Quinn Roemer

Engineering – 303

Lab 2

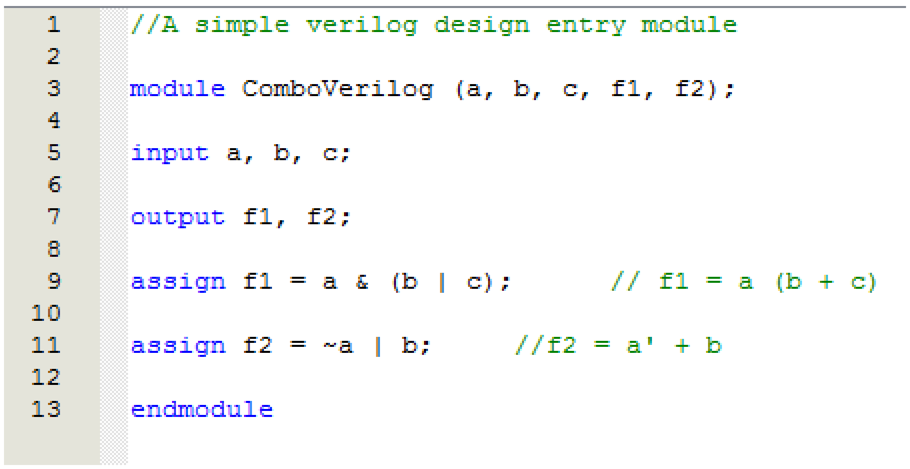
2/1/2017

Introduction/Description

The purpose of this lab was to continue to learn how to build more complex circuits in Quartus using Verilog and Block-Diagrams. I was supposed to learn how to better build more circuits, create algebra expressions from circuit diagrams, and be able to determine if two circuits are equivalent or not.

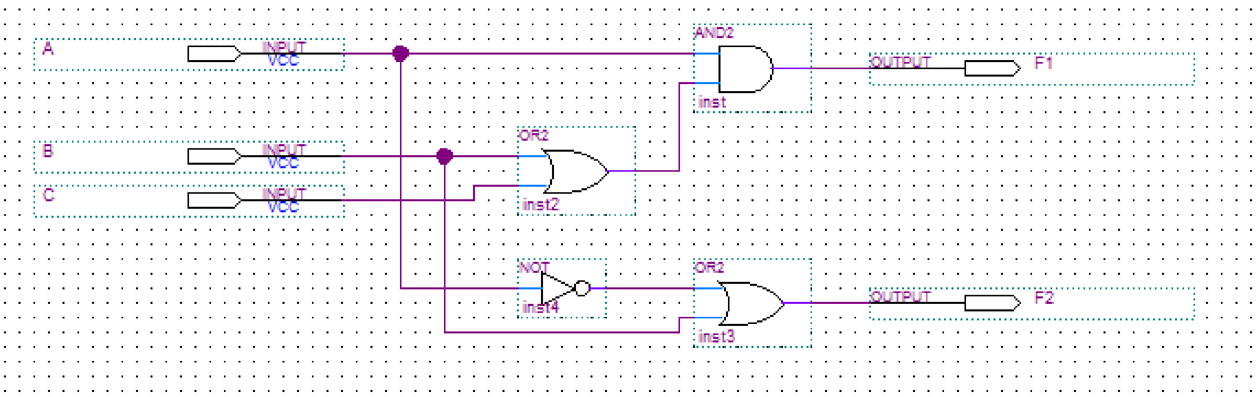
Design

In the first part of the lab I was supposed to design a simple combinatorial circuit in Verilog using a given Verilog design from the lab.



```
//A simple verilog design entry module

module ComboVerilog (a, b, c, f1, f2);

input a, b, c;

output f1, f2;

assign f1 = a & (b | c);       // f1 = a (b + c)

assign f2 = ~a | b;       //f2 = a' + b

endmodule
```

In addition I was supposed to implement the same circuit above in a Block-Diagram. This circuit was supposed to perform the same function as the one made in Verilog.

