Anika Haque, 20283

San Francisco Bay University
EE461 Digital Design and HDL

Week#4 Combinational Logic (continue)

1. Lab Outlines:

1. Continuous assignment in combinational logic
2. Combinational logic in "always" block
3. Exercises

2. Lab Procedures

I. Exercises

a. Design the following 2-bits RTL level combinational logic modules in continuous assignment and always blocks for each gate and write the testbench to verify your design covering all the testcases by showing the waveforms,

1. 1-bit NOT gate
2. 2-bits AND gate
3. 2-bits OR gate
4. 2-bits NAND gate
5. 2-bits NOR gate
6. 2-bits XOR gate
7. 2-bits XNOR gate

<u>CONTINUOUS</u>

MODULES

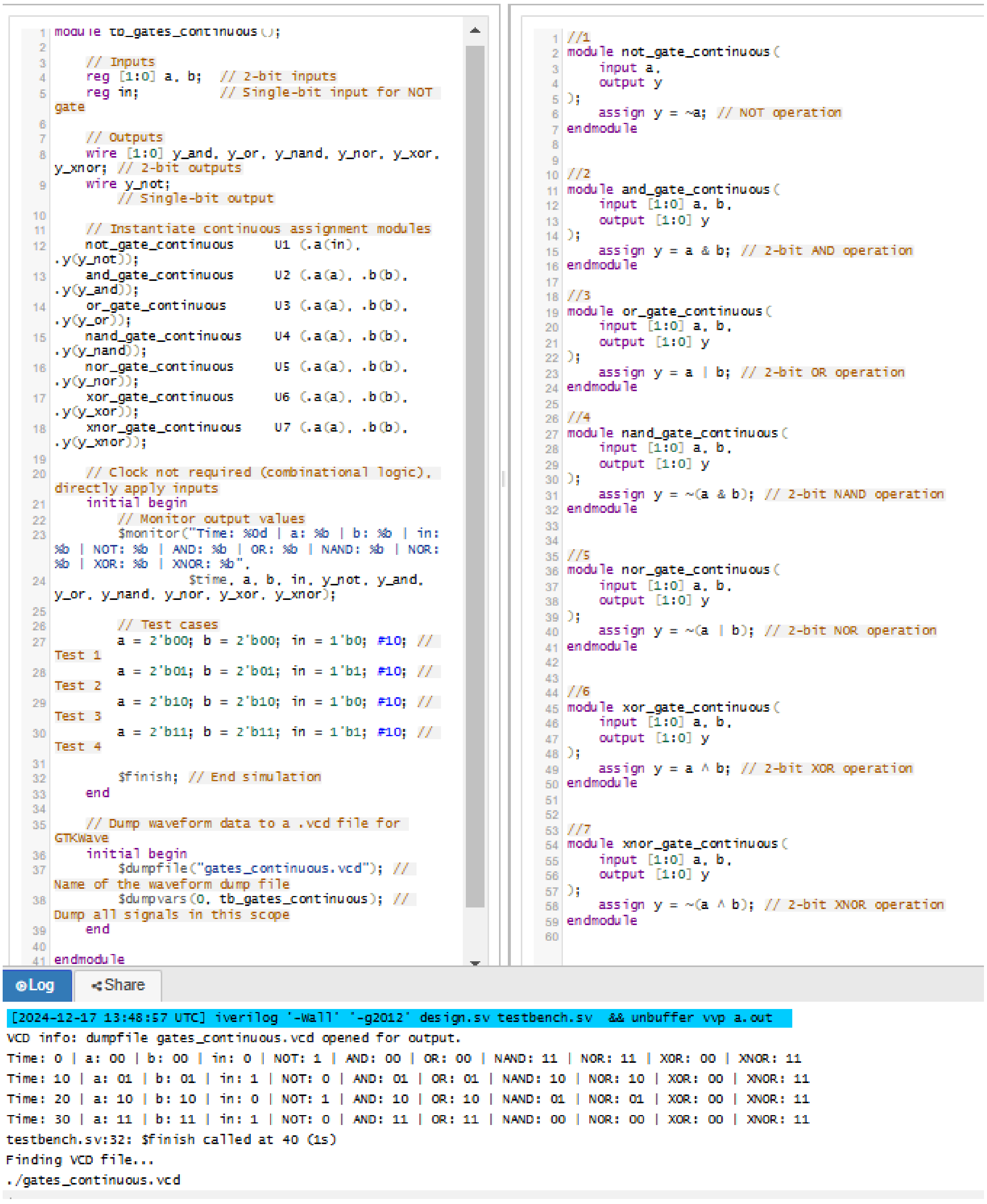
```
//1
module not_gate_continuous(
   input a,
   output y
);
   assign y = ~a; // NOT operation
endmodule



//2
module and_gate_continuous(
   input [1:0] a, b,
   output [1:0] y
);
   assign y = a & b; // 2-bit AND operation
endmodule

//3
module or_gate_continuous(
   input [1:0] a, b,
   output [1:0] y
);
```

```
    assign y = a | b; // 2-bit OR operation
endmodule

//4
module nand_gate_continuous(
    input [1:0] a, b,
    output [1:0] y
);
    assign y = ~(a & b); // 2-bit NAND operation
endmodule


//5
module nor_gate_continuous(
    input [1:0] a, b,
    output [1:0] y
);
    assign y = ~(a | b); // 2-bit NOR operation
endmodule


//6
module xor_gate_continuous(
    input [1:0] a, b,
    output [1:0] y
);
    assign y = a ^ b; // 2-bit XOR operation
endmodule


//7
module xnor_gate_continuous(
    input [1:0] a, b,
    output [1:0] y
);
    assign y = ~(a ^ b); // 2-bit XNOR operation
endmodule
```

