UNIVERSITY OF BATANGAS – LIPA CAMPUS

COLLEGE OF ENGINEERING AND ARCHITECTURE

COMPUTER ENGINEERING

MODULE INSTANTIATION and TEST BENCHES

Experiment #1

Hardware Description Language

NAME: Castillo, Joaquin Iverson M.

STUDENT NUMBER
2020759

DATE OF SUBMISSION:
October 09, 2022

PROFESSOR: Sir Charles Juanillas

Discussion:

A hardware description language (HDL) is a computer-based language that describes the hardware of digital systems in a textual form. It resembles an ordinary computer programming language, such as C, but is specifically oriented to describing hardware structures and behaviour of logic circuits.

HDLs are used in several major steps in the design flow of an integrated circuit:

- design entry,
- functional simulation or verification,
- logic synthesis,
- timing verification,
- fault simulation.

Companies that design integrated circuits use proprietary and public HDLs. In the public domain, there are two standard HDLs that are supported by the IEEE: VHDL and Verilog.

VHDL

- *stands for VHSIC (very high speed integrated circuit) HDL*
- *a Department of Defense mandated language*

Verilog

- *began as a proprietary of companies and universities known as Open Verilog International (OVI) as a step leading to its adoption as an IEEE standard.*
- *It was initially approved as a standard HDL in 1995; revised and enhanced versions of the language were approved in 2001 and 2005.*

Throughout this course, the Verilog HDL descriptions will be listed to introduce a design methodology based on the concept of computer-aided modelling of digital systems.

Module Declaration

In particular, a Verilog model is composed of text using keywords, of which there are about 100. Keywords are predefined lowercase identifiers that define the language constructs. Any text following the two forward slashes is interpreted as a comment and will have no effect on a simulation using the model. Multiline comments begin with /* and terminate with */. Blank spaces are ignored, but they may not appear within the text of a keyword, an identifier, an operator, or the representation of a number. Verilog is case-sensitive, which means that the uppercase and lowercase letters are distinguishable.

A module is the fundamental descriptive unit in the Verilog language. It is declared by the keyword **module** and must always be terminated by the keyword **endmodule**.

```
module module_name (port list);
        //Verilog
statements endmodule
```

The port list of a module is the interface between the module and its environment. This list is enclosed in parentheses, and commas are used to separate elements of the list. The statement is terminated with a semicolon (;). The keywords **input** and **output** specify which of the ports are inputs and which are outputs. Internal connections are declared as wires. This connection is declared with the keyword **wire**.

```
input      in1;
output out1,
out2; wire x, y, z;
```

Test Benches

In order to simulate a circuit with an HDL, it is necessary to apply inputs to the circuit so that the simulator will generate an output response. An HDL description that provides the stimulus to a design is called a test bench. In its simplest form, a test bench is a module containing a single generator and an instantiation of the model that is to be verified. Note that it has no input or output ports, because it does not interact with its environment. Within the test bench, the inputs to the circuit are declared with keyword **reg** and the outputs are declared with the keyword **wire**. Note that using a test bench is similar to testing actual hardware by attaching signal generators to the inputs of a circuit and attaching probes (wires) to the outputs of the circuit).

```
reg  a,  b,  c;
wire w1, w2;
```

The **initial** keyword is used with a set of statements that begin executing when the simulation is initialized, and terminates execution when the last statement has finished executing. The set of statements to be executed is called a block statement andconsists of several statements enclosed by keywords **begin** and **end**. The action specified by the statements begins when the simulation is launched, and the statementsare executed in sequence.

The response to the stimulus generated by the initial and always blocks will appear in text format as standard output and as waveforms (timing diagrams) in simulators having graphical output capability. Numerical outputs are displayed by using Verilog system tasks. These are built-in system functions that are recognized by keywords thatbegin with the symbol **$.** Some of the system tasks that are useful for display are:

$display – display a one-time value of variables or strings with an end-of-line return,
$write – same as $display, but without going to next line,
$monitor – display variables whenever a value changes during a simulation run,
$time – display the simulation time,
$finish – terminate the simulation.

The syntax is of the form:

Task_name(format specification, argument list);

The format specification uses the symbol % to specify radix of numbers that are to be displayed and may have a string enclosed in quotes. The base may be binary (%b), decimal (%d), and hexadecimal (%h)

Exercises

A. Module Declaration

The HDL description of the circuit of the Fig 1.1 is shown in the example below. Open your editor and place the code inside as drill1_1.v.

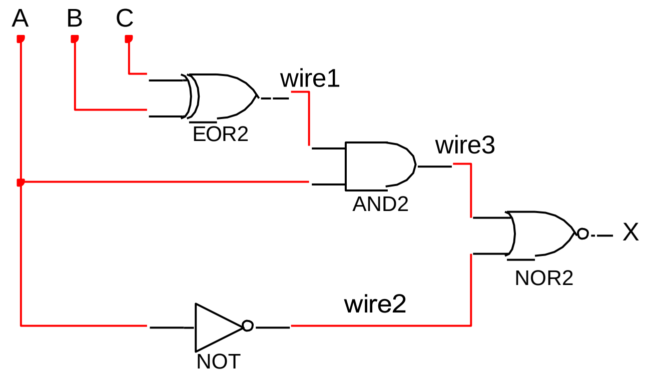
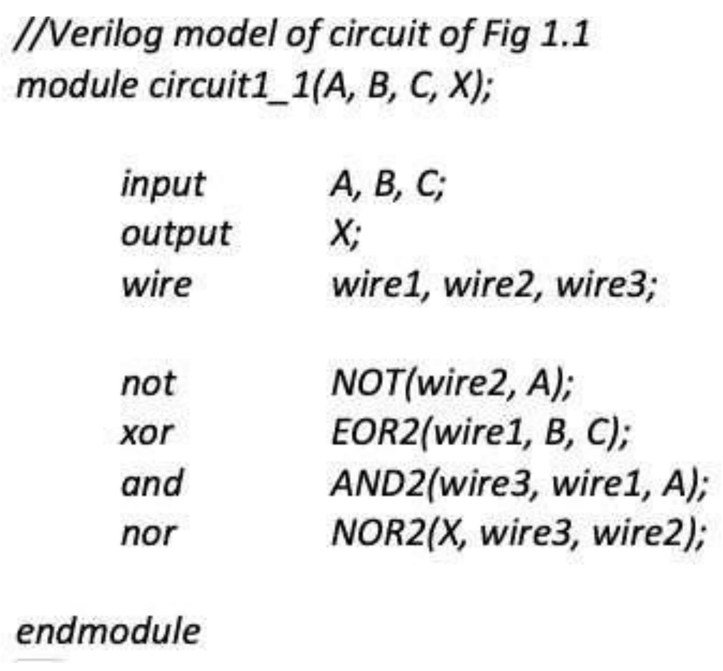


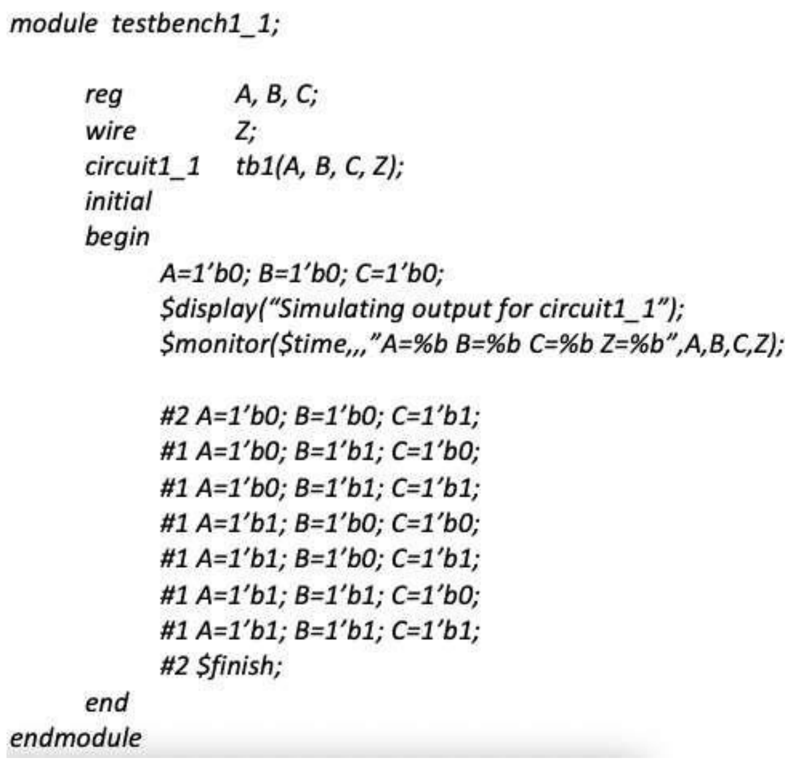
Fig. 1.1



```
//Verilog model of circuit of Fig 1.1
module circuit1_1(A, B, C, X);

        input       A, B, C;
        output      X;
        wire        wire1, wire2, wire3;

        not         NOT(wire2, A);
        xor         EOR2(wire1, B, C);
        and         AND2(wire3, wire1, A);
        nor         NOR2(X, wire3, wire2);

endmodule
```

