CSC 327 – Lab 3

Shyang Kao

ID: 5206454

Course Name: CSC 326

Due Date: February 26, 2022

Experiment Description

Part I -- Hierarchical Ripple Schematic Capture Design Entry and Implementation using Quartus II and performing Functional and Timing Simulations

Hierarchical Design Methods:

In this part of the laboratory each student is to utilize schematic capture techniques to create the Adder/Subtractor module using hierarchical schematic capture techniques. As the name implies, hierarchical designs utilize a hierarchy of logical levels to represent the design. In this type of design one does not implement the design in a flat manner where the entire design is placed on a single design sheet that utilizes the lowest-level gate/flip-flop components (i.e. primitives) but rather, the design is implemented on multiple sheets that are used to form a hierarchy of components. In this manner complex components are created using a set of less complex components, which themselves may be made up of more basic components. This continues until the components that are the lowest level to be modeled are encountered. These primitive components are usually at the gate, flip-flop or latch level.

Design Capture Assignment:

Students are to develop of multi-level hierarchical design of an Adder/Subtractor using schematic capture techniques. The top level of the design should be as shown in Figure 1. It is to be made up of two four-bit adders that are cascaded together by connecting the *B_Cout* output of one Four_BIT_Adder_Sub_module to the *B_Cin* input of the other module, as shown in Figure 2.

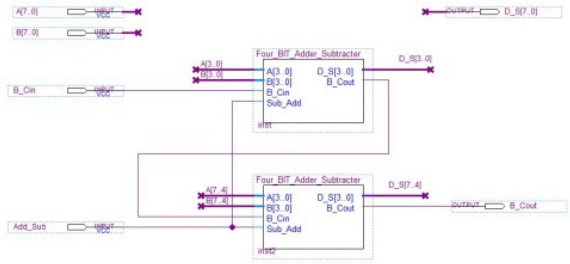


Figure 2 Eight_BIT_Adder_Sub Module created form two Four_BIT_Add_Sub_module

Each Four_BIT_Adder_Sub_module should be made up of four one-bit full subtractor/adder module as shown in Figure 3.

Eight-bit Adder/Subtractor Module

The example design that is to be created and evaluated in this laboratory experiment is an eight-bit adder/subtractor module as shown in Figure 1.

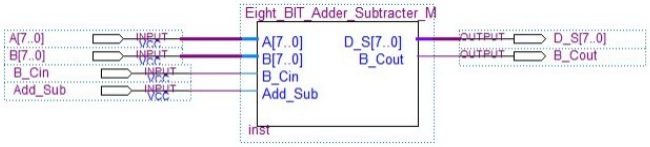


Figure 1: (a) Eight-Bit Subtractor/Adder Module

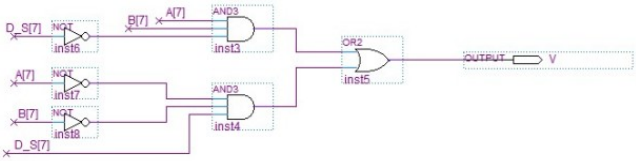


Figure 1: (b) Overflow Signal (V)

Its function is to output the valid difference or sum between two eight-bit quantities that are present on its *A* and *B* inputs. Figure 1(a) can be cascaded to support the creation of proportionately larger size subtraction and addition operations so there is also a borrow/carry input, *B_Cin*, and a borrow/carry output, *B_Cout*. Whenever the *Add_Sub* line is a logic high the *D_S* output produces a difference of *A-B* and the *B_Cout* indicates if there is a borrow is required at the most significant bit position. Whenever the *Add_Sub* line is at a logic low, the *D_S* output produces the sum of *A+B* and the *B_Cout* line indicates if a carry is generated at the most significant bit position. Figure 1 (b) shows the Overflow signal created from the MSB bits of A, B and D_S to indicate if there is any overflow in the addition/subtraction operation. The module itself can be implemented in a number of ways, such as a carry-lookahead subtractor/adder, with each style of implementation presenting various complexity performance trade-offs.

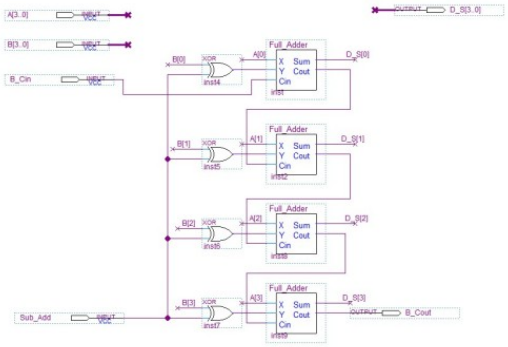


Figure 3 Four-BIT_Adder_Sub_module created from four one-bit full adder and Ex_OR gates

Each Adder_Sub_Module should then be represented as shown in Figure 4.

