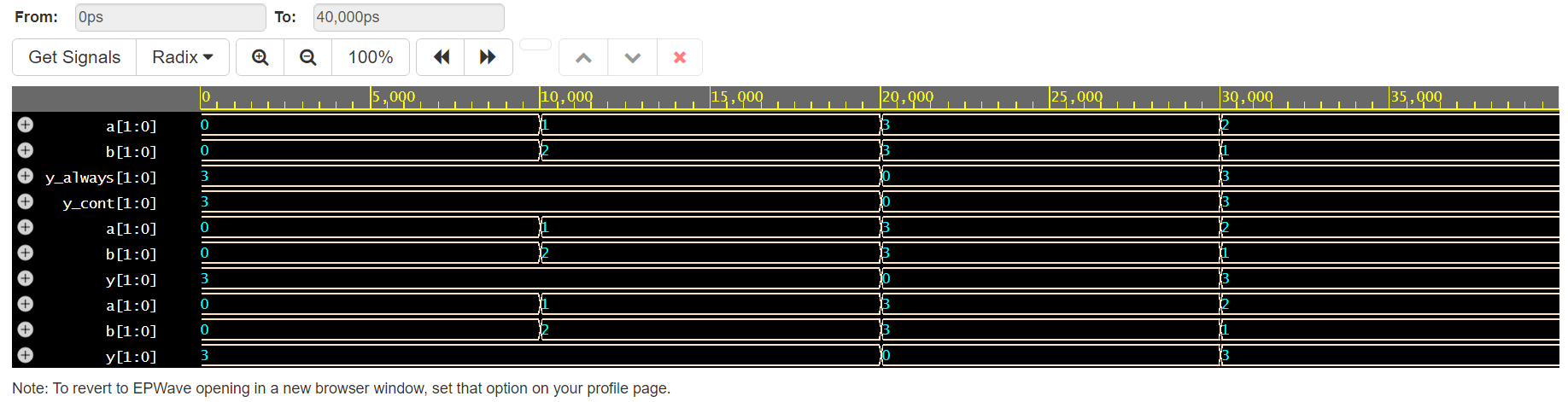
**Time=10000 | a=01 | b=10 | y\_cont=11 | y\_always=11**

**Time=20000 | a=11 | b=11 | y\_cont=00 | y\_always=00**

**Time=30000 | a=10 | b=01 | y\_cont=11 | y\_always=11**

**testbench.sv:52: $finish called at 40000 (1ps)**

**Done**

1. **2-bits NOR gate**

**Design module**

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

module tb\_nor\_gate();

// Declare inputs for the testbench

reg [1:0] a;

reg [1:0] b;

// Declare outputs for both modules

wire [1:0] y\_cont;

wire [1:0] y\_always;

// Instantiate the nor\_gate\_cont module

nor\_gate\_cont u1 (

.a(a),

.b(b),

.y(y\_cont)

);

// Instantiate the nor\_gate\_always module

nor\_gate\_always u2 (

.a(a),

.b(b),

.y(y\_always)

);

// Testbench initial block

initial begin

// Enable VCD dumping for waveform generation

$dumpfile("dump.vcd");

$dumpvars(0, tb\_nor\_gate);

// Display results in the console

$monitor("Time=%0t | a=%b | b=%b | y\_cont=%b | y\_always=%b", $time, a, b, y\_cont, y\_always);

// Test case 1: a = 00, b = 00

a = 2'b00; b = 2'b00;

#10;

// Test case 2: a = 01, b = 10

a = 2'b01; b = 2'b10;

#10;

// Test case 3: a = 11, b = 11

a = 2'b11; b = 2'b11;

#10;

// Test case 4: a = 10, b = 01

a = 2'b10; b = 2'b01;

#10;

// Finish simulation

$finish;

end

endmodule

**Testbench module**

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

module tb\_nor\_gate();

// Declare inputs for the testbench

reg [1:0] a;

reg [1:0] b;

// Declare outputs for both modules

wire [1:0] y\_cont;

wire [1:0] y\_always;

// Instantiate the nor\_gate\_cont module

nor\_gate\_cont u1 (

.a(a),

.b(b),

.y(y\_cont)

);

// Instantiate the nor\_gate\_always module

nor\_gate\_always u2 (

.a(a),

.b(b),

.y(y\_always)

);

// Testbench initial block

initial begin

// Enable VCD dumping for waveform generation

$dumpfile("dump.vcd");

$dumpvars(0, tb\_nor\_gate);

// Display results in the console

$monitor("Time=%0t | a=%b | b=%b | y\_cont=%b | y\_always=%b", $time, a, b, y\_cont, y\_always);

// Test case 1: a = 00, b = 00

a = 2'b00; b = 2'b00;

#10;

// Test case 2: a = 01, b = 10

a = 2'b01; b = 2'b10;

#10;

// Test case 3: a = 11, b = 11

a = 2'b11; b = 2'b11;

#10;

// Test case 4: a = 10, b = 01

a = 2'b10; b = 2'b01;

#10;