B. Test Bench

Edit the saved drill1_1 by placing the following code below the previous code.



```
module  testbench1_1;

          reg            A, B, C;
          wire           Z;
          circuit1_1     tb1(A, B, C, Z);
          initial
          begin
                    A=1'b0; B=1'b0; C=1'b0;
                    $display("Simulating output for circuit1_1");
                    $monitor($time,,,"A=%b B=%b C=%b Z=%b",A,B,C,Z);

                    #2 A=1'b0; B=1'b0; C=1'b1;
                    #1 A=1'b0; B=1'b1; C=1'b0;
                    #1 A=1'b0; B=1'b1; C=1'b1;
                    #1 A=1'b1; B=1'b0; C=1'b0;
                    #1 A=1'b1; B=1'b0; C=1'b1;
                    #1 A=1'b1; B=1'b1; C=1'b0;
                    #1 A=1'b1; B=1'b1; C=1'b1;
                    #2 $finish;
          end
endmodule
```

C. Code the next example and save it under the filename Drill1_2.v

```
//dataflow Verilog code for a simple 1-bit full subtracter.
module full_subtract(diff, borrowOut, a, b, borrowIn);
        output        diff;
        output        borrowOut;
        input         a, b, borrowIn;
        assign        {borrowOut, diff} = a – b – borrowIn;
        //result of subtraction is two bits; the MSB is borrowOut and the LSB
        //is diff.
endmodule

module testingFS();

        reg           a, b,borrowIn;
        wire          diff, borrowOut;
        full_subtract        fs(diff, borrowOut, a, b, borrowIn);

        initial begin
        a=1’b1; b=1’b1; borrowIn=1’b0;
        end

        initial begin
        #10 a=1’b1;
        #10 a=1’b0; b=1’b1;
        #10 a=1’b1; b=1’b0;
        #10 borrowIn=1’b1;
        end

        initial begin
        $display(“   a    b   borrowIn    difference   borrowOut  time”);
        $monitor(“ %b   %b   %b     %b     %b   %b ”, a, b, borrowIn, diff, borrowOut,
                $time);
        #10 $finish;
        end
endmodule
```

Programming Exercise

1. Copy the code below, and then create a test bench for it to determine what a combination circuit is being simulated. Save as exercise1_1.v

```
module exercise1_1(W, X, Y, Z);
        output       [0:3] W;
        input        X, Y;
        input        Z;
        wire         X1, Y1, Z1;

        not    G1(X1, X), G2(Y1, Y),G3(Z1, Z);

        nand  G4(W[0], X1, Y1, Z1),G5(W[1], X1, Y, Z1),
              G6(W[2], X, Y1, Z1),G7(W[3], X, Y, Z1);
endmodule
```

2. Edit the test bench from Drill1_1 in such a way that the same output will be obtained without using the **$monitor** task. Terminate your simulation after 80ns. Save the new file as exercise1_2.v

3. Describe the circuit that is being simulated by the program below. Verify your answer by displaying the truth table using an appropriate test bench. Save the new file as exercise1_3.v

```
module exercise1_3(
        output       var1, x_4,
        input        x_1, x_2, x_3);
        xor          EOR1(x_4, x_1, x_2, x_3);
        xor          EOR2(var1, x_1, x_2, x_3, x_4);
endmodule
```

Review Questions

1. Based from Drill1_1, notice that wire1, wire2, and wire3 are outputs of gates EOR2, NOT and AND2 respectively. Why are they declared as wire instead of output?

Verilog wire functions represent actual wires that connect gates and/or modules. Wires can only be assigned to a particular value by repeatedly assigning them or connecting them to real outputs from the inputs and their outputs are meaningless.

2. How do you instantiate a module within a test bench?

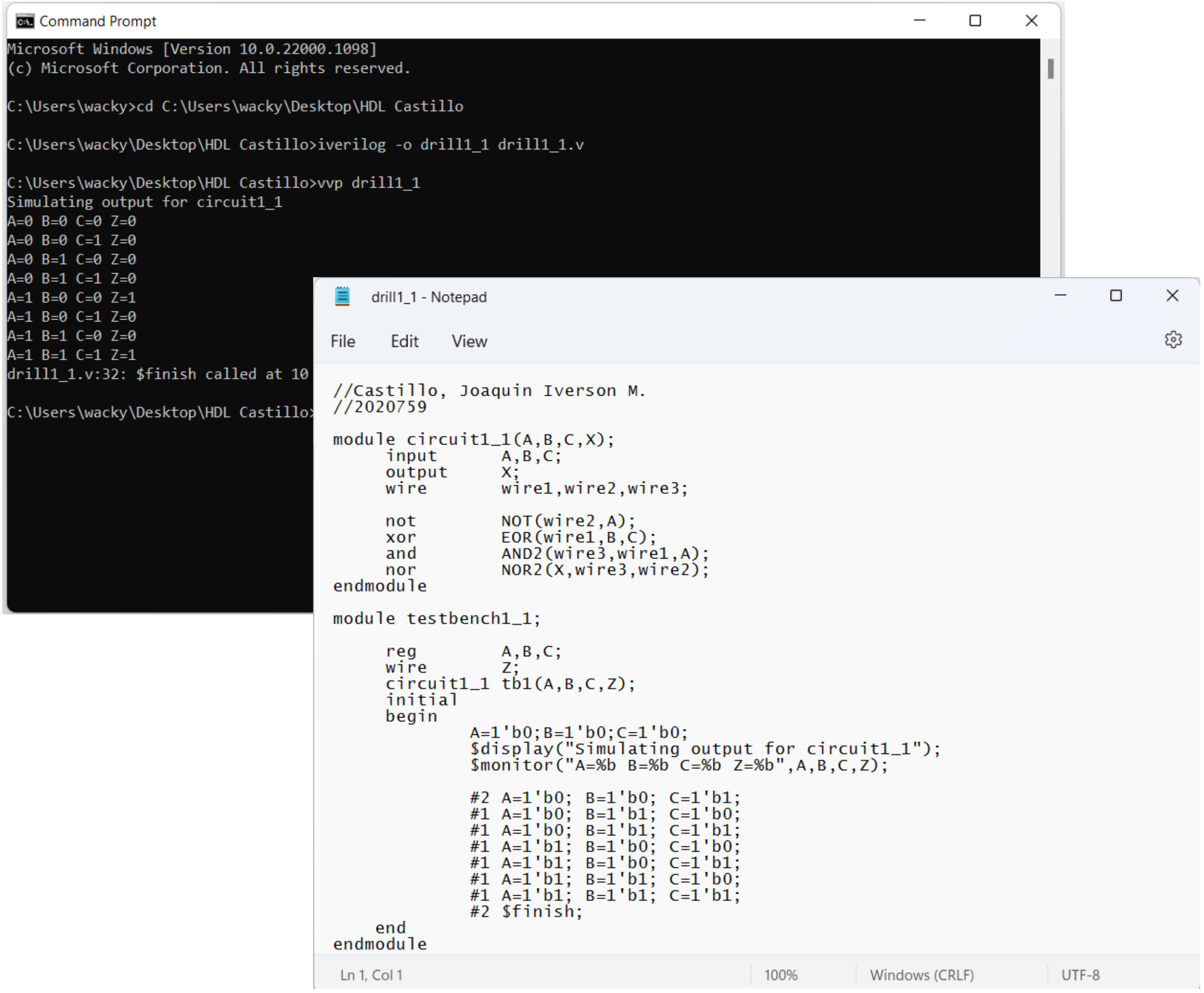
By executing a Verilog test bench file, a module is created to be tested. Hardware implementation is not available for this test bench. Our proposed hardware design is simulated using a software simulator that injects input stimuli into instantiated modules. In contrast to traditional synthesizable Verilog, test benches do not need to be synthesized.

