Lab 3: Adder/Subtractor

Introduction

As the digital circuit designs within Vivado become more complex, it is convenient to create separate modules and then combine them. Lab 2 introduced this concept, and this lab will serve to reinforce this practice. In this lab, one will write separate modules, then call them from another function This practice will allow users to create complex designs from simpler, less complex modules. To demonstrate this process a 4-bit full adder/subtractor will be designed and tested in this lab.

1 Prelab

1.1. Figure 1 shows the implementation of a Full Adder. Write out the truth table for a Full Adder.

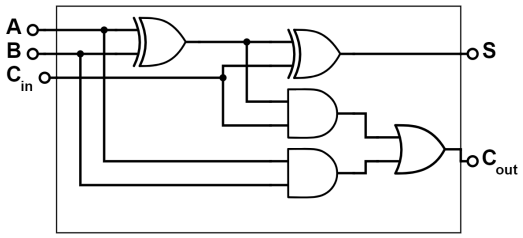

Figure 1: Full Adder Circuit

1.2. Write the truth table for a full subtractor.

1.3. Show how you can use half adders to build a full adder (Draw the block diagram, and box any half-adders used).

1.4. Figure 2 shows how to implement a 4-bit ripple adder using a sequence of 1-bit full adders. Using an example, verify that this circuit functions as a 4-bit adder.

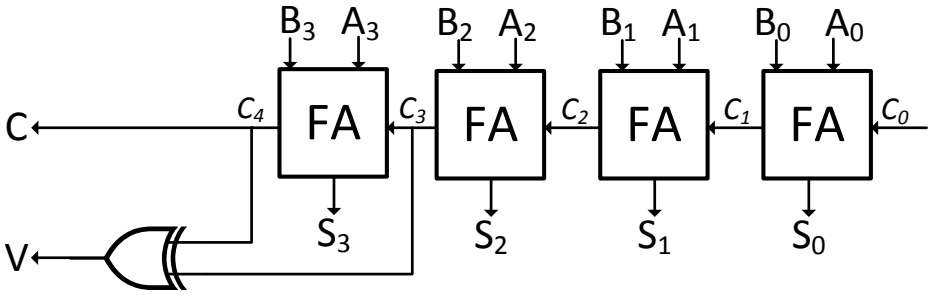

Figure 2: Ripple Carry Adder Circuit

1.5. **What does the output V, computed as $V = C_3 \oplus C_4$, represent?** ($\oplus$ means XOR) Consider when adding two positive numbers and when are adding two negative numbers.

1.6. By slightly modifying the circuit shown in Figure 2 we can design an adder/subtractor as shown in Figure 3. **Why does this circuit make an adder when the $Sel == 0$, and why does it behave as a subtractor when the $Sel == 1$?**

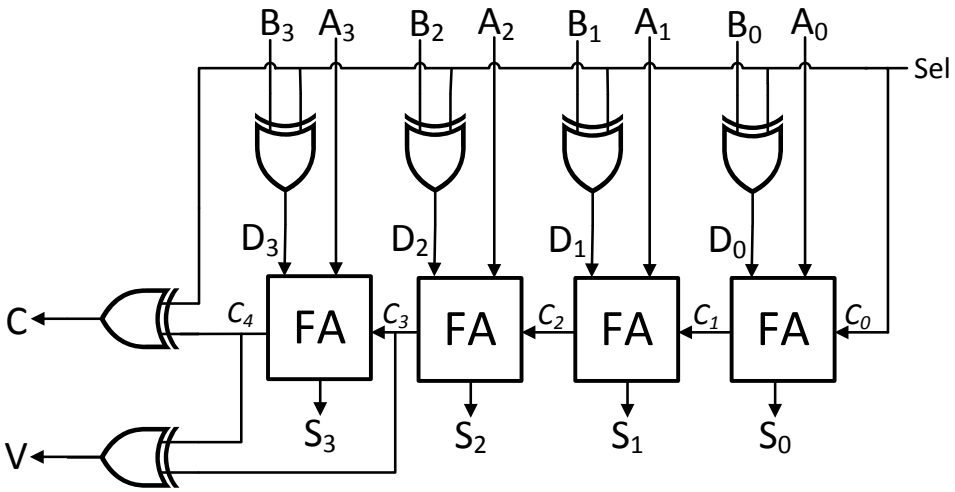

Figure 3: Ripple Carry Adder/Subtractor Circuit

1.7. Fill in Table 1.

Table 1: Outputs of an adder/subtractor

`sel`	**Input B**	**Output D in terms of B**
0	$B_3B_2B_1B_0$	
1	$B_3B_2B_1B_0$	

1.8. Using Table 1, write a Verilog program to implement a decoder that selects the proper input to the full adder depending on the `sel` signal. This should be a simple module, executing an XOR for each bit of `B` with `Sel`.

2 Lab

In the following sample programs (Listing 1, 2), the module **Inverter** is instantiated (called) five times as *Inv1*, *Inv2*, *Inv3*, *Inv4*, and *Inv5* in the higher level module, **FiveInverters**. This code implements a circuit similar to the one created by discrete logic chips in lab 1, but serves as an excellent example on calling modules in Verilog.
In Vivado, Each module will be placed in an individual .v file, under 'Sources'. To create new modules, one may click on the + icon above 'Sources' (To the left in the 'Flow Navigator'). Once a module has been called by another module, it will automatically be nested within the higher module. Vivado will then set the highest level module as the 'top' module to be executed.

Listing 1: An Example of a Module That Creates an Inverter

```
// Module stored in Inverter.v
module Inverter(output Inv_out, input Inv_in);
       not(Inv_out, Inv_in);   // create single inverter gate
endmodule
// end of module, end of .v file, with only one module per file.
```

Listing 2: An Example of a Module That calls 5 Inverters from the Previous Module

```
// Module stored in FiveInverters.v (This would be considered your 'main' module)
module FiveInverters(output FiveInvOut, input FiveInvIn);
    wire A, B, C, D;                // Create wires to connect between inverters.
    Inverter Inv1(A, FiveInvIn);    // Attach input to first Inverter,
    Inverter Inv2(B, A);            // Then second Inverter,
    Inverter Inv3(C, B);            // ...
    Inverter Inv4(D, C);            // ...
    Inverter Inv5(FiveInvOut, D);   // Last Inverter
endmodule
```

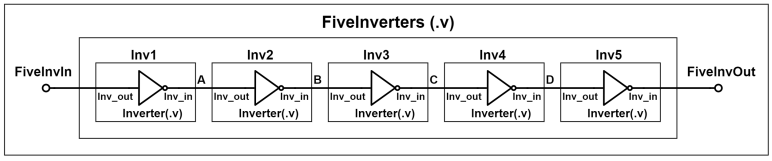

Figure 4: Block Diagram of the Module FiveInverters (.v)

2.1. Design a 4-bit adder using the half adders.

2.2. Add the required components to be able to select whether your design acts as a 4-bit adder or a 4-bit subtractor, based on an input `sel` signal.

2.3. Simulate your circuit and verify its operation. You will need to try enough combinations to verify that the results of the addition and subtraction, along with `C` and `V` are correct.

2.4. Add the BCD decoder you designed in Lab 2 to output the adder/subtractor so you can display the output on one of the 7-segment displays.

2.5. Connect your `sel` and your inputs (`A` and `B` to hardware switches (e.g. DIP Switches), and connect the output `S` to the 7-segment display. if you like you may also attach the outputs `C` and `V` to the on-board LEDs.