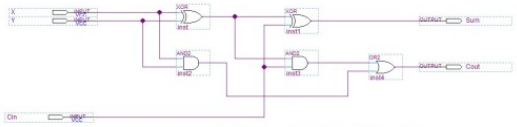


Figure 4 Representation of one-bit full adder (i.e. Adder_Sub_Module of our design)

Components are to be created from the bottom level up (i.e. you should start with *Adder_Sub_Module* and then build all the way up to *Eight_BIT_Adder_Subtracter_Module*). In other words, first create the *Adder_Sub_Module* (as shown in Fig. 4) and save it as *Adder_Sub_Module.bdf* in your current project directory and then create the symbol file of the module (in Quartus follow the sequence (Figs. 5 and 6), ***File → Create/Update → Create Symbol Files for Current File***) then sequentially create *Four_BIT_Adder_Sub_module.bdf* and all the way up to top level entity, *Eight_BIT_Adder_Subtracter_Module.bdf* module.

Implementation Assignment:

After the successful demonstration of the simulation with the assigned input stimulus the design is to be downloaded and implemented using the DE2-115 rapid prototyping platform. Before doing this student should modify the top level of the design so that all of the input signals, which are to be connected to (SW0-SW17), are also run to the discrete LEDs (LEDR0-LEDR17) on the DE2-

115 as shown in Figure 10. The pin assignments for all of the pins need to be set to the Cyclone IV E pin numbers as shown in Figure 11 (which shows the pin assignment of the output signals of the design (i.e. B_COUT to LEDG8 (PIN_F17) and D_S[7..0] to LDEG7..LEDG0 (PIN_G21..PIN_E21), respectively)). You are to complete the rest of the pin assignments (input signals) of your design. Assign A[7]…A[0] to SW15…SW8, respectively, B[7]…B[0] to SW7…SW0, respectively, the output of **Overflow** to LEDR16, and *Add_Sub* to SW17. In order to see the status of your inputs (whether 0 or 1), LEDR17-LEDR0 are also to be connected to the respective pins of the board. Use the DE2-115 user manual for the specified pin assignment.

Simulation Assignment:

To perform the simulation, the circuit should be tested with some known inputs to check whether it is producing the desired results. The circuits designed in Figure 1 can be used to perform subtraction operation using either 2's complement or 1's complement technique. Keep it in mind that no matter what approach you use the circuit should produce the same result. Depending on which technique you want to use, the Circuit in Figure 1 needs a minor modification.

If you decide to use 2's complement technique, the Add-Sub control line should be connected to the B_Cin input line, and B[7] and Add_Sub signals are Ex-OR' inputs and the output of the Ex-OR is connected to the B7 input of the overflow module, as shown in Figure 7. (**Why is so?? Explain**)

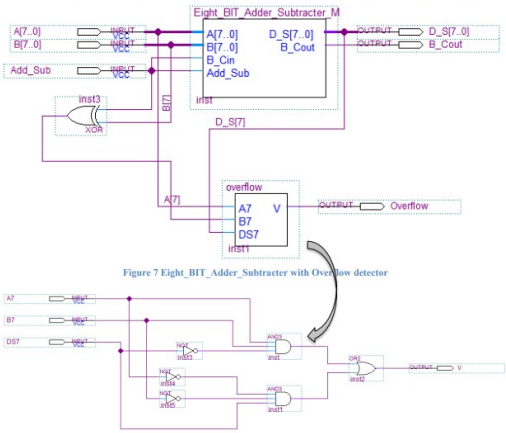


Figure 7 Eight_BIT_Adder_Subtracter with Overflow detector

Now simulate the circuit with the following input data sets. Simulation result is shown in Figure 9. To verify results, the corresponding Decimal Values are shown as signed decimal number on the simulation.