3. Can we instantiate a test bench from another test bench? Why or why not?

No, if multiple test benches are connected to one another, inputs and outputs may have conflict.

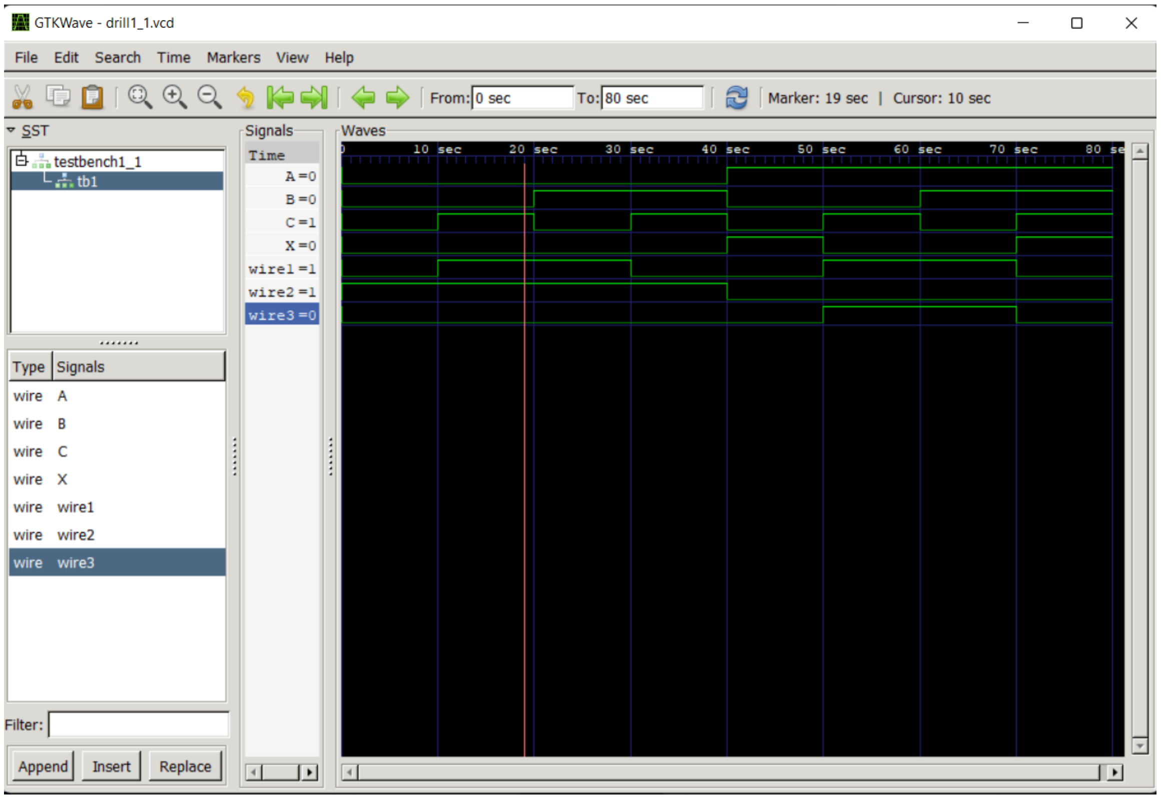
Results and Discussion:

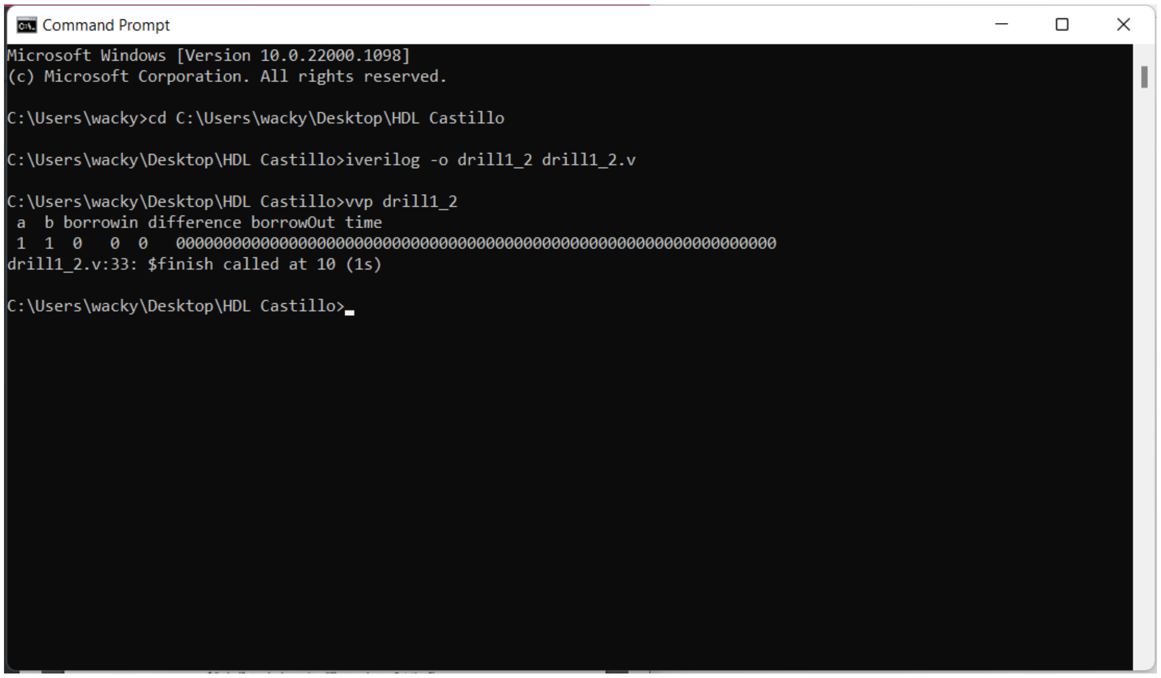
Although I had some difficulties when carrying out the exercises for this experiment, I was able to resolve them and complete the task. I successfully passed the Iverilog Icarus path to the environment variable in my computer's settings so that the application would work in the command prompt. It was challenging to type in the command prompt because each keyword entered must be correct for the experiment to work. As I conducted the experiment, I captured some screenshots, and everything is now ready for submission.



