LAB 1 EE4301

ALEX LEMA

Lemac001

Lab Notebook deliverables:

• Brief description of what you did in the lab, what steps were followed

The goal of this lab is to create a full adder adding binary numbers together with the carry amounts. A full adder of a bit adds three bits, often written as A, B, and Cin; being A and B the addends and Cin the carry that comes from the previous less significant stage. The circuit produces a two-bit output, as does the semi-adder, called the carry-out (Cout) and sum S. A full adder can be implemented in many ways, in this case is made up of gates.

• Discussion of results:

a. What were the test cases you used and discuss why you chose the set of test cases?

In this case I changed some values of A, B from the original values of the given test bench. The final summation S and Cout values are correct.

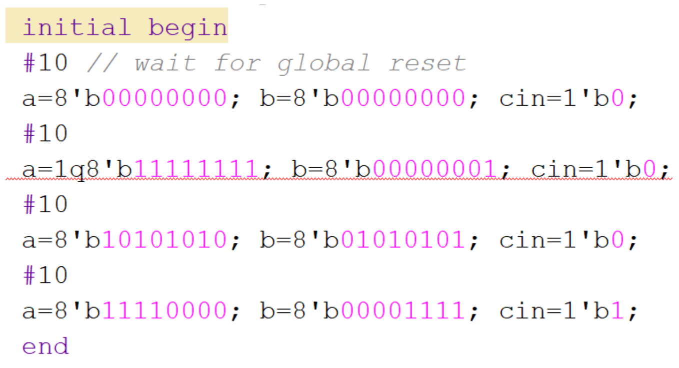
b. If you had to modify your design to meet your design discuss what methodology you followed, etc.

For my project to work I had to fix some bugs. The first was the file full_adder.v, because it was missing this line of code: "input cin." It was difficult to add the test bench file because it gave me simulation errors because I tried to add file instead of creating a file. I had to delete all the files related to the test bench in the directory. I created the file and pasted the given code and it worked correctly when I ran the simulation.

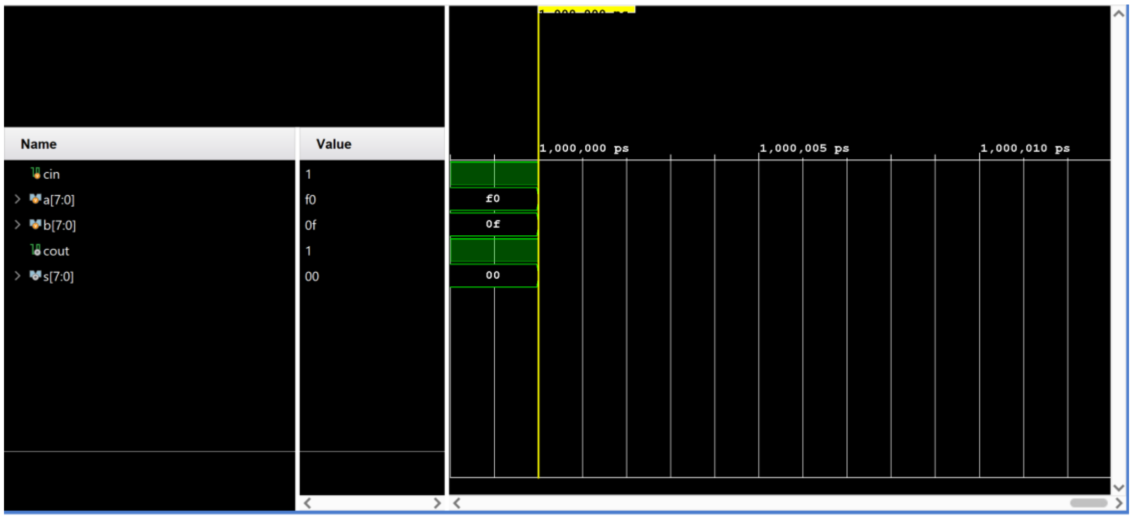
c. Attach snippet of the reports of the tool.

i. Table and Simulation waveforms showing all the testcases used above.

