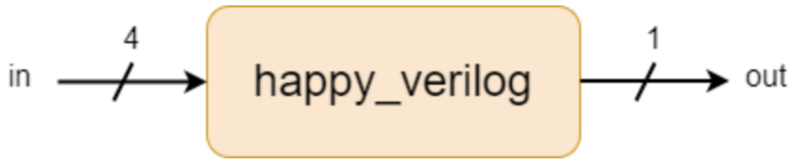
Lab 1 Multiples of 2 or 3 Detector

Example

To help you understand Verilog, we provide an example below.



Here is a 4-bit module named "happy_verilog", and its function detecting prime numbers between 0 and 15.
We will use it to help you understand three different ways of describing a circuit.

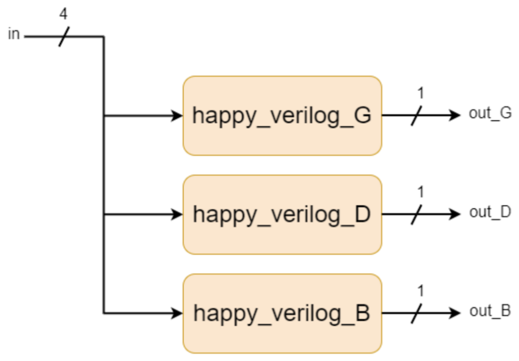
- Prime numbers: 2, 3, 5, 7, 11, 13, 17, 19, …

Therefore, here we need to find 2, 3, 5, 7, 11, 13.

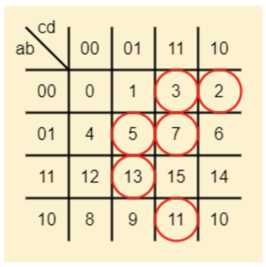
Description

1. Three ways to describe the circuit.

There are three ways to describe the circuit, “gate level”, “dataflow”, and “behavior”. We will talk more about it in the following parts.



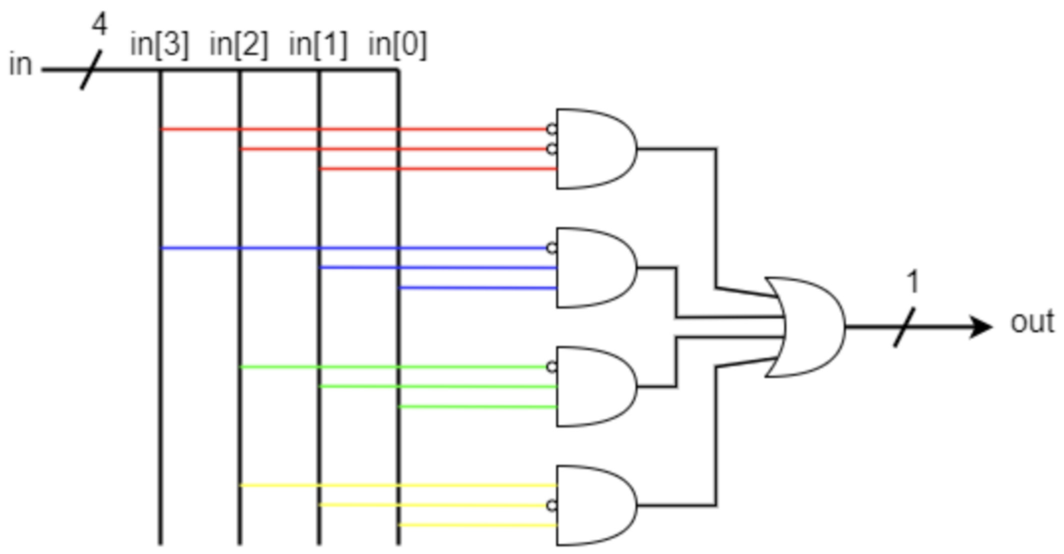
2. K-map



ab \ cd	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

3. Structure

Simplify the K-map, you will get F(a, b, c, d) = a’b’c + a’cd + b’cd + bc’d



a, b, c, d are in[3], in[2], in[1], in[0] in the circuit.