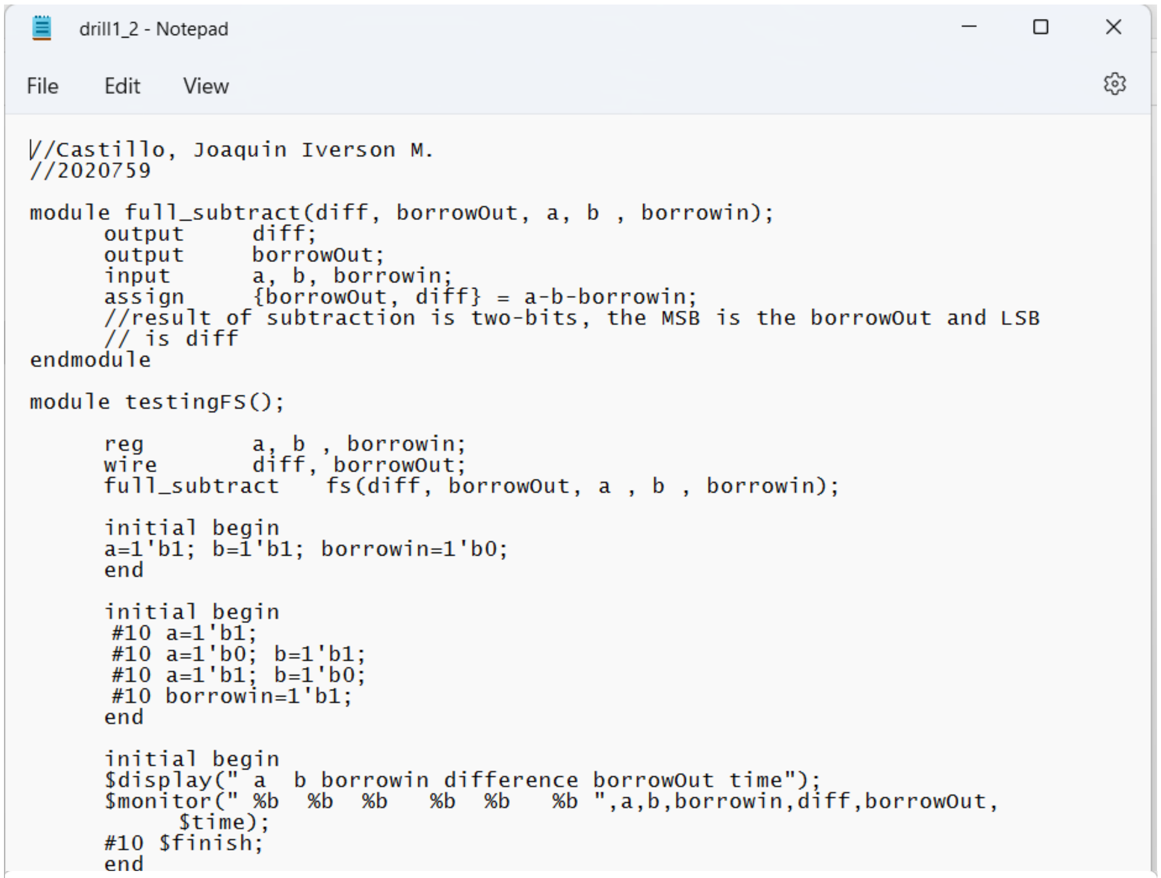
```
Microsoft Windows [Version 10.0.22000.1098]
(c) Microsoft Corporation. All rights reserved.

C:\Users\wacky>cd C:\Users\wacky\Desktop\HDL Castillo

C:\Users\wacky\Desktop\HDL Castillo>iverilog -o drill1_1 drill1_1.v

C:\Users\wacky\Desktop\HDL Castillo>vvp drill1_1
Simulating output for circuit1_1
A=0 B=0 C=0 Z=0
A=0 B=0 C=1 Z=0
A=0 B=1 C=0 Z=0
A=0 B=1 C=1 Z=0
A=1 B=0 C=0 Z=1
A=1 B=0 C=1 Z=0
A=1 B=1 C=0 Z=0
A=1 B=1 C=1 Z=1
drill1_1.v:32: $finish called at 10

C:\Users\wacky\Desktop\HDL Castillo>
```

```
//Castillo, Joaquin Iverson M.
//2020759

module circuit1_1(A,B,C,X);
     input      A,B,C;
     output     X;
     wire       wire1,wire2,wire3;

     not        NOT(wire2,A);
     xor        EOR(wire1,B,C);
     and        AND2(wire3,wire1,A);
     nor        NOR2(X,wire3,wire2);
endmodule

module testbench1_1;

     reg        A,B,C;
     wire       Z;
     circuit1_1 tb1(A,B,C,Z);
     initial
     begin
             A=1'b0;B=1'b0;C=1'b0;
             $display("Simulating output for circuit1_1");
             $monitor("A=%b B=%b C=%b Z=%b",A,B,C,Z);

             #2 A=1'b0; B=1'b0; C=1'b1;
             #1 A=1'b0; B=1'b1; C=1'b0;
             #1 A=1'b0; B=1'b1; C=1'b1;
             #1 A=1'b1; B=1'b0; C=1'b0;
             #1 A=1'b1; B=1'b0; C=1'b1;
             #1 A=1'b1; B=1'b1; C=1'b0;
             #1 A=1'b1; B=1'b1; C=1'b1;
             #2 $finish;
    end
endmodule
```





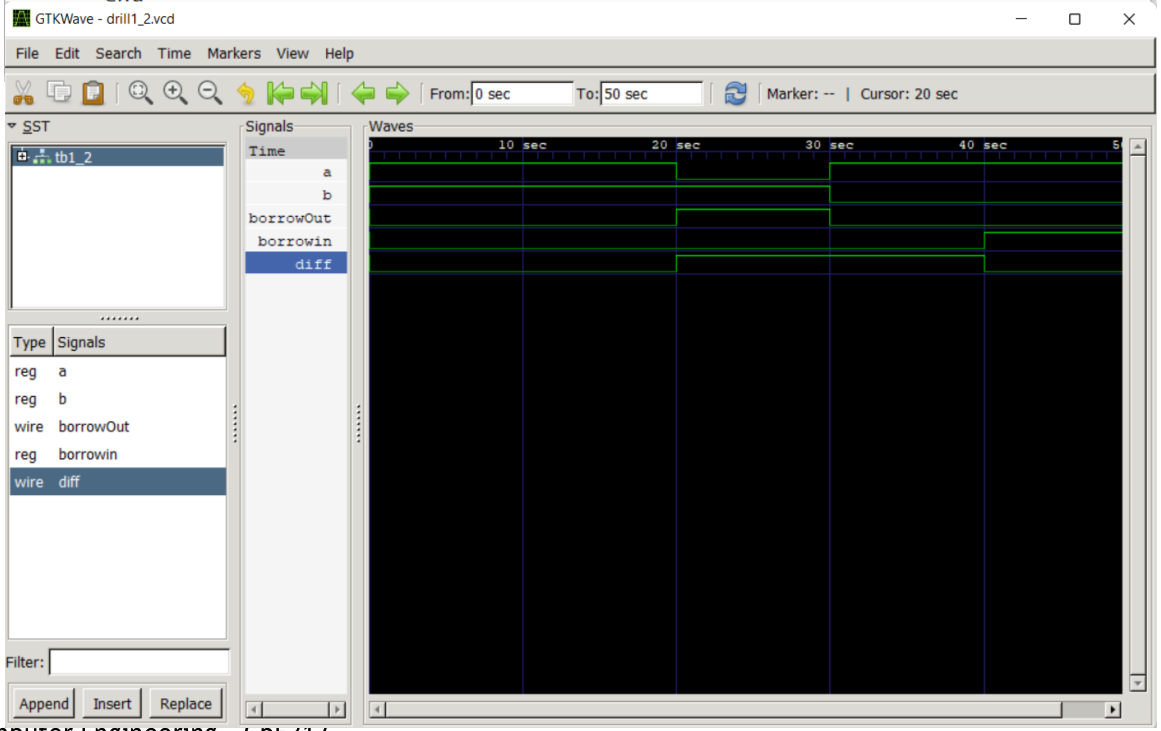
```
Microsoft Windows [Version 10.0.22000.1098]
(c) Microsoft Corporation. All rights reserved.

C:\Users\wacky>cd C:\Users\wacky\Desktop\HDL Castillo

C:\Users\wacky\Desktop\HDL Castillo>iverilog -o drill1_2 drill1_2.v

C:\Users\wacky\Desktop\HDL Castillo>vvp drill1_2
 a  b borrowin difference borrowOut time
 1  1  0    0  0   0000000000000000000000000000000000000000000000000000000000000000
drill1_2.v:33: $finish called at 10 (1s)

C:\Users\wacky\Desktop\HDL Castillo>_
```