// Finish simulation

$finish;

end

endmodule

**Results**

**[2024-10-22 03:25:56 UTC] iverilog '-Wall' '-g2012' design.sv testbench.sv && unbuffer vvp a.out**

**VCD info: dumpfile dump.vcd opened for output.**

**Time=0 | a=00 | b=00 | y\_cont=11 | y\_always=11**

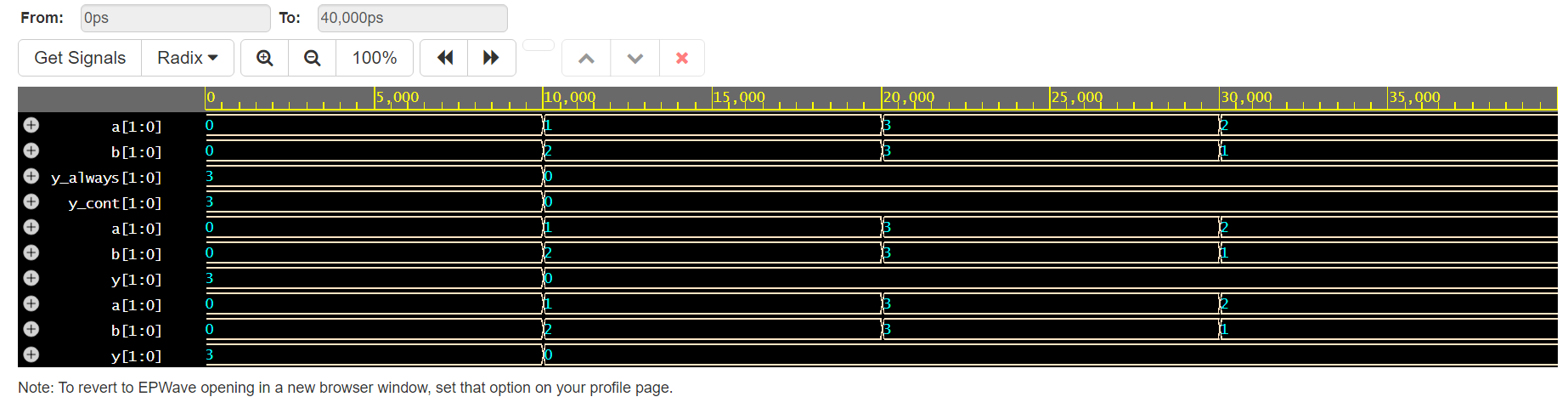
**Time=10000 | a=01 | b=10 | y\_cont=00 | y\_always=00**

**Time=20000 | a=11 | b=11 | y\_cont=00 | y\_always=00**

**Time=30000 | a=10 | b=01 | y\_cont=00 | y\_always=00**

**testbench.sv:52: $finish called at 40000 (1ps)**

**Done**

1. **2-bits XOR gate**

**Design module**

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

// 2-bit XOR gate using continuous assignment

module xor\_gate\_cont (

input wire [1:0] a, // 2-bit input

input wire [1:0] b, // 2-bit input

output wire [1:0] y // 2-bit output

);

assign y = a ^ b; // Continuous assignment for XOR operation

endmodule

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

// 2-bit XOR gate using always block

module xor\_gate\_always (

input wire [1:0] a, // 2-bit input

input wire [1:0] b, // 2-bit input

output reg [1:0] y // 2-bit output (reg type for always block)

);

always @(\*) begin

y = a ^ b; // XOR operation inside always block

end

endmodule

**Testbench module**

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

module tb\_xor\_gate();

// Declare inputs for the testbench

reg [1:0] a;

reg [1:0] b;

// Declare outputs for both modules

wire [1:0] y\_cont;

wire [1:0] y\_always;

// Instantiate the xor\_gate\_cont module

xor\_gate\_cont u1 (

.a(a),

.b(b),

.y(y\_cont)

);

// Instantiate the xor\_gate\_always module

xor\_gate\_always u2 (

.a(a),

.b(b),

.y(y\_always)

);

// Testbench initial block

initial begin

// Enable VCD dumping for waveform generation

$dumpfile("dump.vcd");

$dumpvars(0, tb\_xor\_gate);

// Display results in the console

$monitor("Time=%0t | a=%b | b=%b | y\_cont=%b | y\_always=%b", $time, a, b, y\_cont, y\_always);

// Test case 1: a = 00, b = 00

a = 2'b00; b = 2'b00;

#10;

// Test case 2: a = 01, b = 10

a = 2'b01; b = 2'b10;

#10;

// Test case 3: a = 11, b = 11

a = 2'b11; b = 2'b11;

#10;

// Test case 4: a = 10, b = 01

a = 2'b10; b = 2'b01;

#10;

