Nang Thiri Wutyi
20113
10/21/2024

Verilog Lab #4

1. **1-bit NOT gate**

Design module

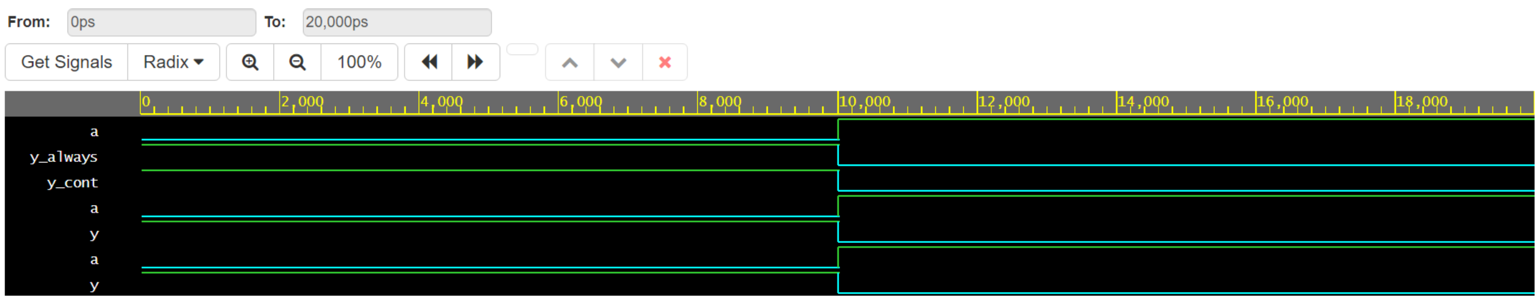
```
`timescale 1ns / 1ps  // Specify time unit of 1 nanosecond and time precision of 1
picosecond

// 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
```



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

2. 2-bits AND gate

Design module

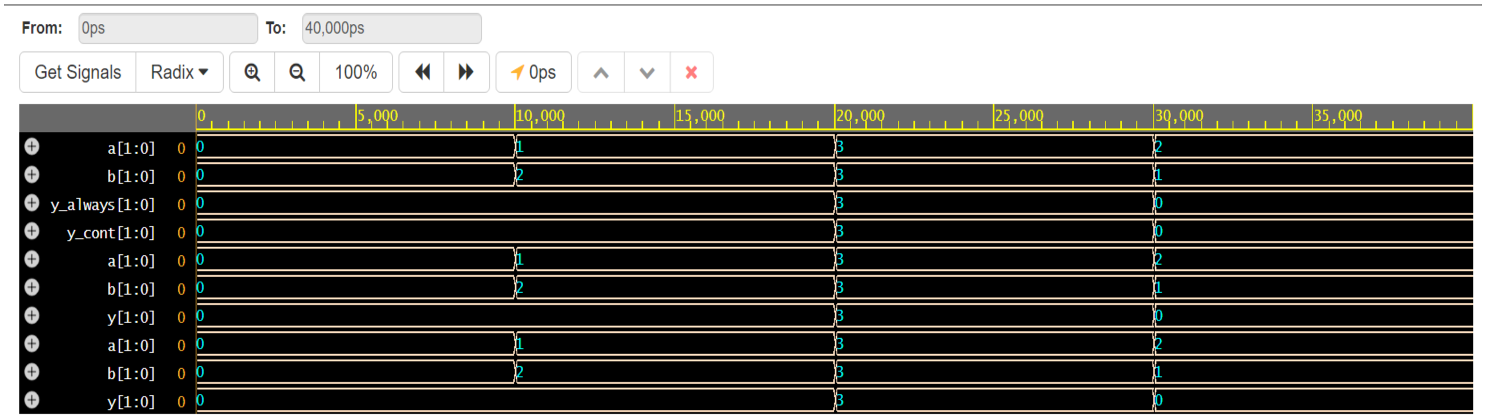
```
`timescale 1ns / 1ps  // Time unit is 1ns, and time precision is 1ps

// 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
```