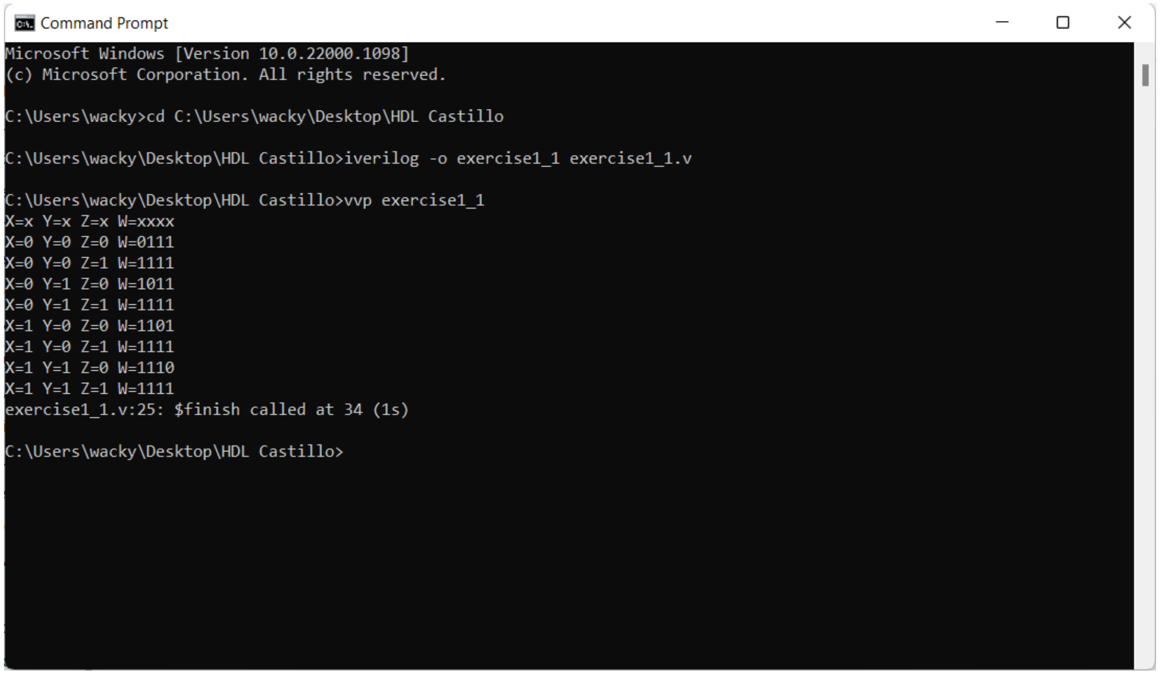


drill1_2 - Notepad

File Edit View

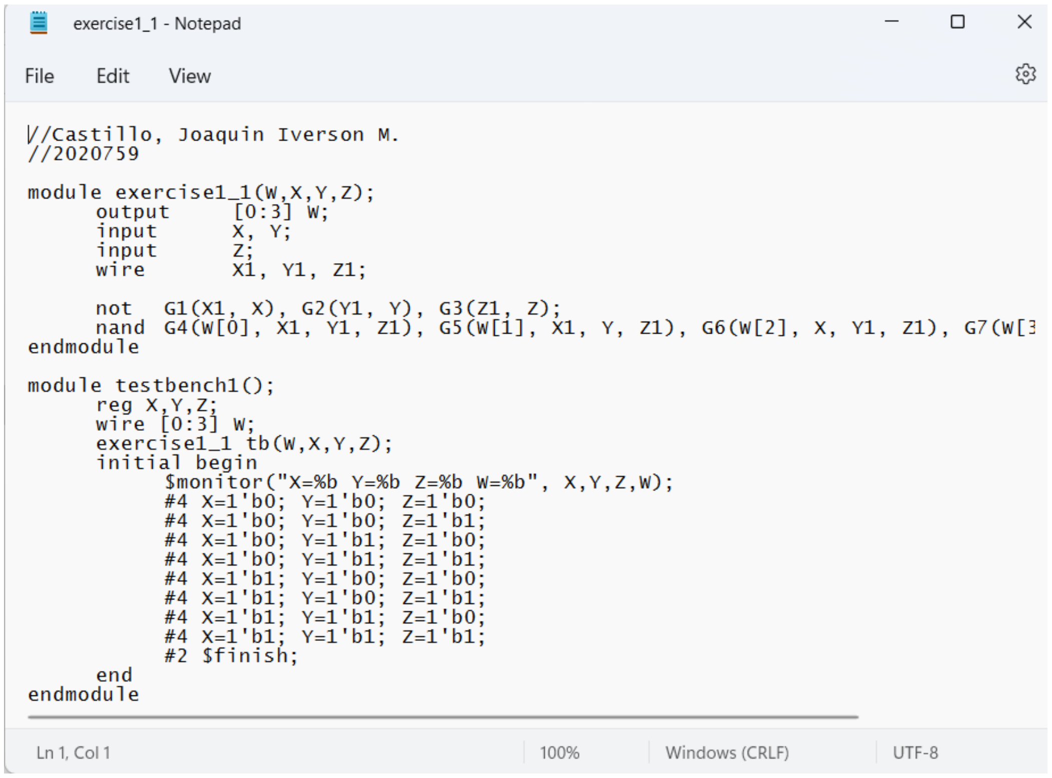
```
//Castillo, Joaquin Iverson M.
//2020759

module full_subtract(diff, borrowOut, a, b , borrowin);
       output     diff;
       output     borrowOut;
       input      a, b, borrowin;
       assign     {borrowOut, diff} = a-b-borrowin;
       //result of subtraction is two-bits, the MSB is the borrowOut and LSB
       // is diff
endmodule

module testingFS();

       reg        a, b , borrowin;
       wire       diff, borrowOut;
       full_subtract   fs(diff, borrowOut, a , b , borrowin);

       initial begin
       a=1'b1; b=1'b1; borrowin=1'b0;
       end

       initial begin
        #10 a=1'b1;
        #10 a=1'b0; b=1'b1;
        #10 a=1'b1; b=1'b0;
        #10 borrowin=1'b1;
       end

       initial begin
       $display(" a  b borrowin difference borrowOut time");
       $monitor(" %b  %b  %b   %b  %b   %b ",a,b,borrowin,diff,borrowOut,
           $time);
       #10 $finish;
       end
```





Command Prompt

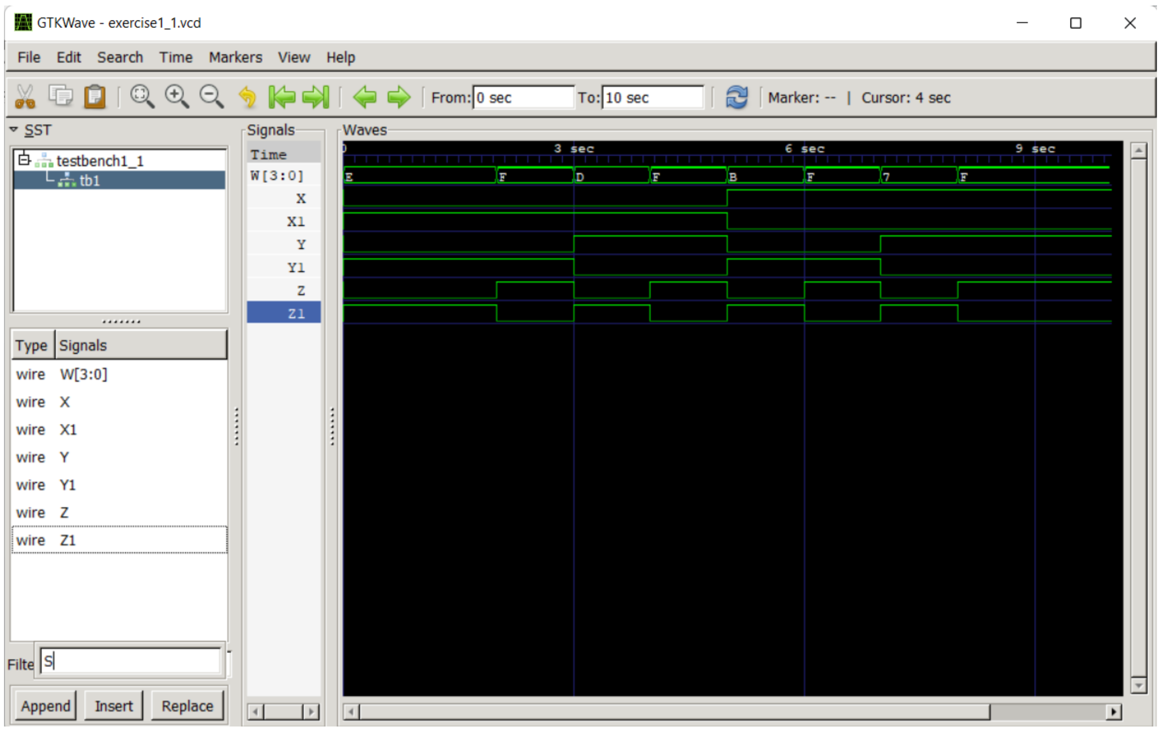
```
Microsoft Windows [Version 10.0.22000.1098]
(c) Microsoft Corporation. All rights reserved.

C:\Users\wacky>cd C:\Users\wacky\Desktop\HDL Castillo

C:\Users\wacky\Desktop\HDL Castillo>iverilog -o exercise1_1 exercise1_1.v

C:\Users\wacky\Desktop\HDL Castillo>vvp exercise1_1
X=x Y=x Z=x W=xxxx
X=0 Y=0 Z=0 W=0111
X=0 Y=0 Z=1 W=1111
X=0 Y=1 Z=0 W=1011
X=0 Y=1 Z=1 W=1111
X=1 Y=0 Z=0 W=1101
X=1 Y=0 Z=1 W=1111
X=1 Y=1 Z=0 W=1110
X=1 Y=1 Z=1 W=1111
exercise1_1.v:25: $finish called at 34 (1s)

C:\Users\wacky\Desktop\HDL Castillo>
```

