Lab 2 - Adder Subtractor

CECS 341 – Computer Architecture Organization

Student Antonio Ojeda SID
Student Bridget Naylor SID 025531413
Professor:Jose Aceves

California State University, Long Beach
College of Engineering
1250 Bellflower Blvd, Long Beach, CA 90840

Lab 2/7/2021

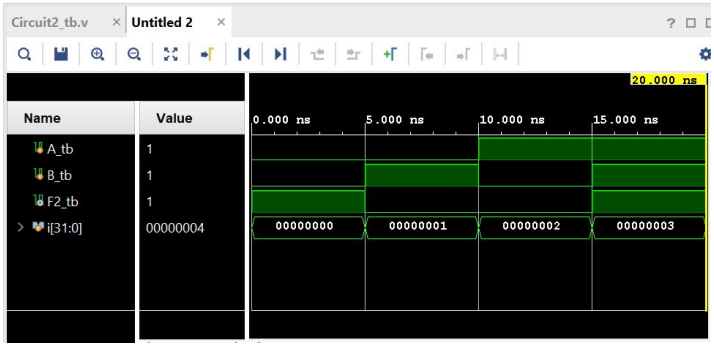
Lab 2 - Adder Subtractor CECS 341 Spring 2021

Goal/Objective:

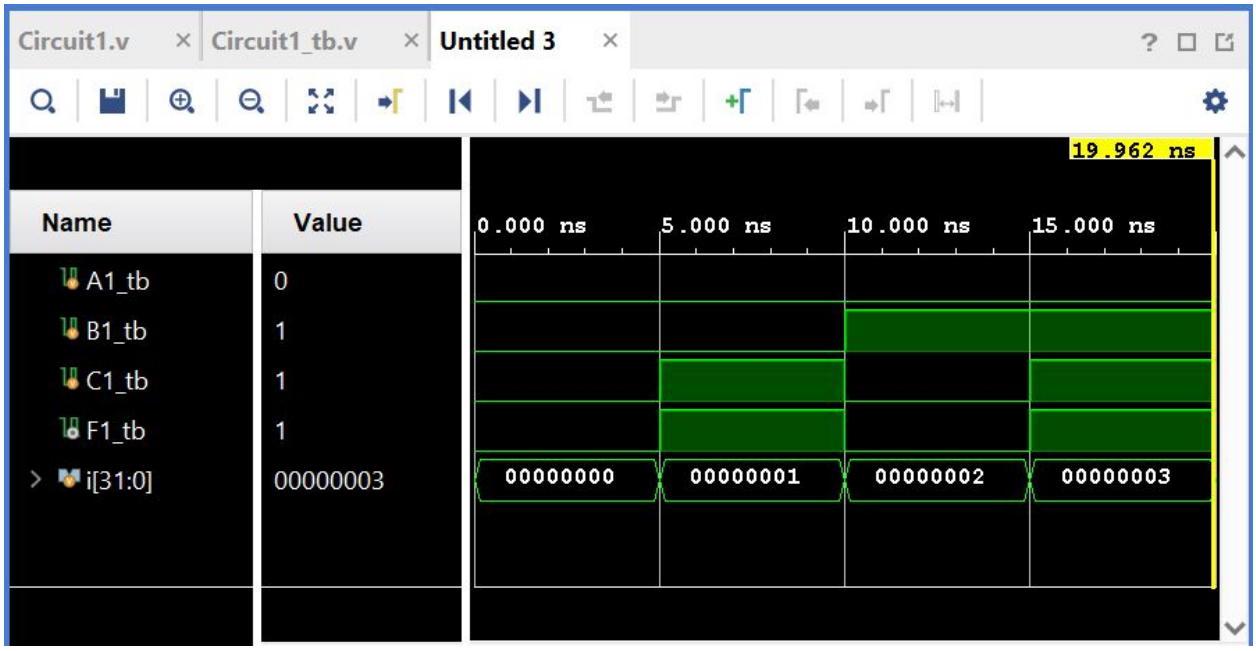
One of our goals for this lab was learning how to use bit wise operands like or(||), and(&&), as well as not(~) to represent schematics.
Our other goals for this lab was to design and display an adder subtractor.