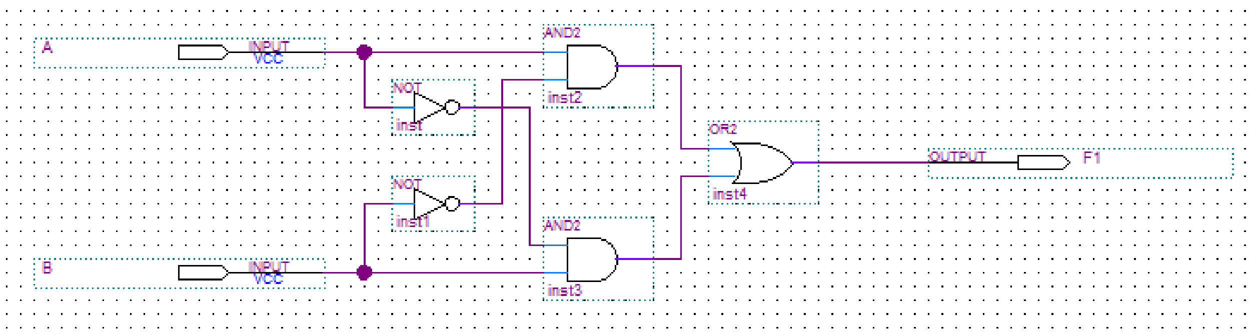


Here are a couple truth tables that encompass both outputs in both of the circuits that were made in the first part of the lab.

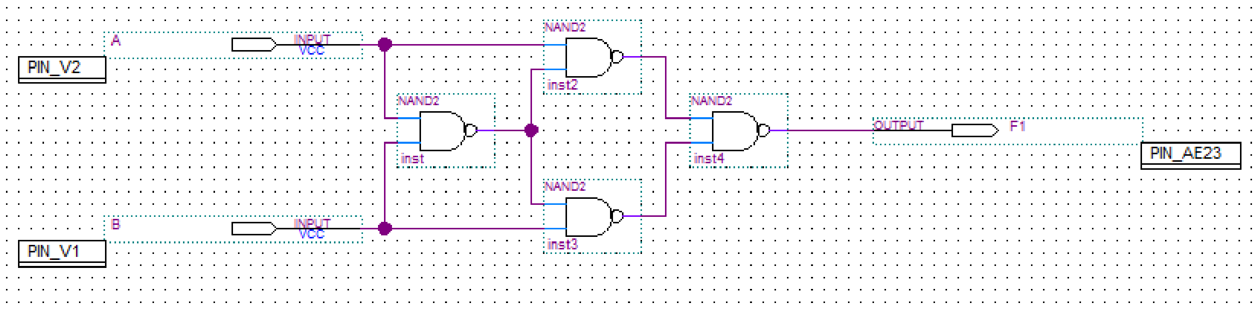
Input A	Input B	Input C	**B + C**	**F1 = A(B+C)**
0	0	0	**0**	**0**
0	0	1	**1**	**0**
0	1	0	**1**	**0**
0	1	1	**1**	**0**
1	0	0	**0**	**0**
1	0	1	**1**	**1**
1	1	0	**1**	**1**
1	1	1	**1**	**1**

Input A	Input B	Input C	**A'**	**F2 = A' + B**
0	0	0	**1**	**1**
0	0	1	**1**	**1**
0	1	0	**1**	**1**
0	1	1	**1**	**1**
1	0	0	**0**	**0**
1	0	1	**0**	**0**
1	1	0	**0**	**1**
1	1	1	**0**	**1**

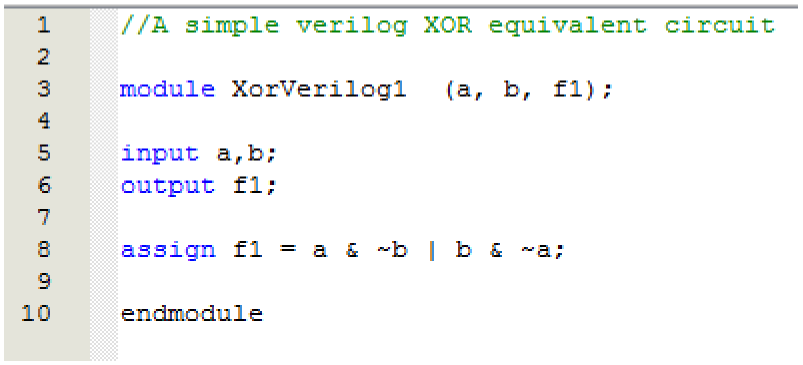
In the next part of the lab I was supposed to design a XOR equivalent gate by using a Block-Diagram. This circuit consisted of NOT gates, AND gates, and even an OR gate.



In addition, I was told to create another XOR equivalent gate. However, this one was different as it consisted of only NAND gates.