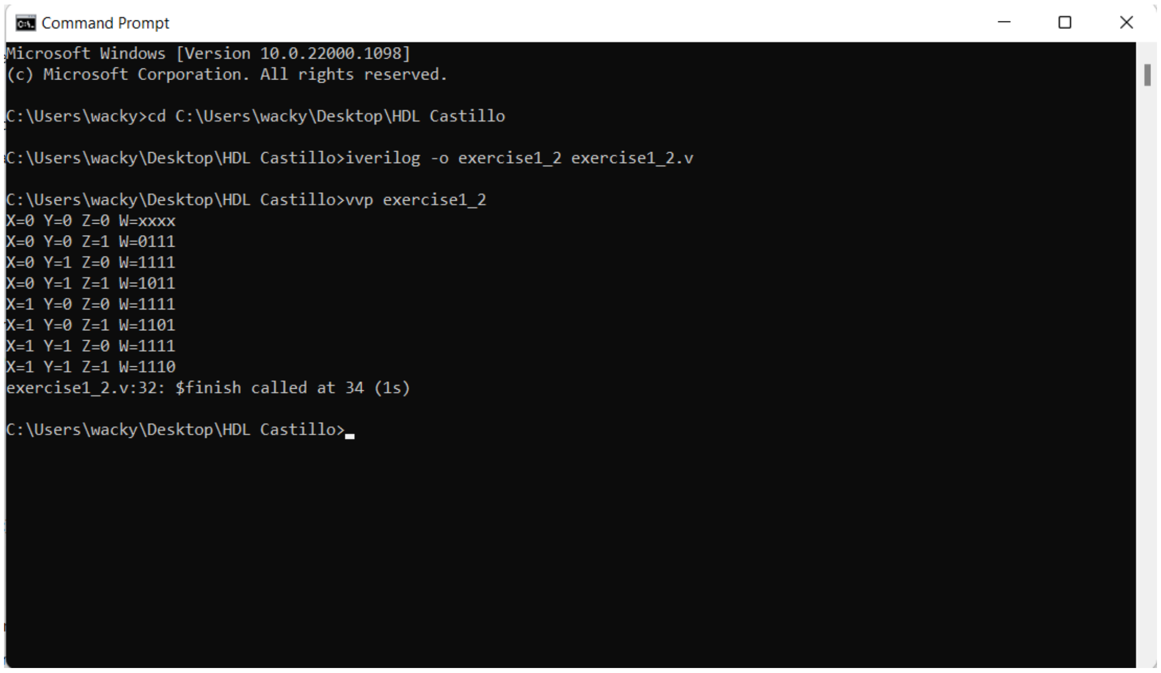


exercise1_1 - Notepad

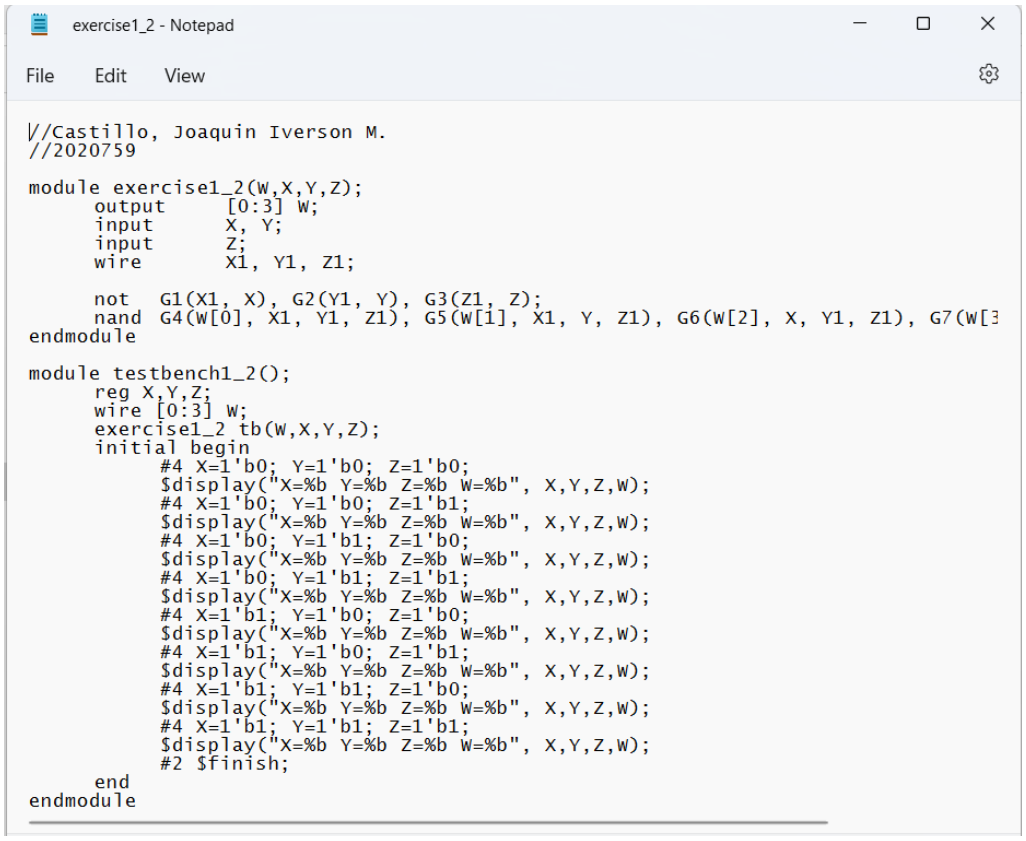
```
//Castillo, Joaquin Iverson M.
//2020759

module exercise1_1(W,X,Y,Z);
      output     [0:3] W;
      input      X, Y;
      input      Z;
      wire       X1, Y1, Z1;

      not   G1(X1, X), G2(Y1, Y), G3(Z1, Z);
      nand  G4(W[0], X1, Y1, Z1), G5(W[1], X1, Y, Z1), G6(W[2], X, Y1, Z1), G7(W[3
endmodule

module testbench1();
      reg X,Y,Z;
      wire [0:3] W;
      exercise1_1 tb(W,X,Y,Z);
      initial begin
            $monitor("X=%b Y=%b Z=%b W=%b", X,Y,Z,W);
            #4 X=1'b0; Y=1'b0; Z=1'b0;
            #4 X=1'b0; Y=1'b0; Z=1'b1;
            #4 X=1'b0; Y=1'b1; Z=1'b0;
            #4 X=1'b0; Y=1'b1; Z=1'b1;
            #4 X=1'b1; Y=1'b0; Z=1'b0;
            #4 X=1'b1; Y=1'b0; Z=1'b1;
            #4 X=1'b1; Y=1'b1; Z=1'b0;
            #4 X=1'b1; Y=1'b1; Z=1'b1;
            #2 $finish;
      end
endmodule
```