EXAMPLE 1 with given values



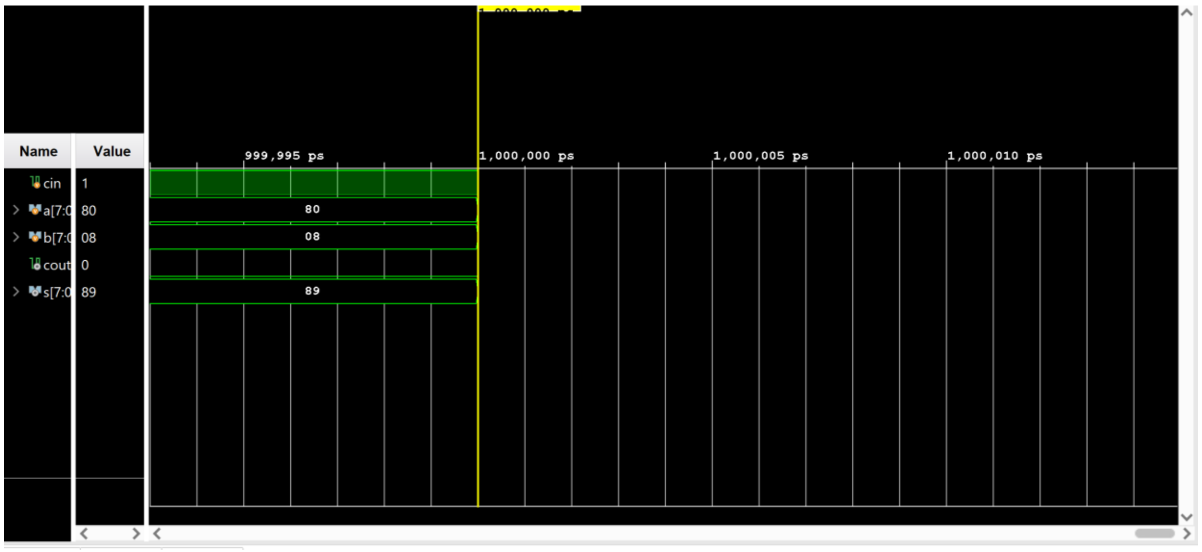
```
initial begin
#10 // wait for global reset
a=8'b00000000; b=8'b00000000; cin=1'b0;
#10
a=1q8'b11111111; b=8'b00000001; cin=1'b0;
#10
a=8'b10101010; b=8'b01010101; cin=1'b0;
#10
a=8'b11110000; b=8'b00001111; cin=1'b1;
end
```



```
cin      1
a[7:0]   f0
b[7:0]   0f
cout     1
s[7:0]   00
```