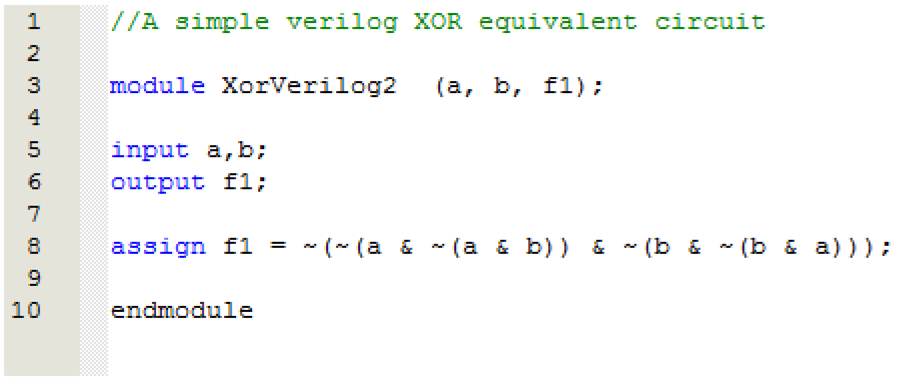


Next, I was instructed to implement both of these circuits using Verilog to program them into Quartus. The first circuit shown is for the XOR equivalent circuit that consisted of more than one type of gate.



```
//A simple verilog XOR equivalent circuit

module XorVerilog1  (a, b, f1);

input a,b;
output f1;

assign f1 = a & ~b | b & ~a;

endmodule
```

The second, is the circuit that consisted of only NAND gates.

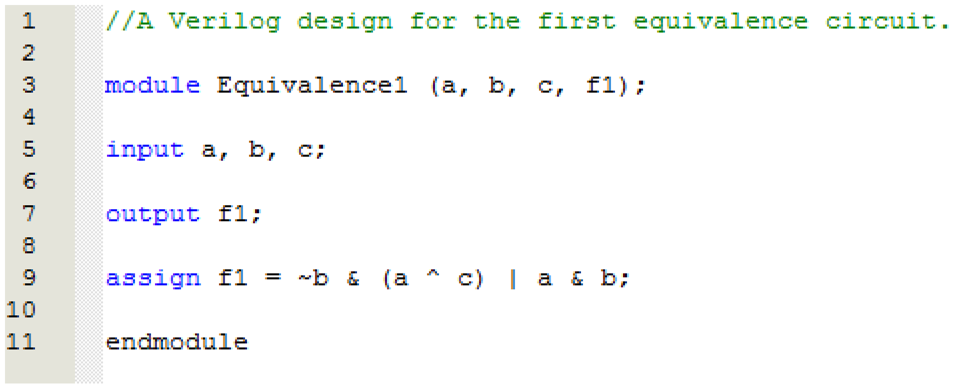


```
//A simple verilog XOR equivalent circuit

module XorVerilog2  (a, b, f1);

input a,b;
output f1;

assign f1 = ~(~(a & ~(a & b)) & ~(b & ~(b & a)));

endmodule
```

Here is the truth table that encompasses the last four circuits shown.

Input A	Input B	Output F1
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**0**

In the last part of the lab I was supposed to test two separate circuits and determine if they are equivalent or not (you will find the result of this task in the conclusion of this report). I translated the algebraic expression that I was given into Verilog, and came up with this circuit.

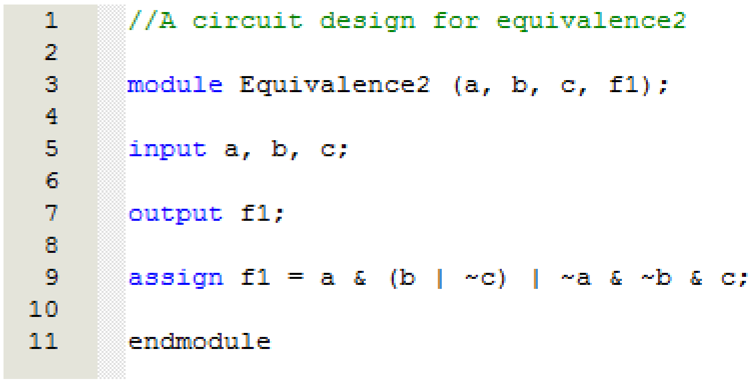


```
//A Verilog design for the first equivalence circuit.

module Equivalence1 (a, b, c, f1);

input a, b, c;

output f1;

assign f1 = ~b & (a ^ c) | a & b;

endmodule
```

Here is the truth table that encompasses this circuit.