Ln 1, Col 1 | 100% | Windows (CRLF) | UTF-8



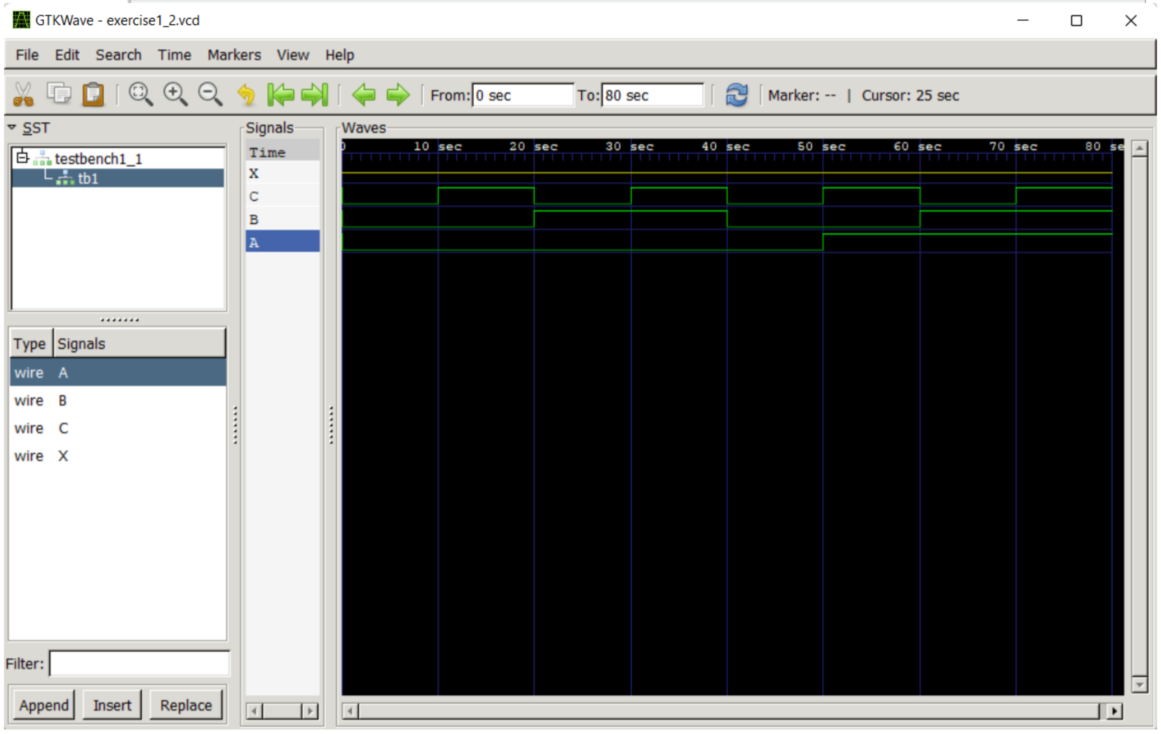


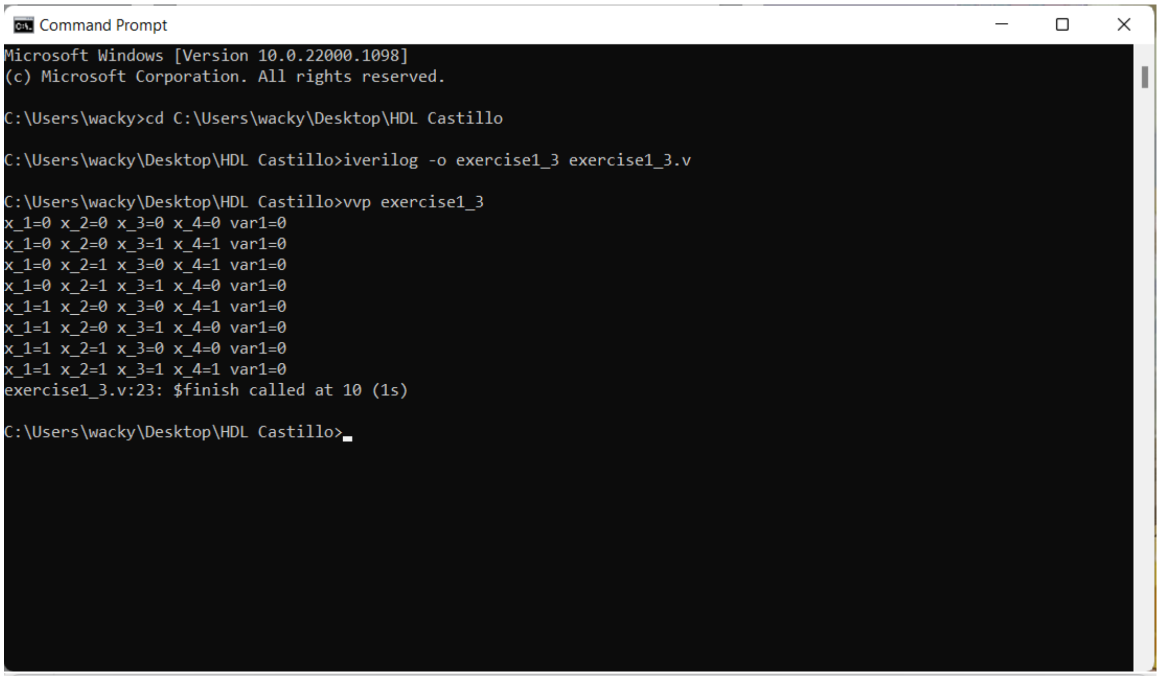
```
Microsoft Windows [Version 10.0.22000.1098]
(c) Microsoft Corporation. All rights reserved.

C:\Users\wacky>cd C:\Users\wacky\Desktop\HDL Castillo

C:\Users\wacky\Desktop\HDL Castillo>iverilog -o exercise1_2 exercise1_2.v

C:\Users\wacky\Desktop\HDL Castillo>vvp exercise1_2
X=0 Y=0 Z=0 W=xxxx
X=0 Y=0 Z=1 W=0111
X=0 Y=1 Z=0 W=1111
X=0 Y=1 Z=1 W=1011
X=1 Y=0 Z=0 W=1111
X=1 Y=0 Z=1 W=1101
X=1 Y=1 Z=0 W=1111
X=1 Y=1 Z=1 W=1110
exercise1_2.v:32: $finish called at 34 (1s)

C:\Users\wacky\Desktop\HDL Castillo>
```



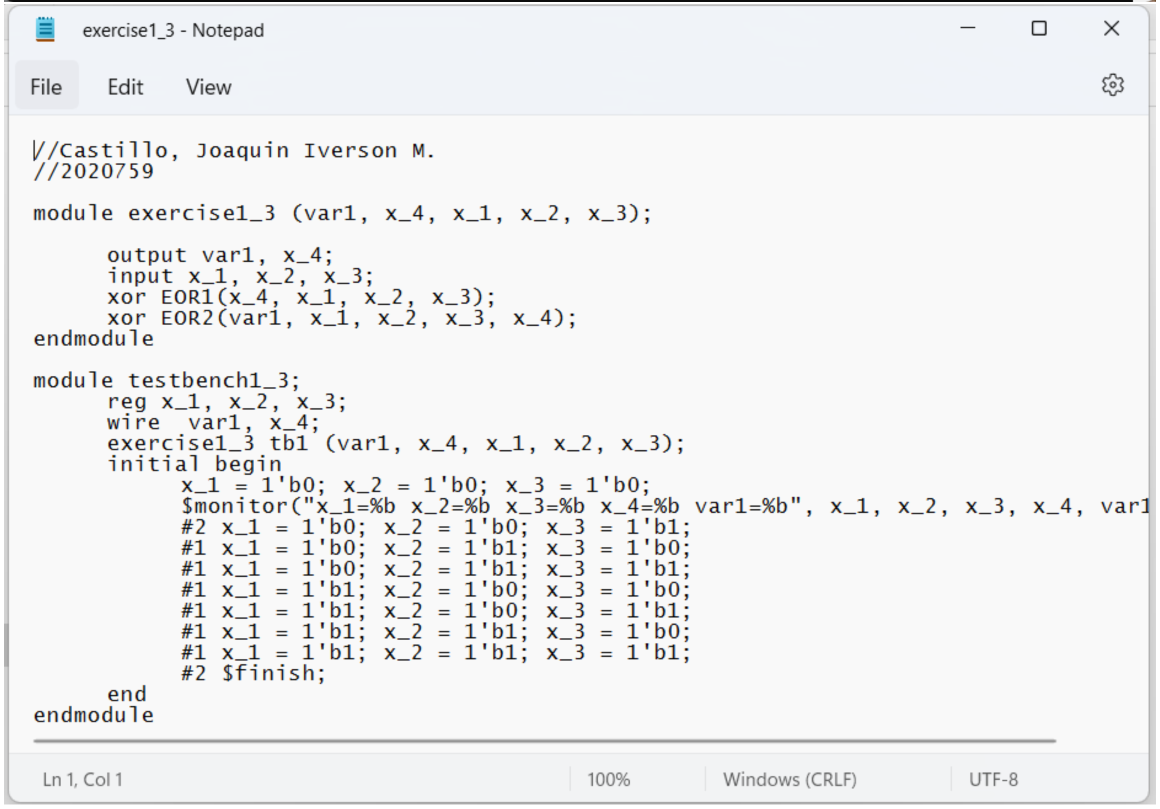
exercise1_2 - Notepad

```
//Castillo, Joaquin Iverson M.
//2020759

module exercise1_2(W,X,Y,Z);
      output     [0:3] W;
      input      X, Y;
      input      Z;
      wire       X1, Y1, Z1;

      not   G1(X1, X), G2(Y1, Y), G3(Z1, Z);
      nand  G4(W[0], X1, Y1, Z1), G5(W[1], X1, Y, Z1), G6(W[2], X, Y1, Z1), G7(W[3
endmodule

module testbench1_2();
      reg X,Y,Z;
      wire [0:3] W;
      exercise1_2 tb(W,X,Y,Z);
      initial begin
            #4 X=1'b0; Y=1'b0; Z=1'b0;
            $display("X=%b Y=%b Z=%b W=%b", X,Y,Z,W);
            #4 X=1'b0; Y=1'b0; Z=1'b1;
            $display("X=%b Y=%b Z=%b W=%b", X,Y,Z,W);
            #4 X=1'b0; Y=1'b1; Z=1'b0;
            $display("X=%b Y=%b Z=%b W=%b", X,Y,Z,W);
            #4 X=1'b0; Y=1'b1; Z=1'b1;
            $display("X=%b Y=%b Z=%b W=%b", X,Y,Z,W);
            #4 X=1'b1; Y=1'b0; Z=1'b0;
            $display("X=%b Y=%b Z=%b W=%b", X,Y,Z,W);
            #4 X=1'b1; Y=1'b0; Z=1'b1;
            $display("X=%b Y=%b Z=%b W=%b", X,Y,Z,W);
            #4 X=1'b1; Y=1'b1; Z=1'b0;
            $display("X=%b Y=%b Z=%b W=%b", X,Y,Z,W);
            #4 X=1'b1; Y=1'b1; Z=1'b1;
            $display("X=%b Y=%b Z=%b W=%b", X,Y,Z,W);
            #2 $finish;
      end
endmodule
```