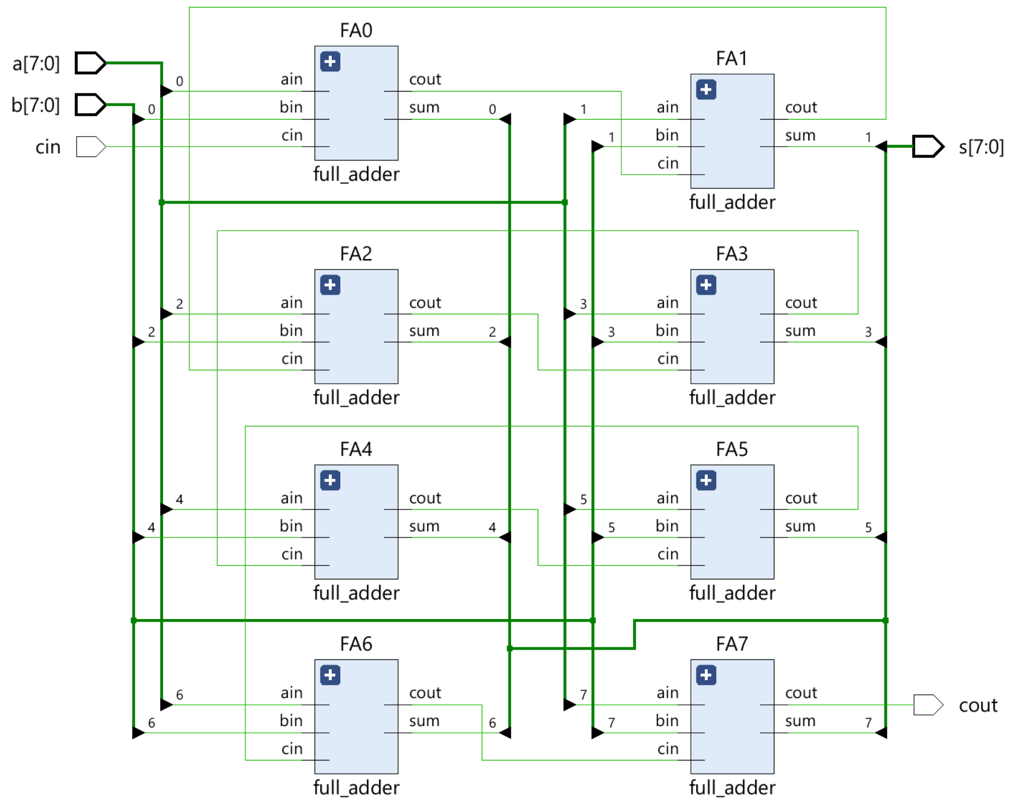
EXAMPLE 2 with different values in test bench

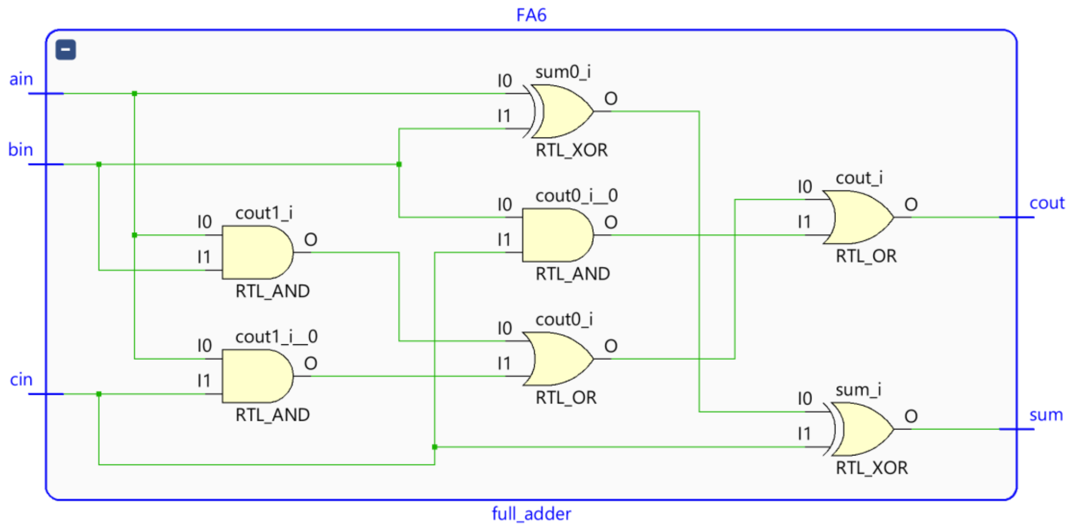
```
a=8'b00000000; b=8'b00000000; cin=1'b0;
#10
a=8'b11111111; b=8'b00000001; cin=1'b0;
#10
a=8'b10101010; b=8'b01010101; cin=1'b0;
#10
a=8'b10000000; b=8'b00001000; cin=1'b1;
```



```
cin      1
a[7:0    80
b[7:0    08
cout     0
s[7:0    89
```

ii. Schematic screenshot of synthesized design.