Technical Description/Steps:(include screenshots and description here)

Part 1 one the lab was fairly simple because we were given the circuit and the equation for the circuit and we just needed to model and test it.

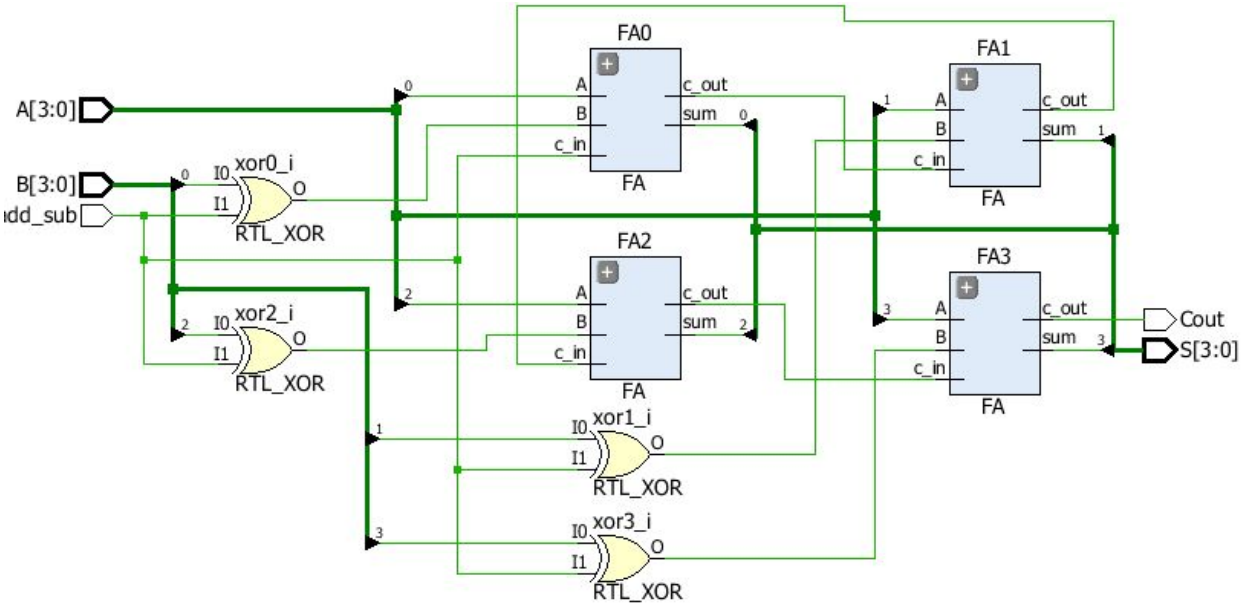


As you can see in this captured waveform from circuit 2 you can drag and drop the cursor to see which inputs lead to which output.
We can check these waveforms against the provided circuits with the wave form. For example in Circuit 1 for all inputs at 0 the output is 0 too.



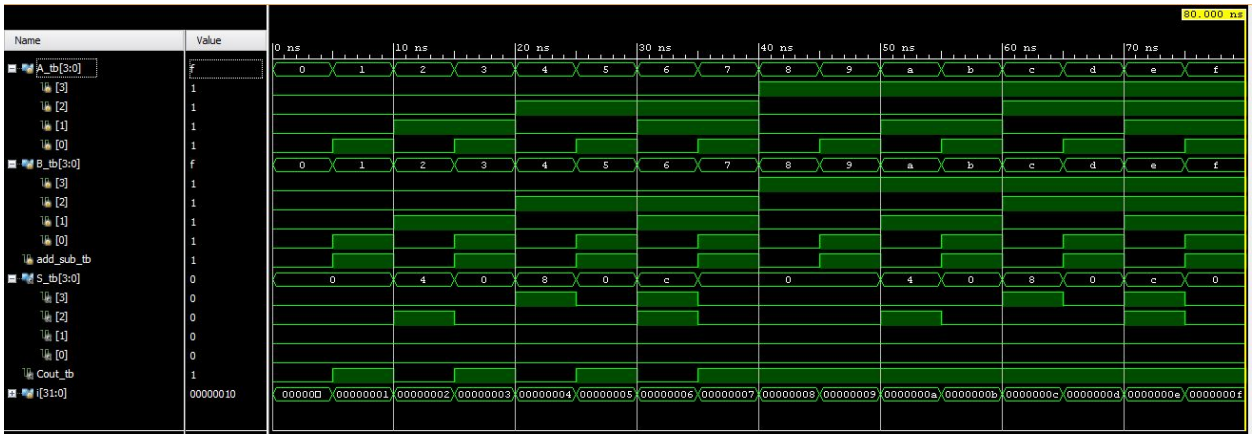
As input A and B change to 0 and input changes to 1 the output changes to 1.