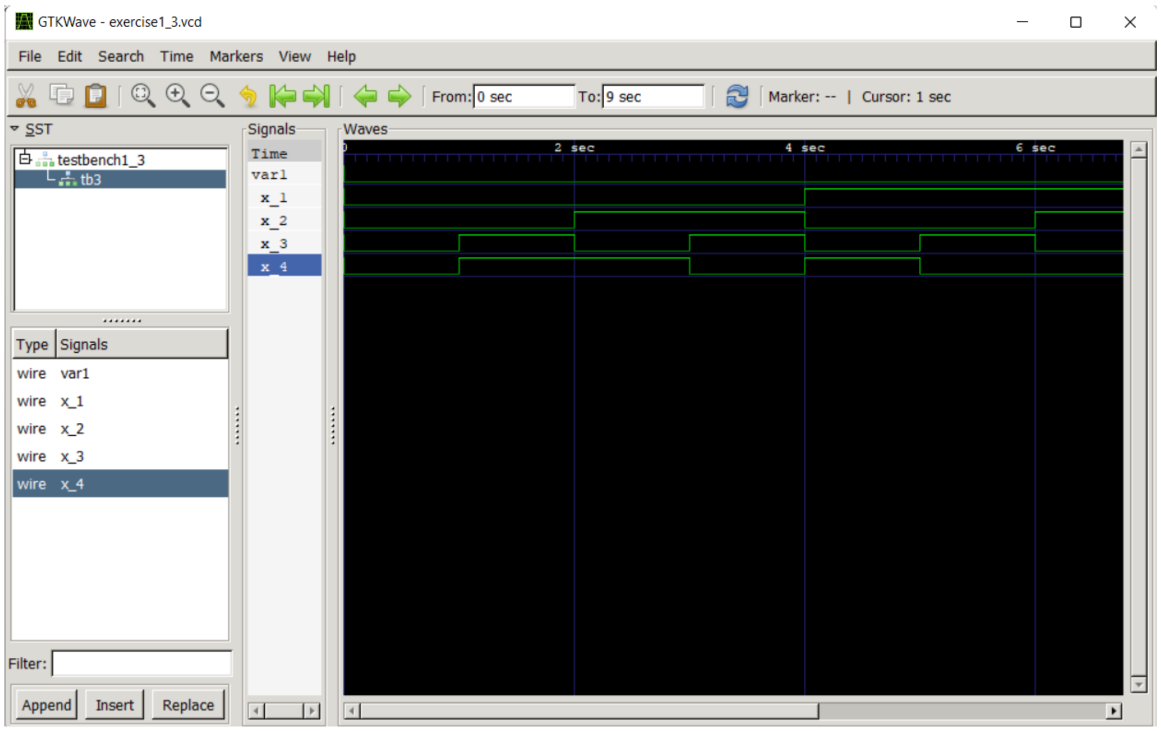




```
Microsoft Windows [Version 10.0.22000.1098]
(c) Microsoft Corporation. All rights reserved.

C:\Users\wacky>cd C:\Users\wacky\Desktop\HDL Castillo

C:\Users\wacky\Desktop\HDL Castillo>iverilog -o exercise1_3 exercise1_3.v

C:\Users\wacky\Desktop\HDL Castillo>vvp exercise1_3
x_1=0 x_2=0 x_3=0 x_4=0 var1=0
x_1=0 x_2=0 x_3=1 x_4=1 var1=0
x_1=0 x_2=1 x_3=0 x_4=1 var1=0
x_1=0 x_2=1 x_3=1 x_4=0 var1=0
x_1=1 x_2=0 x_3=0 x_4=1 var1=0
x_1=1 x_2=0 x_3=1 x_4=0 var1=0
x_1=1 x_2=1 x_3=0 x_4=0 var1=0
x_1=1 x_2=1 x_3=1 x_4=1 var1=0
exercise1_3.v:23: $finish called at 10 (1s)

C:\Users\wacky\Desktop\HDL Castillo>
```



```
//Castillo, Joaquin Iverson M.
//2020759

module exercise1_3 (var1, x_4, x_1, x_2, x_3);

      output var1, x_4;
      input x_1, x_2, x_3;
      xor EOR1(x_4, x_1, x_2, x_3);
      xor EOR2(var1, x_1, x_2, x_3, x_4);
endmodule

module testbench1_3;
      reg x_1, x_2, x_3;
      wire  var1, x_4;
      exercise1_3 tb1 (var1, x_4, x_1, x_2, x_3);
      initial begin
            x_1 = 1'b0; x_2 = 1'b0; x_3 = 1'b0;
            $monitor("x_1=%b x_2=%b x_3=%b x_4=%b var1=%b", x_1, x_2, x_3, x_4, var1
            #2 x_1 = 1'b0; x_2 = 1'b0; x_3 = 1'b1;
            #1 x_1 = 1'b0; x_2 = 1'b1; x_3 = 1'b0;
            #1 x_1 = 1'b0; x_2 = 1'b1; x_3 = 1'b1;
            #1 x_1 = 1'b1; x_2 = 1'b0; x_3 = 1'b0;
            #1 x_1 = 1'b1; x_2 = 1'b0; x_3 = 1'b1;
            #1 x_1 = 1'b1; x_2 = 1'b1; x_3 = 1'b0;
            #1 x_1 = 1'b1; x_2 = 1'b1; x_3 = 1'b1;
            #2 $finish;
      end
endmodule
```


GTKWave - exercise1_3.vcd
File Edit Search Time Markers View Help
From: 0 sec To: 9 sec
Marker: -- | Cursor: 1 sec
SST
testbench1_3
tb3
Signals
Time
var1
x_1
x_2
x_3
x_4
Waves
2 sec
4 sec
6 sec
Type Signals
wire var1
wire x_1
wire x_2
wire x_3
wire x_4
Filter:
Append Insert Replace

Conclusion:

All things considered, I discovered a lot about this experiment, including how the test benches function and what they were employed for. There's no issue happen while installing the Icarus Verilog as the instrumentation for performing this experiment including information on initialization and providing data on the corresponding values. The .v file and the creation of the .vcd file are important for this experiment because the files have their own role to execute this experiment. And finally, I discovered how to execute a GTK wave and display its results on a time graph.

REFERENCES:

- *asic-world.com*