iii. Utilization report from synthesis

```
1. Slice Logic
--------------

+-------------------------+------+-------+-----------+-------+
|        Site Type        | Used | Fixed | Available | Util% |
+-------------------------+------+-------+-----------+-------+
| Slice LUTs*             |    8 |     0 |     20800 |  0.04 |
|   LUT as Logic          |    8 |     0 |     20800 |  0.04 |
|   LUT as Memory         |    0 |     0 |      9600 |  0.00 |
| Slice Registers         |    0 |     0 |     41600 |  0.00 |
|   Register as Flip Flop |    0 |     0 |     41600 |  0.00 |
|   Register as Latch     |    0 |     0 |     41600 |  0.00 |
| F7 Muxes                |    0 |     0 |     16300 |  0.00 |
| F8 Muxes                |    0 |     0 |      8150 |  0.00 |
+-------------------------+------+-------+-----------+-------+
```

NOTE: Only attach relevant sections, not the entire report.

• Summary or Conclusion:

a. Were all design goals met?

My goal was that I have errors when executing this task and thus try to correct them and learn more from Verilog. In this laboratory we satisfactorily study the operation of digital circuits capable of doing the arithmetic sum of 8-bit input.

b. What were the difficulties encountered?

The difficulty was finding where the errors are. It was difficult to navigate and found the errors, this is not like other programming languages like java or phyton.

Everything seems to be working fine so far in "Run Simulation", where it can give many errors.

c. Any improvement you would like to make time permitting? Etc.…

In the manual it mentions about the bech test file, but there is no place on canvas to download. In the manual it should say create file instead of add file.

• Verilog Source code: (Font: Courier, size 8pt, single line spacing, no paragraph spacing) • All files must be labeled (ex. Figure 1: Simulation of eight-bit adder…). You should also highlight more interesting parts of your attachments.

-full_adder.v

```
//`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 09/24/2020 12:54:38 PM
// Design Name:
// Module Name: full_adder
// Project Name:
// Target Devices:
```

```
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
module full_adder(
output cout,
output sum,
input ain,
input bin,
input cin
);
assign sum = ain^bin^cin;
assign cout= (ain&bin)|(ain&cin)|(bin&cin);
endmodule
```

-adder8.v

```
//`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 09/24/2020 12:59:13 PM
// Design Name:
// Module Name: adder8
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module adder8(
 output cout,
 output [7:0] s,
 input [7:0] a,
 input [7:0] b,
 input cin
   );

   wire [7:1] carry;
   full_adder FA0(carry[1],s[0],a[0],b[0],cin);
   full_adder FA1(carry[2],s[1],a[1],b[1],carry[1]);
   full_adder FA2(carry[3],s[2],a[2],b[2],carry[2]);
   full_adder FA3(carry[4],s[3],a[3],b[3],carry[3]);
   full_adder FA4(carry[5],s[4],a[4],b[4],carry[4]);
   full_adder FA5(carry[6],s[5],a[5],b[5],carry[5]);
   full_adder FA6(carry[7],s[6],a[6],b[6],carry[6]);
   full_adder FA7(cout,s[7],a[7],b[7],carry[7]);

endmodule
```

-add8.tb.v

```
`timescale 1ns / 1ps
module adder8_tb(
 );
 reg cin;
 reg [7:0] a,b;
 wire cout;
 wire [7:0] s;

 // instantiate the unit under test uut
 adder8 uut(cout,s,a,b,cin);

 // stimulus (inputs)
 initial begin
 #10 // wait for global reset
 a=8'b00000000; b=8'b00000000; cin=1'b0;
 #10
 a=8'b11111111; b=8'b00000001; cin=1'b0;
 #10
 a=8'b10101010; b=8'b01010101; cin=1'b0;
 #10
 a=8'b11110000; b=8'b00001111; cin=1'b1;

 end

 // results (outputs)
 initial begin
 #5
 $display("a = %b, b = %b, cin = %b",a,b,cin);
 $display("s = %b, cout = %b",s,cout);
 #10
 $display("a = %b, b = %b, cin = %b",a,b,cin);
 $display("s = %b, cout = %b",s,cout);
 #10
 $display("a = %b, b = %b, cin = %b",a,b,cin);
 $display("s = %b, cout = %b",s,cout);
 $display("End Simulation");
 end

endmodule
```