Input A	Input B	Input C	Output F1
0	0	0	**0**
0	0	1	**1**
0	1	0	**0**
0	1	1	**0**
1	0	0	**1**
1	0	1	**0**
1	1	0	**1**
1	1	1	**1**

Next, I translated the second algebraic expression given into a Verilog circuit.



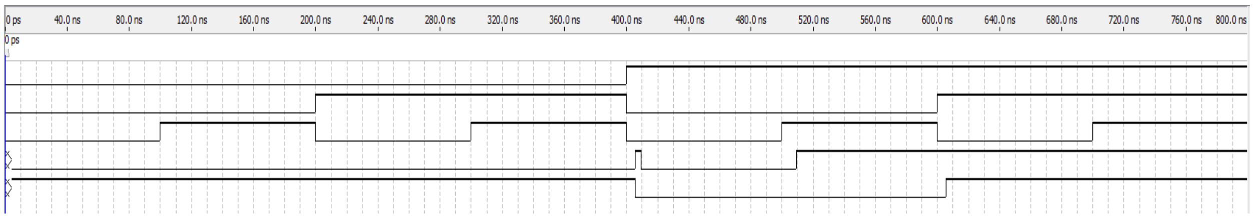
```
//A circuit design for equivalence2

module Equivalence2 (a, b, c, f1);

input a, b, c;

output f1;

assign f1 = a & (b | ~c) | ~a & ~b & c;

endmodule
```

Here is the truth table that encompasses the circuit above.

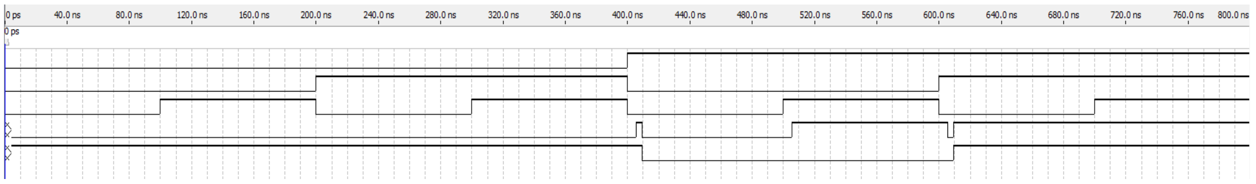
Input A	Input B	Input C	**Output F1**
0	0	0	**0**
0	0	1	**1**
0	1	0	**0**
0	1	1	**0**
1	0	0	**1**
1	0	1	**0**
1	1	0	**1**
1	1	1	**1**

Testing

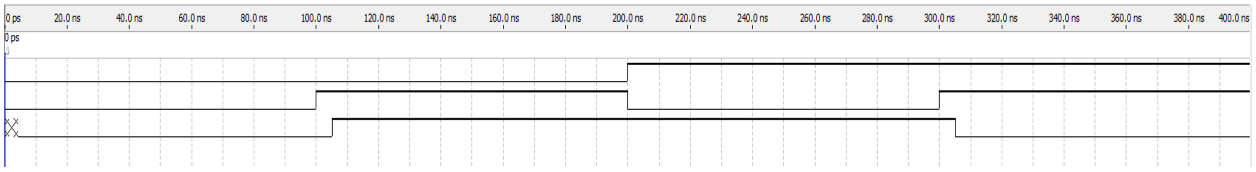
When testing the combinatorial circuit that I created using Verilog, I encountered no problems and it worked as expected for every single input combination.



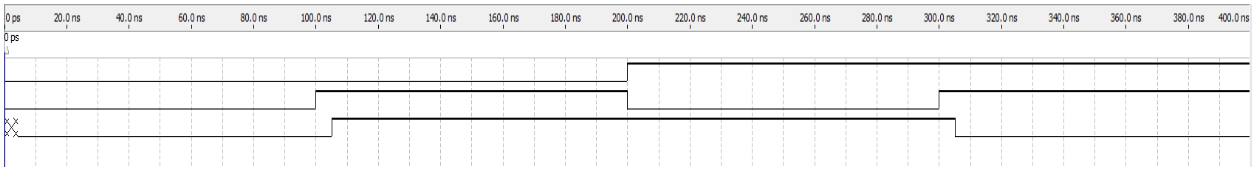
When testing the combinatorial circuit that I created using a Block-Diagram, I encountered no problems and it worked as expected for every single input combination.