First Pair:	**Second Pair:**	**Third Pair:**	**Fourth Pair:**
$A = 64_{10} = 01000000_2$;	$A = 64_{10} = 01000000_2$;	$A = 64_{10} = 01000000_2$;	$A = 32_{10} = 00100000_2$
$B = 32_{10} = 00100000_2$;	$B = 16_{10} = 00010000_2$;	$B = 64_{10} = 01000000_2$;	$B = 64_{10} = 01000000_2$

Part II:

Simulation Assignment: Modify the circuit in Figure 7 such a way that it would perform subtraction using 1's-Complement technique. After you design the circuit simulate the circuit with the input pairs given below, and demonstrate that your circuit is producing the expected results.

First Pair:	**Second Pair:**	**Third Pair:**	**Fourth Pair:**
$A = 64_{10} = 01000000_2$;	$A = 64_{10} = 01000000_2$;	$A = 64_{10} = 01000000_2$;	$A = 32_{10} = 00100000_2$
$B = 32_{10} = 00100000_2$;	$B = 16_{10} = 00010000_2$;	$B = 64_{10} = 01000000_2$;	$B = 64_{10} = 01000000_2$

To see the propagation delay of the output D_S signal, use **timing simulation** instead of functional simulation. To see the propagation delays more clearly, we changed the simulation end time and grid size from µs to ns. After you have reproduced the simulation of Figure 9, you will find that D_S is not produced right at the edges of the transition. It takes some time before you get the correct result. Which demonstrate the effect of propagation delays (Figure 12)?

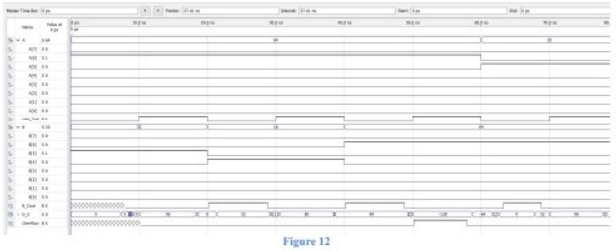


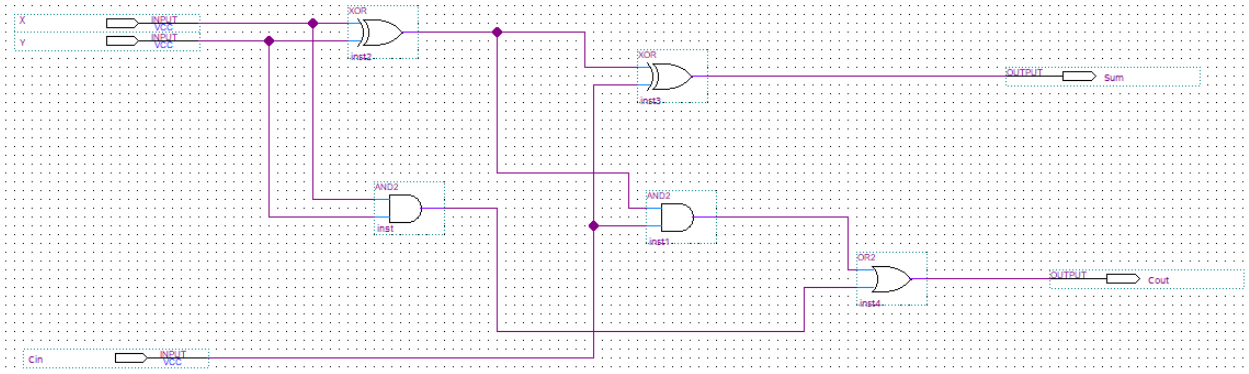
Figure 12

Implementation Assignment:

