



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

CONTINUOUS

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
```

```

1 module tb_gates_continuous();
2
3     // Inputs
4     reg [1:0] a, b; // 2-bit inputs
5     reg in; // Single-bit input for NOT
6
7     // Outputs
8     wire [1:0] y_and, y_or, y_nand, y_nor, y_xor,
9     y_xnor; // 2-bit outputs
10    wire y_not;
11    // Single-bit output
12
13    // Instantiate continuous assignment modules
14    not_gate_continuous U1 (.a(in),
15    .y(y_not));
16    and_gate_continuous U2 (.a(a), .b(b),
17    .y(y_and));
18    or_gate_continuous U3 (.a(a), .b(b),
19    .y(y_or));
20    nand_gate_continuous U4 (.a(a), .b(b),
21    .y(y_nand));
22    nor_gate_continuous U5 (.a(a), .b(b),
23    .y(y_nor));
24    xor_gate_continuous U6 (.a(a), .b(b),
25    .y(y_xor));
26    xnor_gate_continuous U7 (.a(a), .b(b),
27    .y(y_xnor));
28
29    // Clock not required (combinational logic),
30    directly apply inputs
31    initial begin
32        // Monitor output values
33        $monitor("Time: %0d | a: %b | b: %b | in:
34        %b | NOT: %b | AND: %b | OR: %b | NAND: %b | NOR:
35        %b | XOR: %b | XNOR: %b",
36        $time, a, b, in, y_not, y_and,
37        y_or, y_nand, y_nor, y_xor, y_xnor);
38
39        // Test cases
40        a = 2'b00; b = 2'b00; in = 1'b0; #10; //
41    Test 1
42        a = 2'b01; b = 2'b01; in = 1'b1; #10; //
43    Test 2
44        a = 2'b10; b = 2'b10; in = 1'b0; #10; //
45    Test 3
46        a = 2'b11; b = 2'b11; in = 1'b1; #10; //
47    Test 4
48
49        $finish; // End simulation
50    end
51
52    // Dump waveform data to a .vcd file for
53    GTKWave
54    initial begin
55        $dumpfile("gates_continuous.vcd"); //
56        Name of the waveform dump file
57        $dumpvars(0, tb_gates_continuous); //
58        Dump all signals in this scope
59    end
60
61 endmodule

```

```

1 //1
2 module not_gate_continuous(
3     input a,
4     output y
5 );
6     assign y = ~a; // NOT operation
7 endmodule
8
9 //2
10 module and_gate_continuous(
11     input [1:0] a, b,
12     output [1:0] y
13 );
14     assign y = a & b; // 2-bit AND operation
15 endmodule
16
17 //3
18 module or_gate_continuous(
19     input [1:0] a, b,
20     output [1:0] y
21 );
22     assign y = a | b; // 2-bit OR operation
23 endmodule
24
25 //4
26 module nand_gate_continuous(
27     input [1:0] a, b,
28     output [1:0] y
29 );
30     assign y = ~(a & b); // 2-bit NAND operation
31 endmodule
32
33 //5
34 module nor_gate_continuous(
35     input [1:0] a, b,
36     output [1:0] y
37 );
38     assign y = ~(a | b); // 2-bit NOR operation
39 endmodule
40
41 //6
42 module xor_gate_continuous(
43     input [1:0] a, b,
44     output [1:0] y
45 );
46     assign y = a ^ b; // 2-bit XOR operation
47 endmodule
48
49 //7
50 module xnor_gate_continuous(
51     input [1:0] a, b,
52     output [1:0] y
53 );
54     assign y = ~(a ^ b); // 2-bit XNOR operation
55 endmodule
56
57
58
59
60

```

Log Share