TESTBENCH

```
module tb_gates_continuous();

    // Inputs
    reg [1:0] a, b;  // 2-bit inputs
    reg in;          // Single-bit input for NOT gate

    // Outputs
    wire [1:0] y_and, y_or, y_nand, y_nor, y_xor, y_xnor; // 2-bit outputs
    wire y_not;                                          // Single-bit output

    // Instantiate continuous assignment modules
    not_gate_continuous     U1 (.a(in), .y(y_not));
    and_gate_continuous     U2 (.a(a), .b(b), .y(y_and));
    or_gate_continuous      U3 (.a(a), .b(b), .y(y_or));
    nand_gate_continuous    U4 (.a(a), .b(b), .y(y_nand));
    nor_gate_continuous     U5 (.a(a), .b(b), .y(y_nor));
    xor_gate_continuous     U6 (.a(a), .b(b), .y(y_xor));
    xnor_gate_continuous    U7 (.a(a), .b(b), .y(y_xnor));

    // Clock not required (combinational logic), directly apply inputs
    initial begin
        // Monitor output values
        $monitor("Time: %0d | a: %b | b: %b | in: %b | NOT: %b | AND: %b | OR: %b | NAND: %b |
NOR: %b | XOR: %b | XNOR: %b",
                 $time, a, b, in, y_not, y_and, y_or, y_nand, y_nor, y_xor, y_xnor);

        // Test cases
        a = 2'b00; b = 2'b00; in = 1'b0; #10; // Test 1
        a = 2'b01; b = 2'b01; in = 1'b1; #10; // Test 2
        a = 2'b10; b = 2'b10; in = 1'b0; #10; // Test 3
        a = 2'b11; b = 2'b11; in = 1'b1; #10; // Test 4

        $finish; // End simulation
    end

    // Dump waveform data to a .vcd file for GTKWave
    initial begin
        $dumpfile("gates_continuous.vcd"); // Name of the waveform dump file
        $dumpvars(0, tb_gates_continuous); // Dump all signals in this scope
    end

endmodule
```