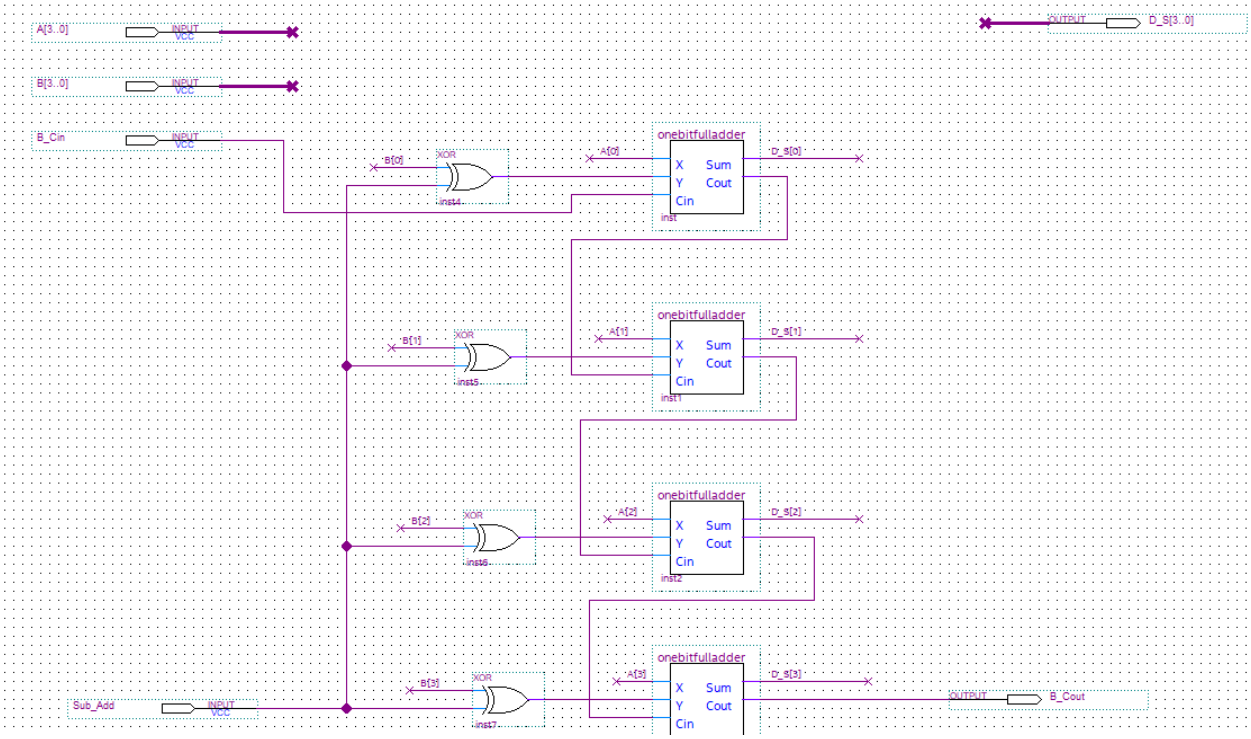
After the successful demonstration of the simulation with the assigned input stimulus the design is to be downloaded and implemented using the DE2-115 rapid prototyping platform. Before doing this student should modify the top level of the design so that all of the input signals, which are to be connected to (SW0-SW17), are also run to the discrete LEDs (LEDR0-LEDR17) on the DE2-115 as shown in Part-I. The pin assignments for all of the pins need to be set to the Cyclone IV E pin numbers as shown in Figure 11 (which shows the pin assignment of the output signals of the design (i.e. B_COUT to LEDG8 (PIN_F17) and D_S[7..0] to LDEG7..LEDG0 (PIN_G21..PIN_E21), respectively)). You are to complete the rest of the pin assignments (input signals) of your design. Assign A[7]…A[0] to SW15…SW8, respectively, B[7]…B[0] to SW7…SW0, respectively, the output of **Overflow** to LEDR[16], and *Add_Sub* to SW17. In order to see the status of your inputs (whether 0 or 1), LEDR17-LEDR0 are also to be connected to the respective pins of the board. Use the DE2-115 user manual for the specified pin assignment.

