CSC 347/ENS 211

Title and Experiment #	Lab7: Introduction to Verilog
Name	Gianna Galard
Date Performed	02-Nov-21
Date Submitted	10-Nov-21

The student pledges this work to be their own *Gianna Galard*

Link: https://www.edaplayground.com/x/Beut

Objective:

This week's lab aims to understand the basics of Verilog and edaplayground.com by implementing a Verilog Design and simulation of a half adder and full adder that will later be simulated using a testbench.

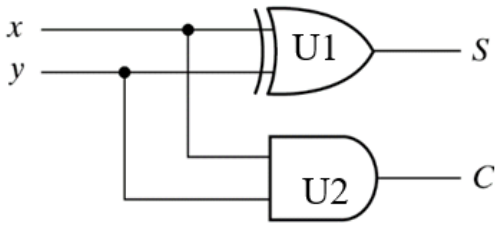
Language and Compiler used:

- Verilog
- Edaplayground.com

Design Procedure:

The students first task was to build a half-adder since a full adder uses two half-adders. This modularizes our Verilog code as we will later instantiate two half-adders inside of our full-adder module.

Figure 1. halfadder logic diagram



(b) $S = x \oplus y$
$C = xy$

Figure 2. halfadder verilog

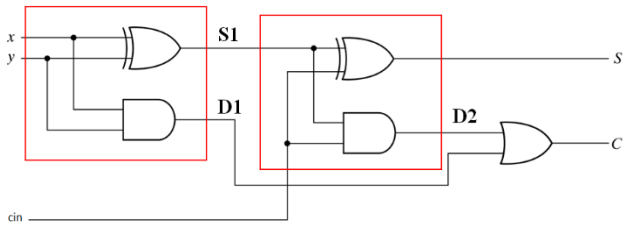
```
module halfadder (S, C, x, y);
  input x, y;
  output S, C;

//Instantiate primitive gates
  xor U1(S, x, y);
  and U2(C, x, y);

endmodule
```

The next step is to define the inputs and outputs using the keywords "input" and "output." Modules begin with the keyword "module" and end with "endmodule." After using "module," follow it with the module's name; for example, the student used "halfadder." Following, the student wrote the outputs and inputs in parenthesis. The outputs are formatted with uppercase letters, and inputs are formatted with lowercase letters. For example, S and C are our outputs, and X and Y are our inputs. Then, they are followed up with the characters declared in the module declaration.

Next, the student built a Full Adder using two Half Adders and an OR gate. The full adder requires an internal signal to declare internal outputs as wires with the "wire" keyword.

The student then finished the full adder by adding an or gate with an output of C to represent the carry and inputs of D2 and D1.

Figure 3. Full adder logic diagram



Implementation of Full Adder with Two Half Adders and an OR Gate

Figure 4. Full adder verilog code

```
module fulladder (S, C, x, y, cin);
   input x, y, cin;
   output S, C;

   //internal signals
   wire S1, D1, D2;

   //Instantiate the half adders
    halfadder    HA1 (S1, D1, x, y);
    halfadder    HA2 (S, D2, S1, cin);
    or U3(C, D2, D1);

endmodule
```

The student then wrote a test bench to test the full adder, module Test, which needed variables to be inputted. The keyword for variables is "regs." In a test module, you need to instantiate a module unit to be tested, for example:

// Instantiate the Unit Under Test (UUT)

fulladder uut (S, C, x, y, cin);

Then, to begin the test, use the keywords "initial" followed by "begin" and close it with "finish" followed by "end." The student then produced a waveform for the lab with the two statements:

```
//the following two statements are needed for creating waveforms
$dumpfile("dump.vcd");  $dumpvars(1, test);


// display the inputs and outputs
$monitor( "x + y + cin = %b + %b + %b = C S = %b %b",
x, y, cin, C, S );
```

The student then initializes outputs and then sets a delay timer with the #10 syntax, which changes the waves' length in the waveform chart.

Figure 5. Halfadder and fulladder modules

```
// Code your design here

module fulladder (S, C, x, y, cin);
  input x, y, cin;
  output S, C;

  //internal signals
  wire S1, D1, D2;

  //Instantiate the half adders
   halfadder    HA1 (S1, D1, x, y);
   halfadder    HA2 (S, D2, S1, cin);
   or U3(C, D2, D1);

endmodule
```

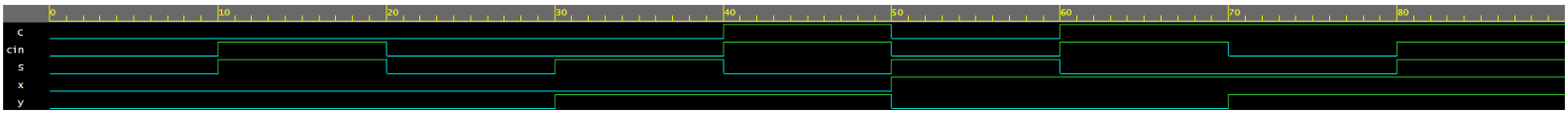
```
module halfadder (S, C, x, y);
   input x, y;
   output S, C;

//Instantiate primitive gates
   xor U1(S, x, y);
   and U2(C, x, y);

endmodule
```