Anika Haque, 20283



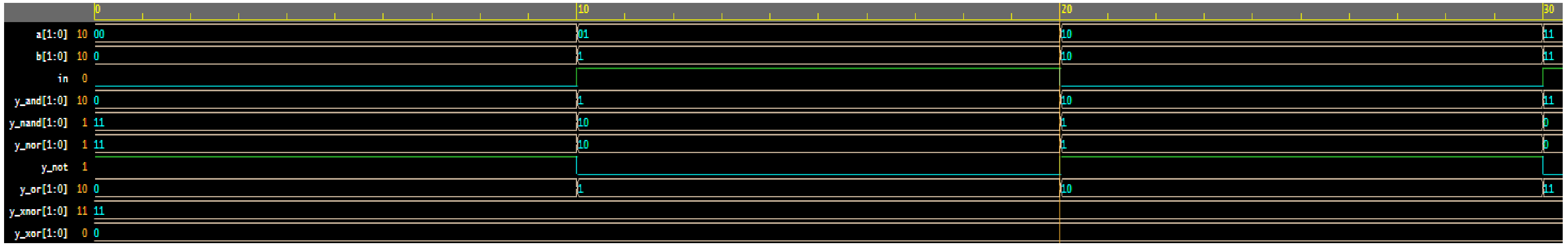
```
module tb_gates_continuous();

    // Inputs
    reg [1:0] a, b;  // 2-bit inputs
    reg in;          // Single-bit input for NOT gate

    // Outputs
    wire [1:0] y_and, y_or, y_nand, y_nor, y_xor, y_xnor; // 2-bit outputs
    wire y_not;              // Single-bit output

    // Instantiate continuous assignment modules
    not_gate_continuous     U1 (.a(in), .y(y_not));
    and_gate_continuous     U2 (.a(a), .b(b), .y(y_and));
    or_gate_continuous      U3 (.a(a), .b(b), .y(y_or));
    nand_gate_continuous    U4 (.a(a), .b(b), .y(y_nand));
    nor_gate_continuous     U5 (.a(a), .b(b), .y(y_nor));
    xor_gate_continuous     U6 (.a(a), .b(b), .y(y_xor));
    xnor_gate_continuous    U7 (.a(a), .b(b), .y(y_xnor));

    // Clock not required (combinational logic), directly apply inputs
    initial begin
        // Monitor output values
        $monitor("Time: %0d | a: %b | b: %b | in: %b | NOT: %b | AND: %b | OR: %b | NAND: %b | NOR: %b | XOR: %b | XNOR: %b",
                 $time, a, b, in, y_not, y_and, y_or, y_nand, y_nor, y_xor, y_xnor);

        // Test cases
        a = 2'b00; b = 2'b00; in = 1'b0; #10; // Test 1
        a = 2'b01; b = 2'b01; in = 1'b1; #10; // Test 2
        a = 2'b10; b = 2'b10; in = 1'b0; #10; // Test 3
        a = 2'b11; b = 2'b11; in = 1'b1; #10; // Test 4

        $finish; // End simulation
    end

    // Dump waveform data to a .vcd file for GTKWave
    initial begin
        $dumpfile("gates_continuous.vcd"); // Name of the waveform dump file
        $dumpvars(0, tb_gates_continuous); // Dump all signals in this scope
    end

endmodule
```

```
//1
module not_gate_continuous(
    input a,
    output y
);
    assign y = ~a; // NOT operation
endmodule


//2
module and_gate_continuous(
    input [1:0] a, b,
    output [1:0] y
);
    assign y = a & b; // 2-bit AND operation
endmodule

//3
module or_gate_continuous(
    input [1:0] a, b,
    output [1:0] y
);
    assign y = a | b; // 2-bit OR operation
endmodule

//4
module nand_gate_continuous(
    input [1:0] a, b,
    output [1:0] y
);
    assign y = ~(a & b); // 2-bit NAND operation
endmodule


//5
module nor_gate_continuous(
    input [1:0] a, b,
    output [1:0] y
);
    assign y = ~(a | b); // 2-bit NOR operation
endmodule


//6
module xor_gate_continuous(
    input [1:0] a, b,
    output [1:0] y
);
    assign y = a ^ b; // 2-bit XOR operation
endmodule


//7
module xnor_gate_continuous(
    input [1:0] a, b,
    output [1:0] y
);
    assign y = ~(a ^ b); // 2-bit XNOR operation
endmodule

```

```
[2024-12-17 13:48:57 UTC] iverilog '-Wall' '-g2012' design.sv testbench.sv  && unbuffer vvp a.out
VCD info: dumpfile gates_continuous.vcd opened for output.
Time: 0 | a: 00 | b: 00 | in: 0 | NOT: 1 | AND: 00 | OR: 00 | NAND: 11 | NOR: 11 | XOR: 00 | XNOR: 11
Time: 10 | a: 01 | b: 01 | in: 1 | NOT: 0 | AND: 01 | OR: 01 | NAND: 10 | NOR: 10 | XOR: 00 | XNOR: 11
Time: 20 | a: 10 | b: 10 | in: 0 | NOT: 1 | AND: 10 | OR: 10 | NAND: 01 | NOR: 01 | XOR: 00 | XNOR: 11
Time: 30 | a: 11 | b: 11 | in: 1 | NOT: 0 | AND: 11 | OR: 11 | NAND: 00 | NOR: 00 | XOR: 00 | XNOR: 11
testbench.sv:32: $finish called at 40 (1s)
Finding VCD file...
./gates_continuous.vcd
```