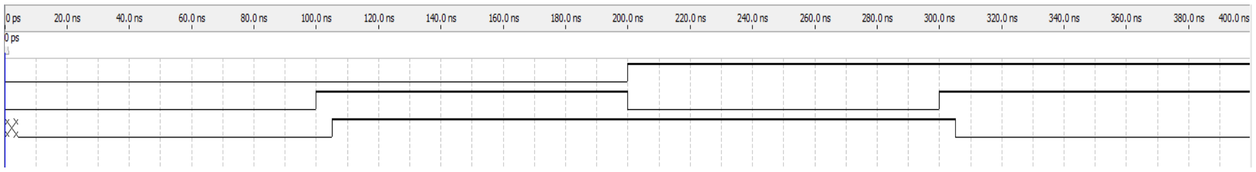
When testing the first XOR equivalent circuit that I created using a Block-Diagram, I encountered no problems and it worked as expected for every single input combination.



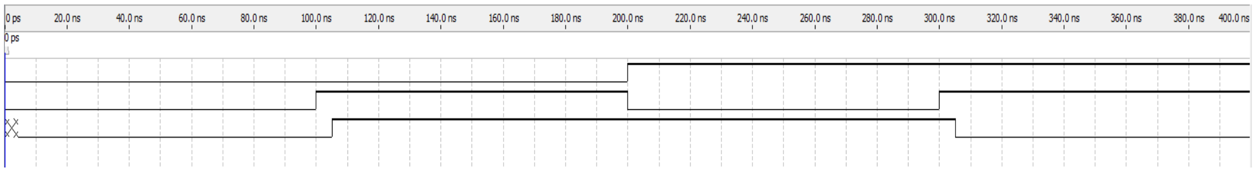
When testing the second XOR equivalent circuit that I created using a Block-Diagram, I encountered no problems and it worked as expected for every single input combination.



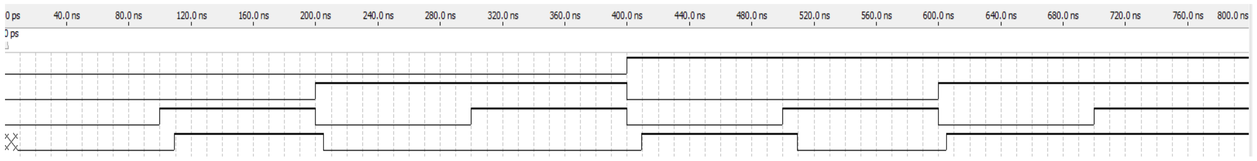
When testing the first XOR equivalent circuit that I created using Verilog, I encountered no problems and it worked as expected for every single input combination.



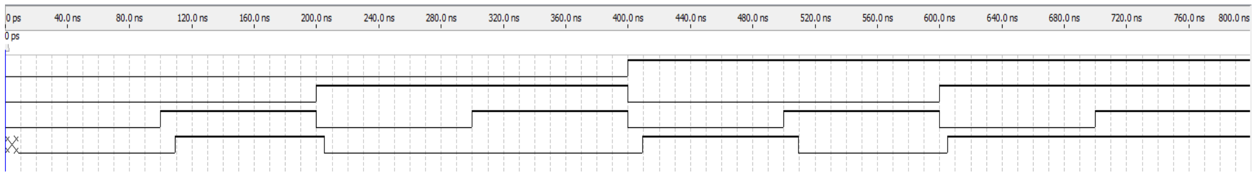
When testing the second XOR equivalent circuit that I created using Verilog, I encountered no problems and it worked as expected for every single input combination.



When testing the first circuit that I created in Verilog during the last part of the lab I encountered no problems and it worked as expected for every single input combination.



When testing the second circuit that I created in Verilog during the last part of the lab I encountered no problems and it worked as expected for every single input combination.



Conclusion

In this lab I better learned how to implement slightly more complex circuits into Quartus using both Verilog and Block-Diagrams. I believe I performed all the tasks set in front of me in a logical fashion and do not require improvement on my current method. In the last part of the lab I was asked to determine if two circuits were equivalent to each other. I determined that they were indeed, as there waveforms and truth tables matched. Overall, I believe this lab did a great job in letting me get more experience in using Quartus.