Experimental Results

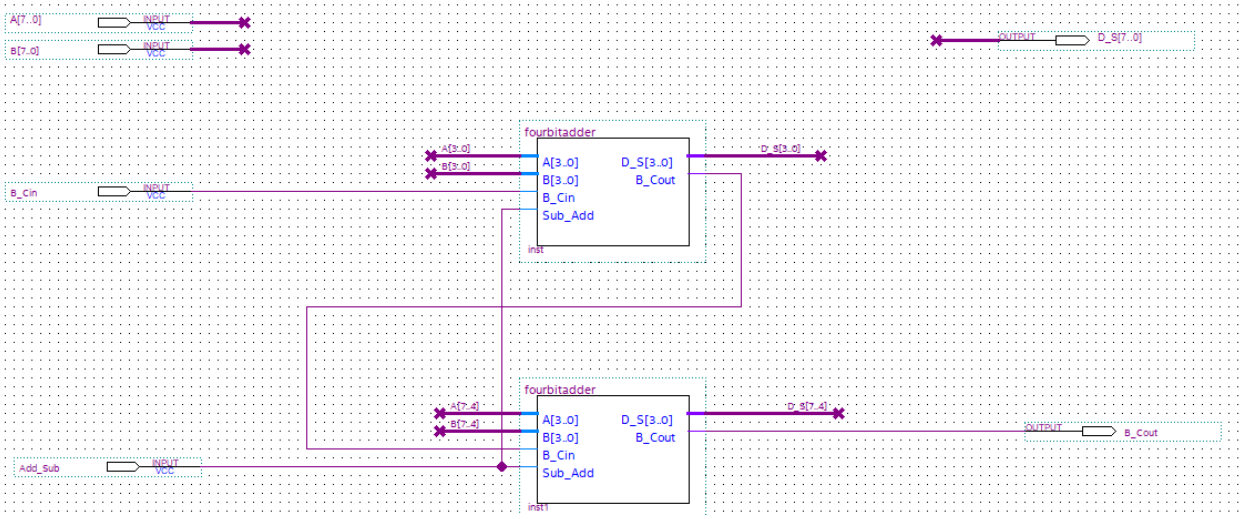
One-bit full adder circuit design



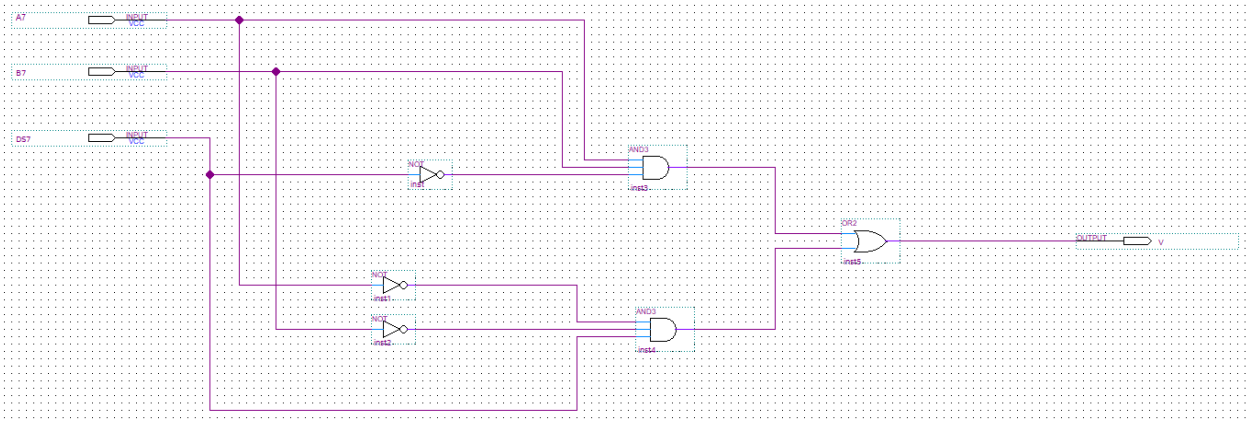
Four-bit adder-subtracter circuit design



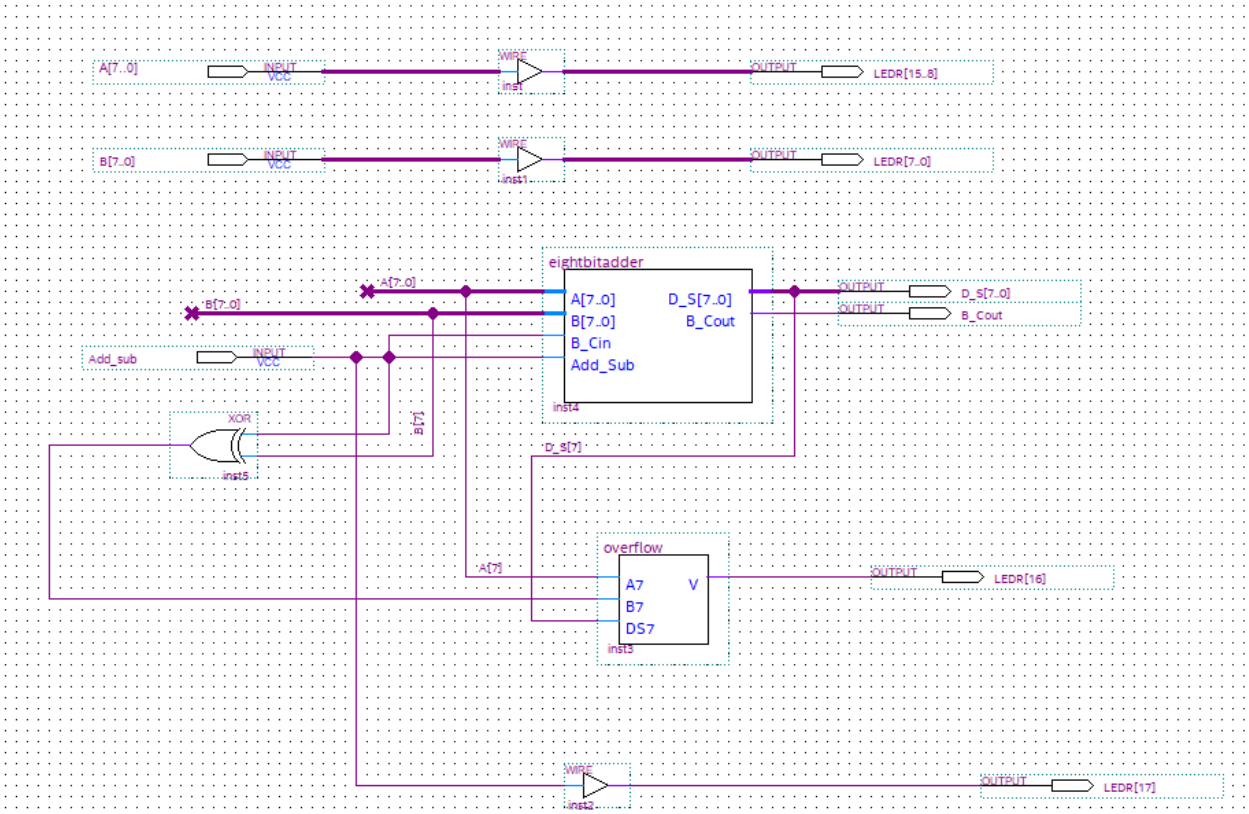
Eight-bit adder-subtracter circuit design