Anika Haque, 20283

WAVEFORM



Anika Haque, 20283

ALWAYS

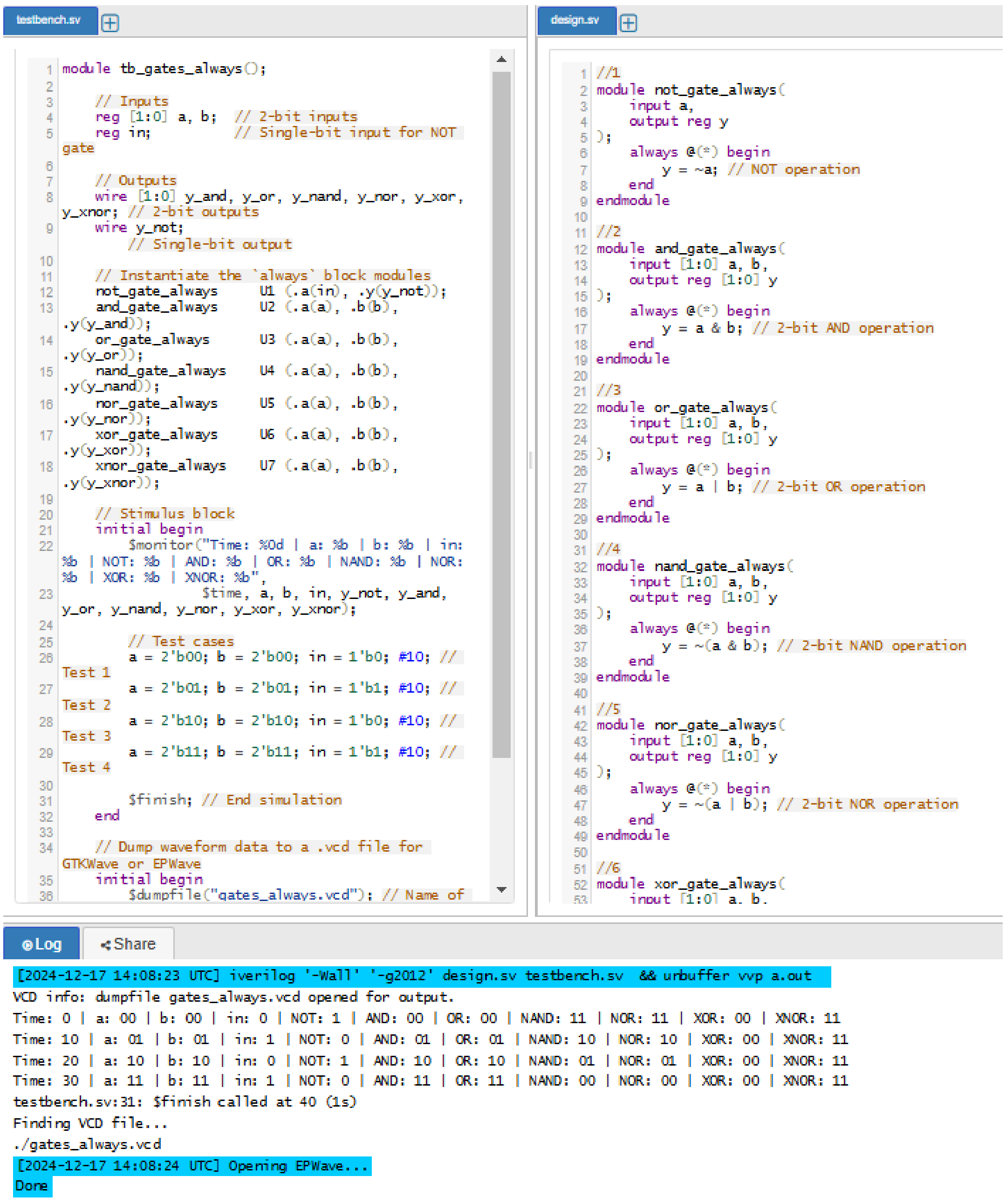
MODULES

```
//1
module not_gate_always(
    input a,
    output reg y
);
    always @(*) begin
        y = ~a; // NOT operation
    end
endmodule

//2
module and_gate_always(
    input [1:0] a, b,
    output reg [1:0] y
);
    always @(*) begin
        y = a & b; // 2-bit AND operation
    end
endmodule

//3
module or_gate_always(
    input [1:0] a, b,
    output reg [1:0] y
);
    always @(*) begin
        y = a | b; // 2-bit OR operation
    end
endmodule

//4
module nand_gate_always(
    input [1:0] a, b,
    output reg [1:0] y
);
    always @(*) begin
        y = ~(a & b); // 2-bit NAND operation
    end
endmodule

//5
module nor_gate_always(
    input [1:0] a, b,
    output reg [1:0] y
);
    always @(*) begin
        y = ~(a | b); // 2-bit NOR operation
    end
endmodule
```

```
//6
module xor_gate_always(
    input [1:0] a, b,
    output reg [1:0] y
);
    always @(*) begin
        y = a ^ b; // 2-bit XOR operation
    end
endmodule

//7
module xnor_gate_always(
    input [1:0] a, b,
    output reg [1:0] y
);
    always @(*) begin
        y = ~(a ^ b); // 2-bit XNOR operation
    end
endmodule
```