// Finish simulation

$finish;

end

endmodule

**Results**

**[2024-10-22 03:31:23 UTC] iverilog '-Wall' '-g2012' design.sv testbench.sv && unbuffer vvp a.out**

**VCD info: dumpfile dump.vcd opened for output.**

**Time=0 | a=00 | b=00 | y\_cont=00 | y\_always=00**

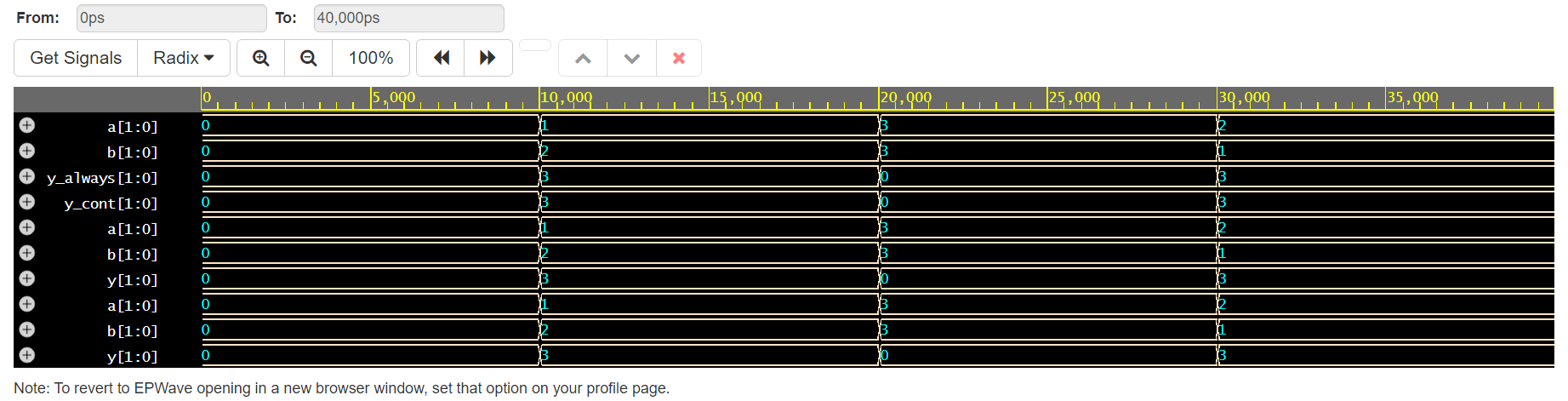
**Time=10000 | a=01 | b=10 | y\_cont=11 | y\_always=11**

**Time=20000 | a=11 | b=11 | y\_cont=00 | y\_always=00**

**Time=30000 | a=10 | b=01 | y\_cont=11 | y\_always=11**

**testbench.sv:52: $finish called at 40000 (1ps)**

**Done**



1. **2-bits XNOR gate**

**Design module**

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

// 2-bit XNOR gate using continuous assignment

module xnor\_gate\_cont (

input wire [1:0] a, // 2-bit input

input wire [1:0] b, // 2-bit input

output wire [1:0] y // 2-bit output

);

assign y = ~(a ^ b); // Continuous assignment for XNOR operation

endmodule

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

// 2-bit XNOR gate using always block

module xnor\_gate\_always (

input wire [1:0] a, // 2-bit input

input wire [1:0] b, // 2-bit input

output reg [1:0] y // 2-bit output (reg type for always block)

);

always @(\*) begin

y = ~(a ^ b); // XNOR operation inside always block

end

endmodule

**Testbench module**

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

module tb\_xnor\_gate();

// Declare inputs for the testbench

reg [1:0] a;

reg [1:0] b;

// Declare outputs for both modules

wire [1:0] y\_cont;

wire [1:0] y\_always;

// Instantiate the xnor\_gate\_cont module

xnor\_gate\_cont u1 (

.a(a),

.b(b),

.y(y\_cont)

);

// Instantiate the xnor\_gate\_always module

xnor\_gate\_always u2 (

.a(a),

.b(b),

.y(y\_always)

);

// Testbench initial block

initial begin

// Enable VCD dumping for waveform generation

$dumpfile("dump.vcd");

$dumpvars(0, tb\_xnor\_gate);

// Display results in the console

$monitor("Time=%0t | a=%b | b=%b | y\_cont=%b | y\_always=%b", $time, a, b, y\_cont, y\_always);

// Test case 1: a = 00, b = 00

a = 2'b00; b = 2'b00;

#10;

// Test case 2: a = 01, b = 10

a = 2'b01; b = 2'b10;

#10;

// Test case 3: a = 11, b = 11

a = 2'b11; b = 2'b11;

#10;

// Test case 4: a = 10, b = 01

a = 2'b10; b = 2'b01;

#10;

// Finish simulation

$finish;

end

endmodule

**Results**

**[2024-10-22 03:36:02 UTC] iverilog '-Wall' '-g2012' design.sv testbench.sv && unbuffer vvp a.out**

**VCD info: dumpfile dump.vcd opened for output.**

**Time=0 | a=00 | b=00 | y\_cont=11 | y\_always=11**

**Time=10000 | a=01 | b=10 | y\_cont=00 | y\_always=00**

**Time=20000 | a=11 | b=11 | y\_cont=11 | y\_always=11**

**Time=30000 | a=10 | b=01 | y\_cont=00 | y\_always=00**

**testbench.sv:52: $finish called at 40000 (1ps)**

**Done**