4. Implement in Verilog

(1) Gate Level Description

```verilog
module happy_verilog_G(in, out);

    parameter n = 4;

    //IO port declaration
    input [n - 1: 0]in;//Input must be wire.
    output out;//Output can be wire or reg, default type is wire.

    //Save the output values from different gates.
    wire not_a, not_b, not_c;
    wire and0, and1, and2, and3;

    //Declare a gate.
    //<gate><gate name>(output,input1,input2,...);
    //Don't use duplicate <gate name>
    not not_gate0(not_a, in[3]);
    not not_gate1(not_b, in[2]);
    not not_gate2(not_c, in[1]);

    and and_gate0(and0, not_a, not_b, in[1]);
    and and_gate1(and1, not_a, in[1], in[0]);
    and and_gate2(and2, not_b, in[1], in[0]);
    and and_gate3(and3, in[2], not_c, in[0]);

    or or_gate(out, and0, and1, and2, and3);

endmodule
```

(2) Dataflow Description

```verilog
module happy_verilog_D(in, out);

    parameter n = 4;

    input [n - 1: 0]in;
    output out;

    //Be careful, "!" and "~" are different in Verilog.
    assign out = (!in[3] & !in[2] & in[1])
                | (!in[3] & in[1] & in[0])
                | (!in[2] & in[1] & in[0])
                | (in[2] & !in[1] & in[0]);

endmodule
```

(3) Behavior Description

```verilog
module happy_verilog_B(in, out);

    parameter n = 4;

    input [n - 1: 0]in;
    output out;
    reg out; //It must be reg type because an always block is used to set it.

    always@(*)begin
        case(in)
            2, 3, 5, 7, 11, 13:begin
                out = 1'b1;
            end
            default:begin
                out = 1'b0;
            end
        endcase
    end

    /*
    or can write as
    always@(*)begin
        out = 1'b0;
        case(in)
            2, 3, 5, 7, 11, 13:begin
                out=1'b1;
            end
        endcase
    end
    */

endmodule
```

5. Testbench

```verilog
module happy_verilog_tb;
    parameter delay = 5;

    wire out_G, out_D, out_B;
    reg [3: 0]in;
    integer i;

    wire is;
    assign is = in == 2 || in == 3 || in == 5 || in == 7 || in == 11 || in == 13;

    initial begin
        //Give initial value.
        in = 4'd0;
        for(i = 0; i < 16; i = i + 1)begin
            #delay;
            $display("time = %4d, in = %b, out_G = %b, out_D = %b, out_B = %b",
                     $time, in, out_G, out_D, out_B);
            if((out_G === 1'bx | out_D === 1'bx | out_B === 1'bx |
                out_G === 1'bz | out_D === 1'bz | out_B === 1'bz) ||
                ( is && (out_G & out_D & out_B) == 0)||
                (!is && (out_G | out_D | out_B) == 1))
                begin
                    $display("You got wrong answer!!");
                    $finish;
                end
            in = in + 4'd1;
        end
        $display("Congratulations!!");
        $finish;
    end

    //Connect with the "gate level" module.
    happy_verilog_G hvg( .in(in), .out(out_G) );

    //Connect with the "dataflow" module.
    happy_verilog_D hvd( .in(in), .out(out_D) );

    //Connect with the "behavior" module.
    happy_verilog_B hvb( .in(in), .out(out_B) );

endmodule
```

Write and run the code by yourself, it’s a great chance to practice.

Use NC-Verilog to simulate and verify

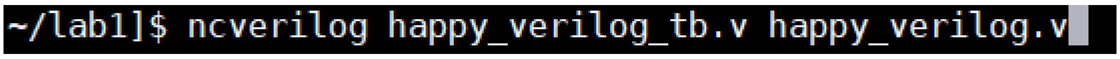
How to execute

Use command, “ncverilog”, to execute testbench and modules.

Here is an example.
Assume testbench is written in a file called “happy_verilog_tb.v”
And all three modules are written in a file called “happy_verilog.v”

Then our command would be:

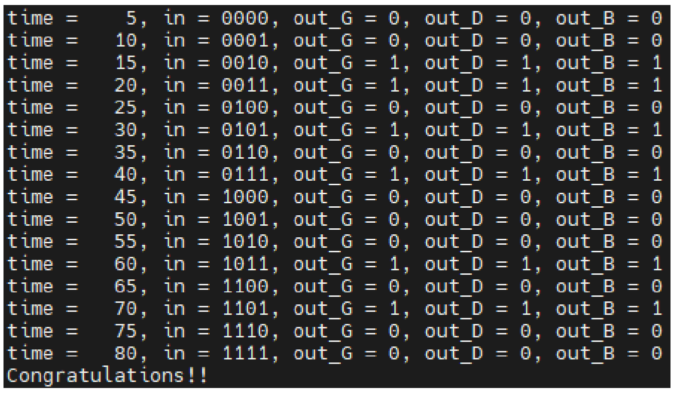
```
ncverilog happy_verilog_tb.v happy_verilog.v
```



```
~/lab1]$ ncverilog happy_verilog_tb.v happy_verilog.v
```