In part 2 we needed to use the adder from last week and get the subtractor working.



This schematic shows the hardware design of a 4-bit Adder/Subtractor module. All A inputs go inside each full adder without going through any logic gates, while all B and add_sub inputs go through an

XNOR gate before going into a full adder. In addition, an extra add_sub input acts as a carry-in for the first full adder. The add_sub acts as a determinant for whether each full adder will conduct addition or subtraction by either a 0 or 1, respectively.
Each carry-out output acts as a carry-in for the next full adder, except for the last one.
The S bus is the output of the summation of both the A and B outputs as determined by the add_sub switch.



To create this waveform, we must register our inputs and wire our outputs. Then, we must put our variables under test via uut and declare our outputs as integers.
After that, we instantiate our module by displaying our for loop with the testbench variables (each inputting as the current iteration of the integer i).
Finally, we run our simulation and the displays appear as this waveform and the console.

Results: (what we would do differently next time)

For part 1 of this lab we were able to see how effectively circuits can be modelled in Vivado and how the waveform changes with the different inputs for every cycle of 5 ns that we had set.

For part 2 we used a 4-bit A input to represent the first binary number and 4-bit B input to represent the second binary number, along with add_sum to represent whether we were doing an addition or a subtraction on the numbers. For our outputs we had the carry digit and the S output for the sum of the addition or the subtraction of the two numbers.

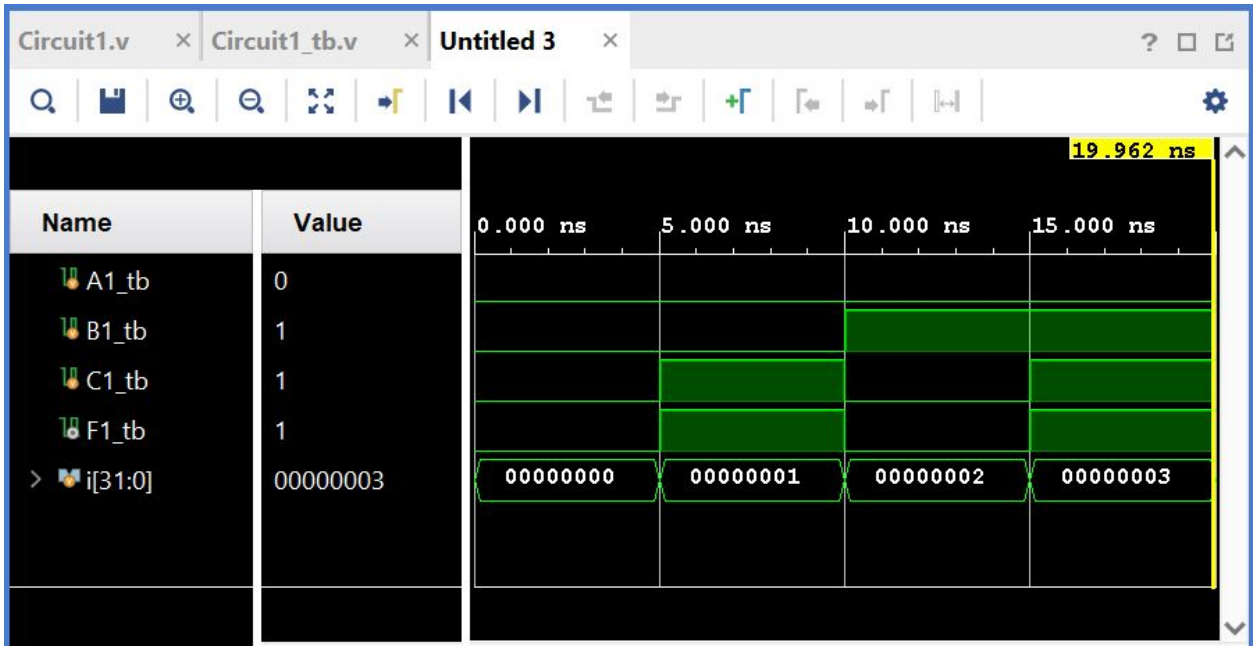
Conclusion:

One small difficulty I had in this lab was declaring multiple Full Adders under one FA declaration. I was putting ";" after each Full Adder name, but for the syntax of multiple declarations you need to put a "," after each declaration.

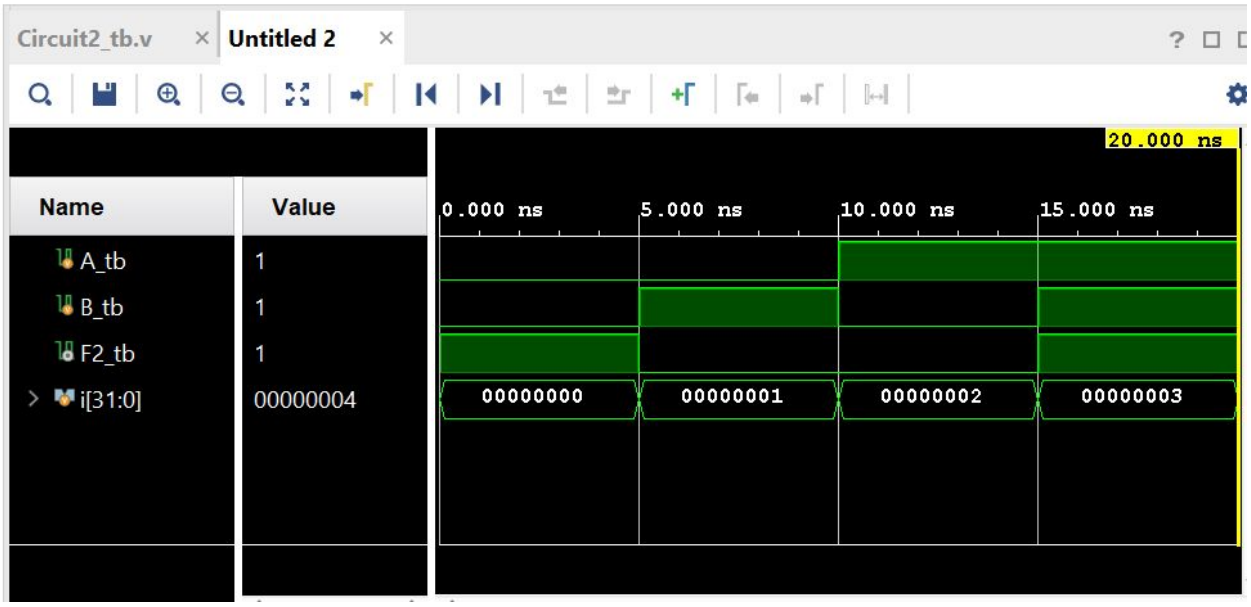
Another difficulty we had was instantiating our adder/subtractor module, but we were able to solve that problem by looking back on our previous labs and using that information to build upon our implementation.

RTL Schematic, Captured WaveForm:

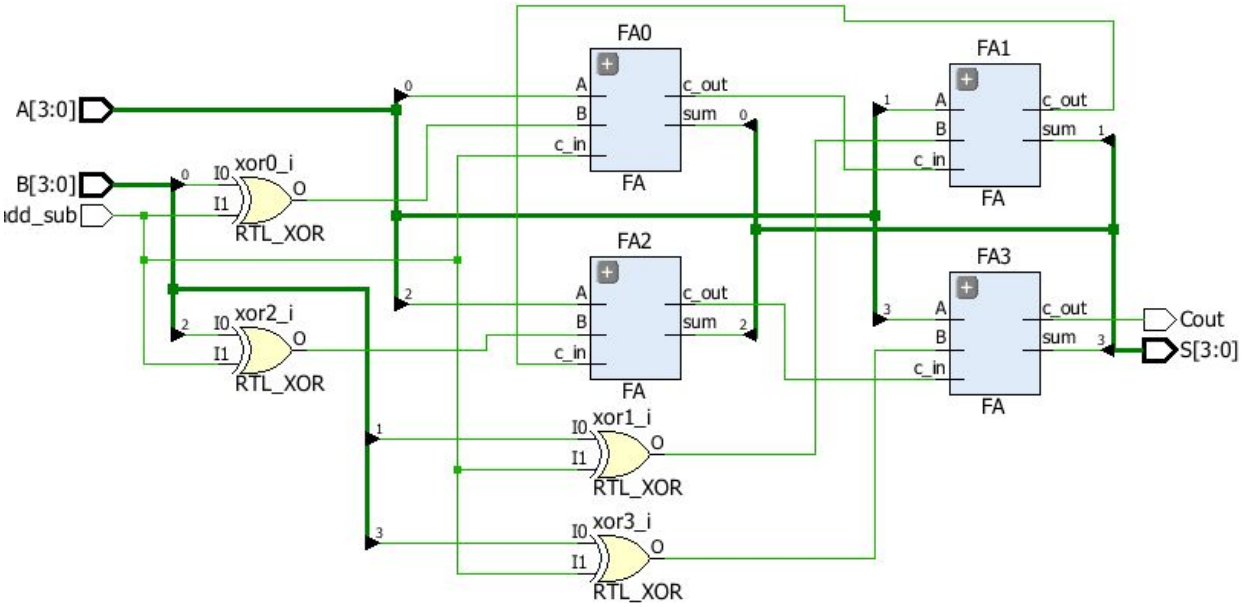
Part 1: Circuit 1

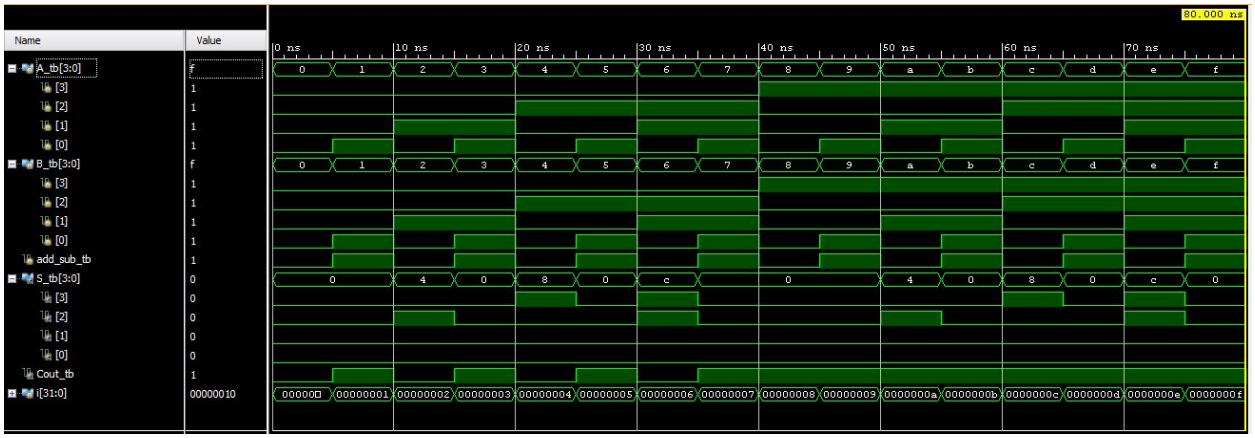


Part 1: Circuit 2



Part 2: Adder Subtractor


A[3:0]
B[3:0]
dd_sub
xor0_i
xor1_i
xor2_i
xor3_i
I0
I1
O
RTL_XOR
FA0
FA1
FA2
FA3
A
B
c_in
c_out
sum
FA
Cout
S[3:0]


Name
Value
A_tb[3:0]
B_tb[3:0]
add_sub_tb
S_tb[3:0]
Cout_tb
i[31:0]
00000010
80.000 ns
0 ns
10 ns
20 ns
30 ns
40 ns
50 ns
60 ns
70 ns