TESTBENCH

```
module tb_gates_always();

    // Inputs
    reg [1:0] a, b;  // 2-bit inputs
    reg in;          // Single-bit input for NOT gate

    // Outputs
    wire [1:0] y_and, y_or, y_nand, y_nor, y_xor, y_xnor; // 2-bit outputs
    wire y_not;                                           // Single-bit output

    // Instantiate the `always` block modules
    not_gate_always     U1 (.a(in), .y(y_not));
    and_gate_always     U2 (.a(a), .b(b), .y(y_and));
    or_gate_always      U3 (.a(a), .b(b), .y(y_or));
    nand_gate_always    U4 (.a(a), .b(b), .y(y_nand));
    nor_gate_always     U5 (.a(a), .b(b), .y(y_nor));
    xor_gate_always     U6 (.a(a), .b(b), .y(y_xor));
    xnor_gate_always    U7 (.a(a), .b(b), .y(y_xnor));

    // Stimulus block
    initial begin
        $monitor("Time: %0d | a: %b | b: %b | in: %b | NOT: %b | AND: %b | OR: %b | NAND: %b |
NOR: %b | XOR: %b | XNOR: %b",
                 $time, a, b, in, y_not, y_and, y_or, y_nand, y_nor, y_xor, y_xnor);

        // Test cases
        a = 2'b00; b = 2'b00; in = 1'b0; #10; // Test 1
        a = 2'b01; b = 2'b01; in = 1'b1; #10; // Test 2
        a = 2'b10; b = 2'b10; in = 1'b0; #10; // Test 3
        a = 2'b11; b = 2'b11; in = 1'b1; #10; // Test 4
        $finish; // End simulation
    end

    // Dump waveform data to a .vcd file for GTKWave or EPWave
    initial begin
        $dumpfile("gates_always.vcd"); // Name of the waveform file
        $dumpvars(0, tb_gates_always); // Dump all variables in this scope
    end

endmodule
```