Figure 6. Testbench containing module test

```
module test;

        reg x, y, cin;    // Inputs
        wire S, C;       // Outputs

        // Instantiate the Unit Under Test (UUT)
        fulladder uut (S, C, x, y, cin);

        initial
          begin
 //the following 2 statements are needed for producing waveform in
edaplayground.com
$dumpfile("dump.vcd");  $dumpvars(1, test);

                // display the inputs and outputs
                $monitor( "x + y + cin = %b + %b + %b = C S = %b %b",
x, y, cin, C, S );

               // Initialize Inputs
                      x = 0;   y = 0;      cin = 0;
                 #10 x = 0;   y = 0;      cin = 1;
```

```
        #10 x = 0;   y = 0;    cin = 0;
        #10 x = 0;   y = 1;    cin = 0;
        #10 x = 0;   y = 1;    cin = 1;
        #10 x = 1;   y = 0;    cin = 0;
        #10 x = 1;   y = 0;    cin = 1;
        #10 x = 1;   y = 1;    cin = 0;
        #10 x = 1;   y = 1;    cin = 1;

        #10 $finish;

    end
endmodule
```

Figure 7. Waveform from test with 10 second delay



Conclusion:

In conclusion, the student instantiated two half adders inside a full adder module and then added an OR gate inside the fullAdder module to complete the full adder implementation. Then, the student wrote a testbench where they declared inputs and internal outputs. The final output of the testbench test module was the following :

Figure 8. Testbench output

```
VCD info: dumpfile dump.vcd opened for output.
x + y + cin = 0 + 0 + 0 = C S = 0 0
x + y + cin = 0 + 0 + 1 = C S = 0 1
x + y + cin = 0 + 0 + 0 = C S = 0 0
x + y + cin = 0 + 1 + 0 = C S = 0 1
x + y + cin = 0 + 1 + 1 = C S = 1 0
x + y + cin = 1 + 0 + 0 = C S = 0 1
x + y + cin = 1 + 0 + 1 = C S = 1 0
x + y + cin = 1 + 1 + 0 = C S = 1 0
x + y + cin = 1 + 1 + 1 = C S = 1 1
Finding VCD file...
./dump.vcd
```

Homework:

Write Verilog design and test bench codes for a 2-bit multiplier based on the following diagram.

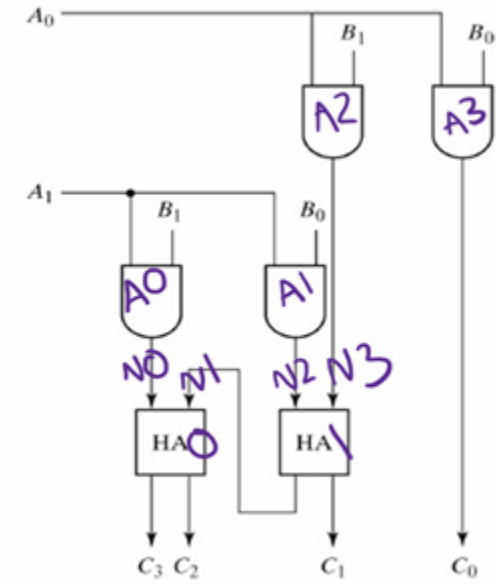


Figure 9. design.sv

```
// AUTHOR : GIANNA GALARD
// LAB 7: INTRO TO VERILOG
// https://www.edaplayground.com/x/Beut

// Code your design here
module multiplier(A0, A1, B0, B1, C0, C1, C2, C3);
   input A0, A1, B0, B1;
   output C0, C1, C2, C3;

  //internal signals (labeled on diagram)
   wire N0, N1, N2, N3;

   //Instantiate the half adders
  halfadder    HA0 (C3,C2,N0,N1);
```

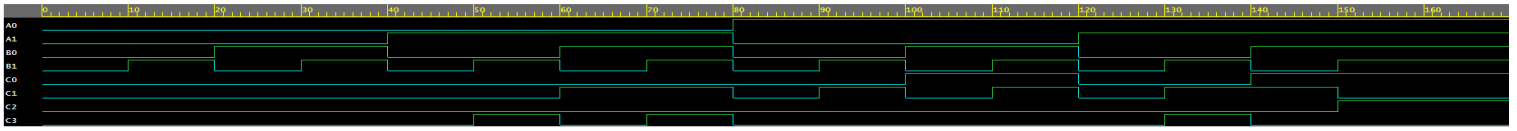
```
  halfadder    HA1 (C1,N1,N2,N3);
  and A0(N0, A1, B1);
  and A1(N2, A1, B0);
  and A2(N3, A0, B1);
  and A3(C0, A0, B0);

endmodule

module halfadder (S, C, x, y);
   input x, y;
   output S, C;

//Instantiate primitive gates
   xor U1(S, x, y);
   and U2(C, x, y);

endmodule
```

Figure 10. test.sv

```
// AUTHOR : GIANNA GALARD
// LAB 7: INTRO TO VERILOG
// https://www.edaplayground.com/x/Beut

module test;

        reg A0, A1, B0, B1;    // Inputs
        wire C0, C1, C2, C3;       // Outputs

        // Instantiate the Unit Under Test (UUT)
  multiplier uut (A0, A1, B0, B1, C0, C1, C2, C3);

        initial
          begin
```

```
//the following 2 statements are needed for producing waveform in
edaplayground.com
$dumpfile("dump.vcd");  $dumpvars(1, test);

        // display the inputs and outputs
        $monitor("%b %b %b %b %b %b %b %b", A0, A1, B0, B1,
C0, C1, C2, C3);

        for(int i = 0; i < 16; i = i + 1) begin
          {A0, A1, B0, B1} = i;
          #10; // 10 delay units
        end
         #10 $finish;

        end
   endmodule
```

Figure 11. Waveforms



LINKS:

FULL ADDER https://www.edaplayground.com/x/wUsp

MULTIPLIER https://www.edaplayground.com/x/Beut