Result

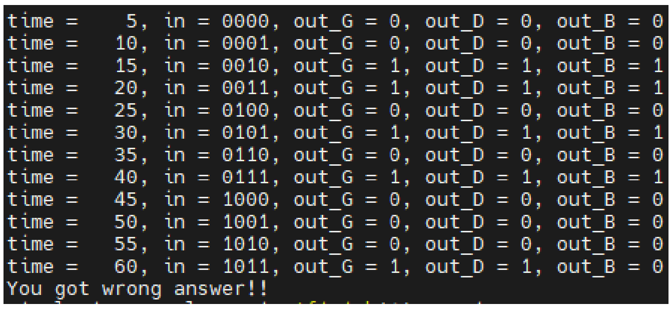
Right answer:



```
time =    5, in = 0000, out_G = 0, out_D = 0, out_B = 0
time =   10, in = 0001, out_G = 0, out_D = 0, out_B = 0
time =   15, in = 0010, out_G = 1, out_D = 1, out_B = 1
time =   20, in = 0011, out_G = 1, out_D = 1, out_B = 1
time =   25, in = 0100, out_G = 0, out_D = 0, out_B = 0
time =   30, in = 0101, out_G = 1, out_D = 1, out_B = 1
time =   35, in = 0110, out_G = 0, out_D = 0, out_B = 0
time =   40, in = 0111, out_G = 1, out_D = 1, out_B = 1
time =   45, in = 1000, out_G = 0, out_D = 0, out_B = 0
time =   50, in = 1001, out_G = 0, out_D = 0, out_B = 0
time =   55, in = 1010, out_G = 0, out_D = 0, out_B = 0
time =   60, in = 1011, out_G = 1, out_D = 1, out_B = 1
time =   65, in = 1100, out_G = 0, out_D = 0, out_B = 0
time =   70, in = 1101, out_G = 1, out_D = 1, out_B = 1
time =   75, in = 1110, out_G = 0, out_D = 0, out_B = 0
time =   80, in = 1111, out_G = 0, out_D = 0, out_B = 0
Congratulations!!
```

(All testcases are correct!)

Wrong answer:



```
time =    5, in = 0000, out_G = 0, out_D = 0, out_B = 0
time =   10, in = 0001, out_G = 0, out_D = 0, out_B = 0
time =   15, in = 0010, out_G = 1, out_D = 1, out_B = 1
time =   20, in = 0011, out_G = 1, out_D = 1, out_B = 1
time =   25, in = 0100, out_G = 0, out_D = 0, out_B = 0
time =   30, in = 0101, out_G = 1, out_D = 1, out_B = 1
time =   35, in = 0110, out_G = 0, out_D = 0, out_B = 0
time =   40, in = 0111, out_G = 1, out_D = 1, out_B = 1
time =   45, in = 1000, out_G = 0, out_D = 0, out_B = 0
time =   50, in = 1001, out_G = 0, out_D = 0, out_B = 0
time =   55, in = 1010, out_G = 0, out_D = 0, out_B = 0
time =   60, in = 1011, out_G = 1, out_D = 1, out_B = 0
You got wrong answer!!
```

(The last testcase in the image is wrong!)

Assignment

Design a 4-bit input circuit that can detect multiples of 2 or 3, and simplify it using Karnaugh maps. Then, write Verilog code for the circuit using gate-level, dataflow, and behavioral descriptions.

- multiples of 2 or 3: 0, 2, 3, 4, 6, 8, 9, …etc.
- Gate Level Description
  - module name: MultiplesDetector_G
  - input name: in
  - output name: out
- Dataflow Description
  - module name: MultiplesDetector_D
  - input name: in
  - output name: out
- Behavior Description
  - module name: MultiplesDetector_B
  - input name: in
  - output name: out

Please write three description in different modules and put all of them in one file, “MultiplesDetector.v”

Write testbench in a file called “MultiplesDetector_tb.v”

Follow the example above, design your testbench and test your own Multiples Detector.

Files to upload

1. MultiplesDetector.v
2. {studentID}_report.pdf (For example, 111062000_report.pdf)
   (1) K-map, circuit diagram, and execution result (screen shot).
   (2) The problem you faced and how you deal with it.
   (3) What have you learned from this lab?

Please upload these two files on the eeclass system.

Be careful of the file name, module name, and whatever listed above.

Wrong format will result in the failure(0 points) of your assignment, please double check before uploading it.

Deadline

- 2023/04/26 23:59

Upload the files before the deadline!

Late assignment will get a 10% penalty per day, and no assignment will be accepted after 3 days from the given due date.

Do not plagiarize!