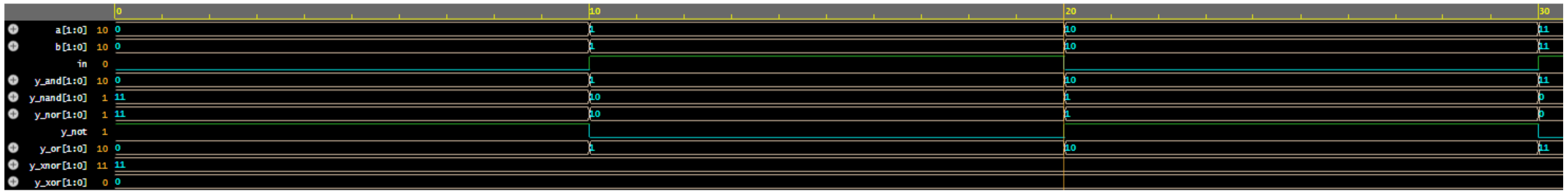
```
module tb_gates_always();

    // Inputs
    reg [1:0] a, b;   // 2-bit inputs
    reg in;           // Single-bit input for NOT
gate

    // Outputs
    wire [1:0] y_and, y_or, y_nand, y_nor, y_xor,
y_xnor; // 2-bit outputs
    wire y_not;
        // Single-bit output

    // Instantiate the `always` block modules
    not_gate_always     U1 (.a(in), .y(y_not));
    and_gate_always     U2 (.a(a), .b(b),
.y(y_and));
    or_gate_always      U3 (.a(a), .b(b),
.y(y_or));
    nand_gate_always    U4 (.a(a), .b(b),
.y(y_nand));
    nor_gate_always     U5 (.a(a), .b(b),
.y(y_nor));
    xor_gate_always     U6 (.a(a), .b(b),
.y(y_xor));
    xnor_gate_always    U7 (.a(a), .b(b),
.y(y_xnor));

    // Stimulus block
    initial begin
        $monitor("Time: %0d | a: %b | b: %b | in:
%b | NOT: %b | AND: %b | OR: %b | NAND: %b | NOR:
%b | XOR: %b | XNOR: %b",
                 $time, a, b, in, y_not, y_and,
y_or, y_nand, y_nor, y_xor, y_xnor);

        // Test cases
        a = 2'b00; b = 2'b00; in = 1'b0; #10; //
Test 1
        a = 2'b01; b = 2'b01; in = 1'b1; #10; //
Test 2
        a = 2'b10; b = 2'b10; in = 1'b0; #10; //
Test 3
        a = 2'b11; b = 2'b11; in = 1'b1; #10; //
Test 4

        $finish; // End simulation
    end

    // Dump waveform data to a .vcd file for
GTKWave or EPWave
    initial begin
        $dumpfile("gates_always.vcd"); // Name of
```

```
//1
module not_gate_always(
    input a,
    output reg y
);
    always @(*) begin
        y = ~a; // NOT operation
    end
endmodule

//2
module and_gate_always(
    input [1:0] a, b,
    output reg [1:0] y
);
    always @(*) begin
        y = a & b; // 2-bit AND operation
    end
endmodule

//3
module or_gate_always(
    input [1:0] a, b,
    output reg [1:0] y
);
    always @(*) begin
        y = a | b; // 2-bit OR operation
    end
endmodule

//4
module nand_gate_always(
    input [1:0] a, b,
    output reg [1:0] y
);
    always @(*) begin
        y = ~(a & b); // 2-bit NAND operation
    end
endmodule

//5
module nor_gate_always(
    input [1:0] a, b,
    output reg [1:0] y
);
    always @(*) begin
        y = ~(a | b); // 2-bit NOR operation
    end
endmodule

//6
module xor_gate_always(
    input [1:0] a, b,
```

```
[2024-12-17 14:08:23 UTC] iverilog '-Wall' '-g2012' design.sv testbench.sv  && unbuffer vvp a.out
VCD info: dumpfile gates_always.vcd opened for output.
Time: 0 | a: 00 | b: 00 | in: 0 | NOT: 1 | AND: 00 | OR: 00 | NAND: 11 | NOR: 11 | XOR: 00 | XNOR: 11
Time: 10 | a: 01 | b: 01 | in: 1 | NOT: 0 | AND: 01 | OR: 01 | NAND: 10 | NOR: 10 | XOR: 00 | XNOR: 11
Time: 20 | a: 10 | b: 10 | in: 0 | NOT: 1 | AND: 10 | OR: 10 | NAND: 01 | NOR: 01 | XOR: 00 | XNOR: 11
Time: 30 | a: 11 | b: 11 | in: 1 | NOT: 0 | AND: 11 | OR: 11 | NAND: 00 | NOR: 00 | XOR: 00 | XNOR: 11
testbench.sv:31: $finish called at 40 (1s)
Finding VCD file...
./gates_always.vcd
[2024-12-17 14:08:24 UTC] Opening EPWave...
Done
```

Anika Haque, 20283

WAVEFORM


a[1:0] 10
b[1:0] 10
in 0
y_and[1:0] 10
y_nand[1:0] 1
y_nor[1:0] 1
y_not 1
y_or[1:0] 10
y_xnor[1:0] 11
y_xor[1:0] 0