

ECE275: Sequential Logic Circuits

Lab 4: Ripple Adders and Verilog Modules

Pascal Francis-Mezger

October 5, 2020

Contents

0	Lab Overview	2
1	Creating a Full Adder	2
1.1	Use the Truth Table to Create Equations and a Verilog Module .	2
1.2	Test Your Full Adder	3
2	Utilize Your Full Adder to Create a 4 bit Ripple Adder	3
2.1	Create the 4 Bit Adder Module	3
2.2	Test Your 4 Bit Adder	4

0 Lab Overview

The goal of this lab will be to cement your ability to use Verilog modules and understand inputs, outputs, and the new topic of wires.

In this lab you will create Verilog code for a full adder, and then reuse it to create a ripple carry adder. This is a method of creating a simple two bit adder, and then tying the output into another adder to add numbers larger than 2 bits. There are less specific instructions for this lab than you may be used to at this point, because the goal is to make sure you understand how modules work. If you are stuck or unsure at any point, refer to last weeks lab.

1 Creating a Full Adder

1.1 Use the Truth Table to Create Equations and a Verilog Module

The truth table for the full adder can be seen in Figure 2. Use this truth table to create a Verilog module where bits A , B , and C_{in} can be passed in, and then S and C_{out} are returned as outputs. A and B represent the 2 input bits to be summed, C_{in} is the carry in from the previous adder, S is the summed output, and C_{out} is the carry out. You can see in Figure 3 how you will be utilizing the module you created to add larger numbers. Do not put the Verilog code for your full adder in your top level, make it a module. To do this refer to last weeks lab where in the second part you created the module BCD_Display.

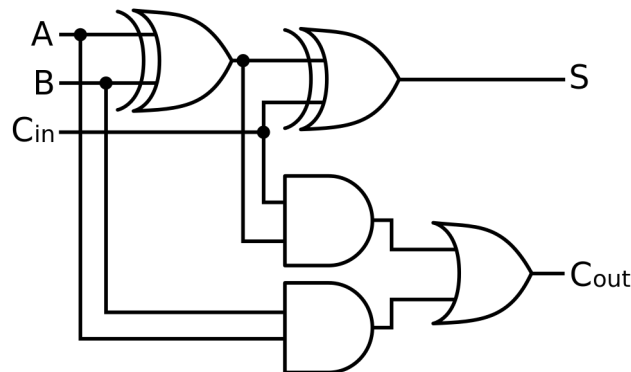


Figure 1: Digital Logic for a Full Adder <https://commons.wikimedia.org/wiki/File:Full-adder.svg>

Inputs			Outputs	
A	B	C _{in}	C _{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Figure 2: Truth Table for a Full Adder [https://en.wikipedia.org/wiki/Adder_\(electronics\)](https://en.wikipedia.org/wiki/Adder_(electronics))

1.2 Test Your Full Adder

Test your full adder by instantiating the full adder module from your top level. Use 3 switches (2 for input bits, and 1 for carry in) for inputs to the module, and 2 LEDs as outputs (sum and carry out). Make sure the switch positions and LEDs match your truth table before moving on.

2 Utilize Your Full Adder to Create a 4 bit Ripple Adder

2.1 Create the 4 Bit Adder Module

In this section you will use the full adder module you created to create a 4 bit ripple carry adder. Create another module where the inputs are A[3:0], B[3:0], C0. The outputs are S[3:0] and C4. You can see how these would be utilized for a four bit adder in Figure 3. The essential goal here is to add 2, 4 bit numbers (A[3:0] and B[3:0]) to create a 5 bit result (S[3:0] and C4). If you think about it, if you take the maximum value 4 bits can represent and sum it (1111+1111) this gives 11110, so the sum of any 2, 4 bit values can be represented by 5 bits. The remaining 0 would be a 1 with a carry in value.

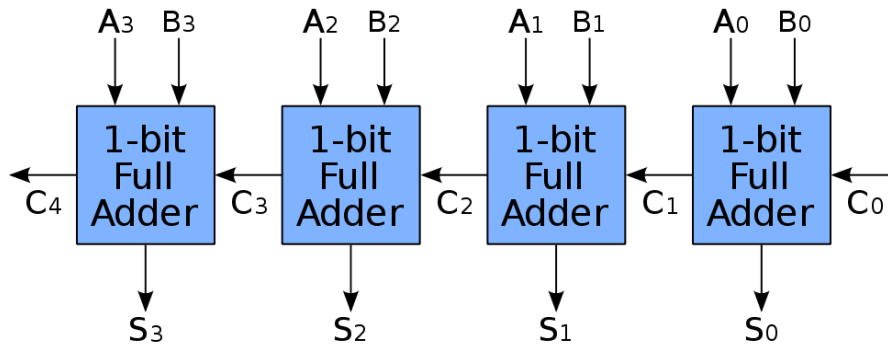


Figure 3: Using Full Adders for a Ripple Carry Adder https://upload.wikimedia.org/wikipedia/commons/thumb/5/5d/4-bit_ripple_carry_adder.svg/1000px-4-bit_ripple_carry_adder.svg.png

Inside the module, instantiate your full adder 4 times, using the form in Figure 3 for passing values from one adder to the next. Remember that you will need use unique values names for each module instantiation.

What you will notice in passing the values to the full adders, is that you are missing something. If you look at Figure 3, you will see that you have created $A[3:0]$, $B[3:0]$, $S[3:0]$, $C0$, and $C4$. What you are missing is $C1$, $C2$, and $C3$. Essentially you will need to create temporary variables to hold those values as outputs from the full adder instantiations to pass to the next instantiation. That is where you need to create a "wire" you would have seen it in the code last week, but this week you will need to create it yourself. A wire in Verilog works how a wire would work in tradition digital electronics. You can use it to tie from an input to an output. You can create a wire in your Verilog 4 bit adder module using "wire [2:0] carries" where $carries[0]$ would be $C1$, $carries[1]$ would be $C2$, and $carries[2]$ would be $C3$. You create this inside the module, not in the header.

2.2 Test Your 4 Bit Adder

In your top level, instantiate your 4 bit adder. You should use $SW[0]$ as the carry in, $SW[4:1]$ as A , $SW[8:5]$ as B , $LEDG[3:0]$ as S , and $LEDG[4]$ as carry out. Test it out and show the TA that the 5 LEDs ($LEDG[4:0]$) represent the sum of the 4 bit values A and B , as well as the carry in.