```

[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

```

WAVEFORM



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);
    end

    // 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
endmodule
```

```
$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
```

```

1 module tb_gates_always();
2
3     // Inputs
4     reg [1:0] a, b; // 2-bit inputs
5     reg in; // Single-bit input for NOT
6
7     // Outputs
8     wire [1:0] y_and, y_or, y_nand, y_nor, y_xor,
9     y_xnor; // 2-bit outputs
10    wire y_not; // Single-bit output
11
12    // Instantiate the 'always' block modules
13    not_gate_always U1 (.a(in), .y(y_not));
14    and_gate_always U2 (.a(a), .b(b),
15    .y(y_and));
16    or_gate_always U3 (.a(a), .b(b),
17    .y(y_or));
18    nand_gate_always U4 (.a(a), .b(b),
19    .y(y_nand));
20    nor_gate_always U5 (.a(a), .b(b),
21    .y(y_nor));
22    xor_gate_always U6 (.a(a), .b(b),
23    .y(y_xor));
24    xnor_gate_always U7 (.a(a), .b(b),
25    .y(y_xnor));
26
27    // Stimulus block
28    initial begin
29        $monitor("Time: %0d | a: %b | b: %b | in: %b | NOT: %b | AND: %b | OR: %b | NAND: %b | NOR: %b | XOR: %b | XNOR: %b",
30        $time, a, b, in, y_not, y_and,
31        y_or, y_nand, y_nor, y_xor, y_xnor);
32
33        // Test cases
34        a = 2'b00; b = 2'b00; in = 1'b0; #10; //
35    Test 1
36        a = 2'b01; b = 2'b01; in = 1'b1; #10; //
37    Test 2
38        a = 2'b10; b = 2'b10; in = 1'b0; #10; //
39    Test 3
40        a = 2'b11; b = 2'b11; in = 1'b1; #10; //
41    Test 4
42
43        $finish; // End simulation
44    end
45
46    // Dump waveform data to a .vcd file for
47    GTKWave or EPWave
48    initial begin
49        $dumpfile("gates_always.vcd"); // Name of
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

```

1 //1
2 module not_gate_always(
3     input a,
4     output reg y
5 );
6     always @(*) begin
7         y = ~a; // NOT operation
8     end
9 endmodule
10
11 //2
12 module and_gate_always(
13     input [1:0] a, b,
14     output reg [1:0] y
15 );
16     always @(*) begin
17         y = a & b; // 2-bit AND operation
18     end
19 endmodule
20
21 //3
22 module or_gate_always(
23     input [1:0] a, b,
24     output reg [1:0] y
25 );
26     always @(*) begin
27         y = a | b; // 2-bit OR operation
28     end
29 endmodule
30
31 //4
32 module nand_gate_always(
33     input [1:0] a, b,
34     output reg [1:0] y
35 );
36     always @(*) begin
37         y = ~(a & b); // 2-bit NAND operation
38     end
39 endmodule
40
41 //5
42 module nor_gate_always(
43     input [1:0] a, b,
44     output reg [1:0] y
45 );
46     always @(*) begin
47         y = ~(a | b); // 2-bit NOR operation
48     end
49 endmodule
50
51 //6
52 module xor_gate_always(
53     input [1:0] a, b,
54     output reg [1:0] y
55 );
56     always @(*) begin
57         y = a ^ b; // 2-bit XOR operation
58     end
59 endmodule
60
61 //7
62 module xnor_gate_always(
63     input [1:0] a, b,
64     output reg [1:0] y
65 );
66     always @(*) begin
67         y = a ^~ b; // 2-bit XNOR operation
68     end
69 endmodule
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

```

@Log  <Share
[2024-12-17 14:08:23 UTC] iverilog '-Wall' '-g2012' design.v testbench.v && 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: 0 | NOT: 0 | AND: 01 | OR: 01 | NAND: 10 | NOR: 10 | XOR: 00 | XNOR: 11
Time: 20 | a: 10 | b: 10 | in: 0 | NOT: 0 | 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.v:31: $finish called at 40 (1s)
Finding VCD file...
./gates_always.vcd
[2024-12-17 14:08:24 UTC] Opening EPWave...
Done

```


WAVEFORM