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

3. **2-bits OR gate**

Design module

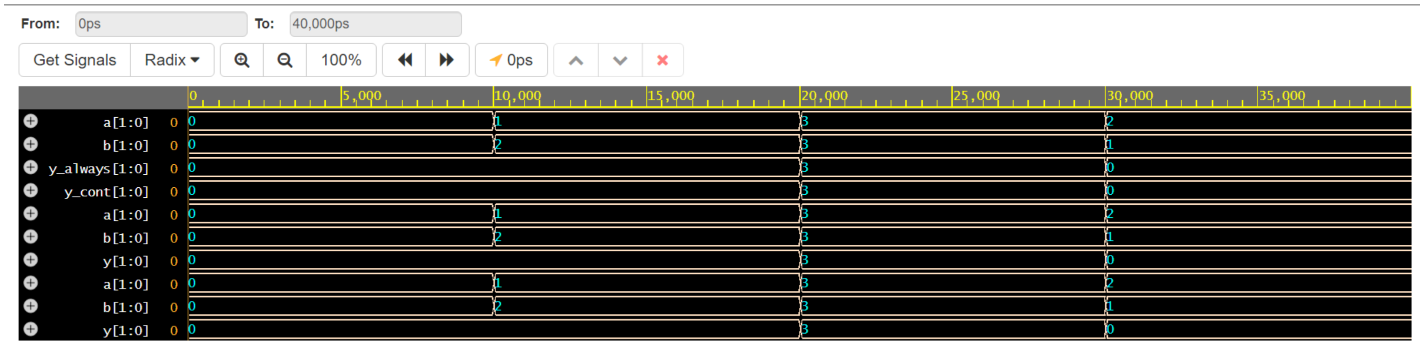
```
`timescale 1ns / 1ps  // Time unit is 1ns, and time precision is 1ps

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
```



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

4. **2-bits NAND gate**

Design module

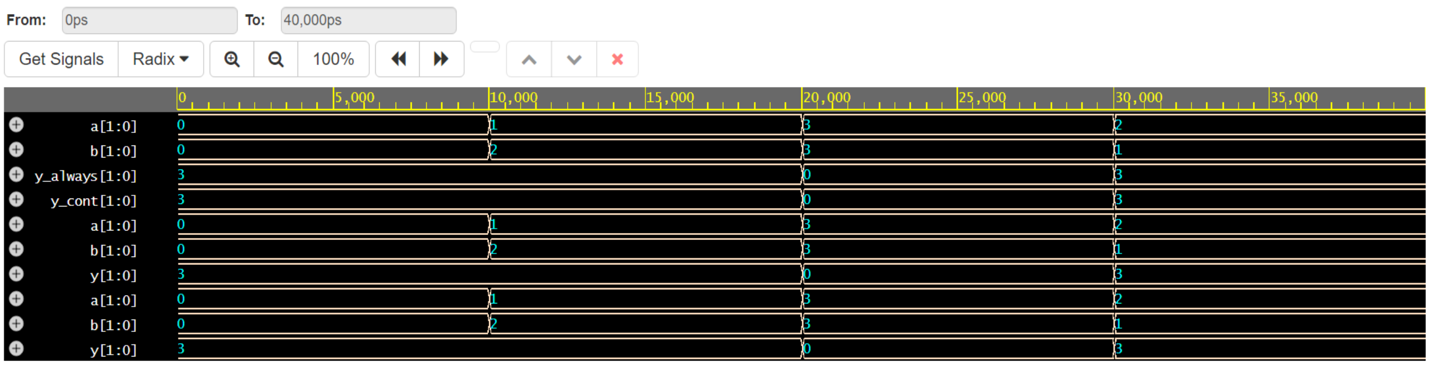
```
`timescale 1ns / 1ps  // Time unit is 1ns, and time precision is 1ps

// 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
```



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

5. **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
```

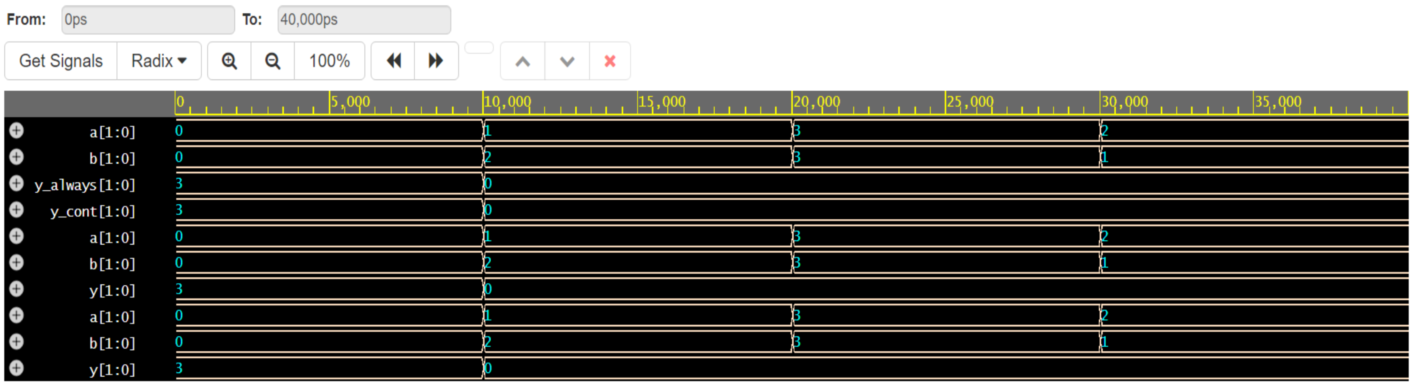
Testbench module

```
`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
```



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

6. 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
```

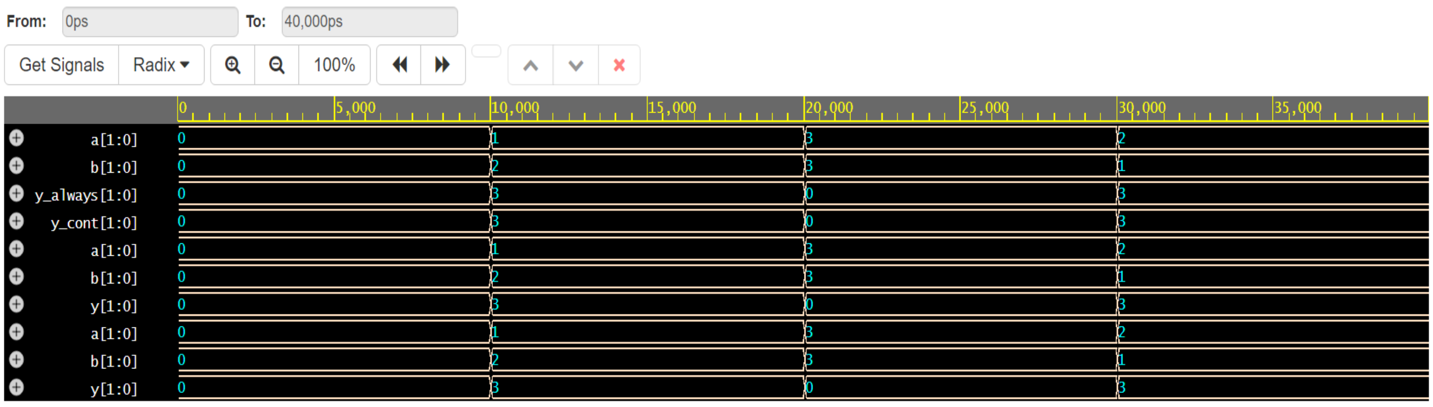
Testbench module

```
`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
```



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

7. 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
```

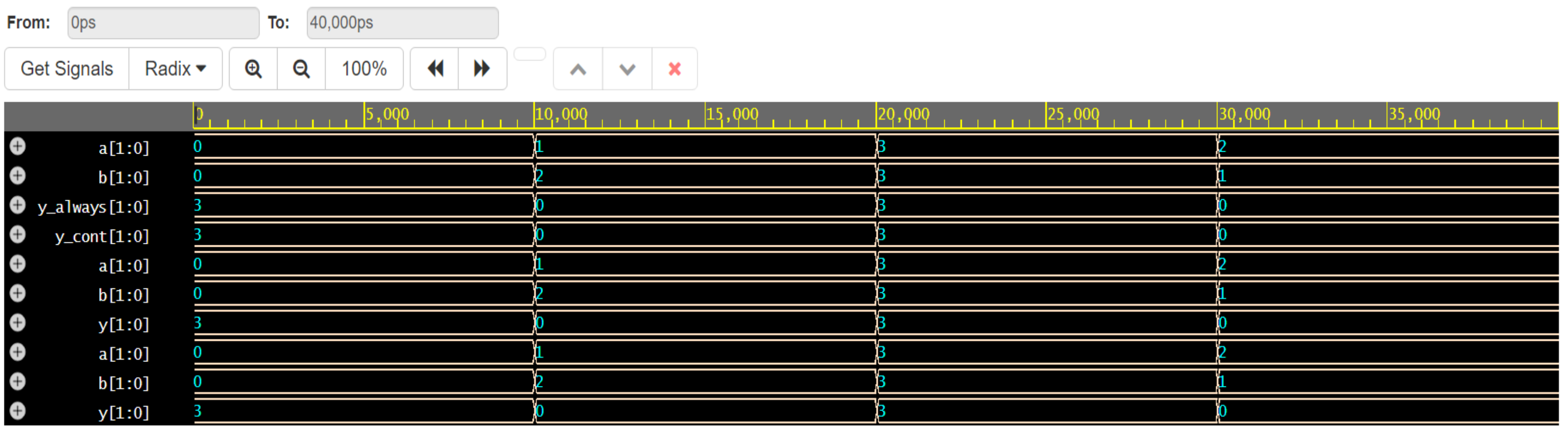
Testbench module

```
`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
```



```
a[1:0]         0   1   3   2
b[1:0]         0   2   3   1
y_always[1:0]  3   0   3   0
y_cont[1:0]    3   0   3   0
a[1:0]         0   1   3   2
b[1:0]         0   2   3   1
y[1:0]         3   0   3   0
a[1:0]         0   1   3   2
b[1:0]         0   2   3   1
y[1:0]         3   0   3   0
```

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