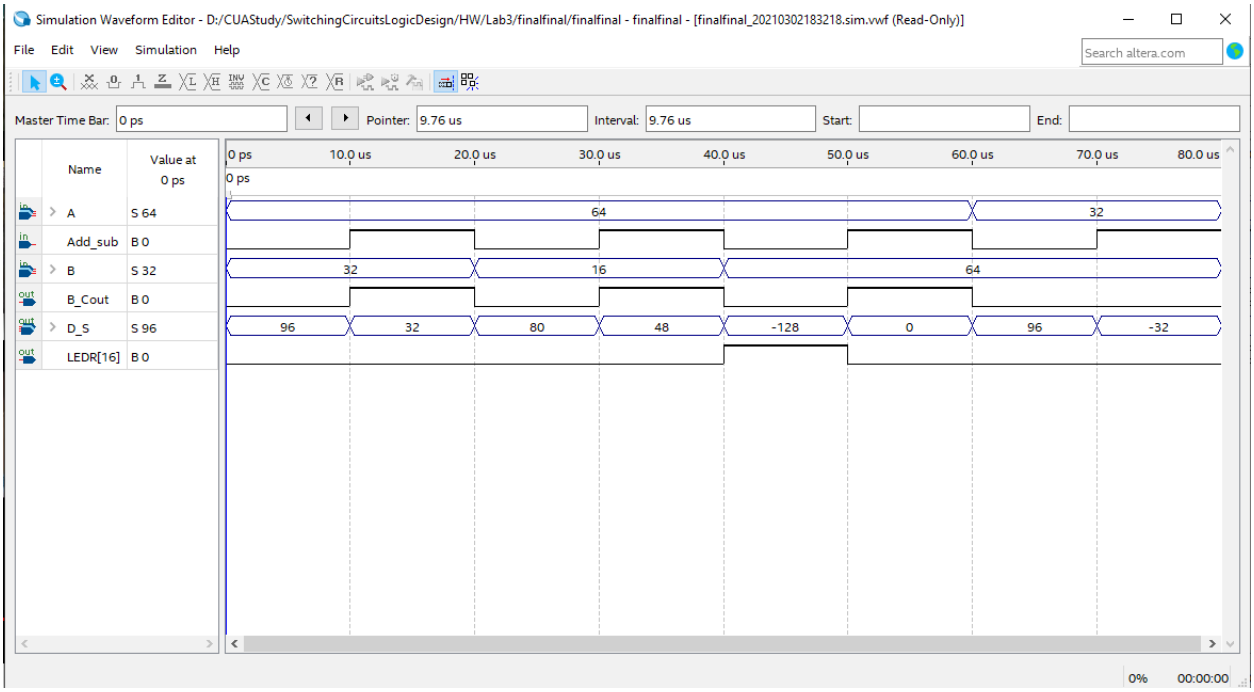
Overflow circuit design



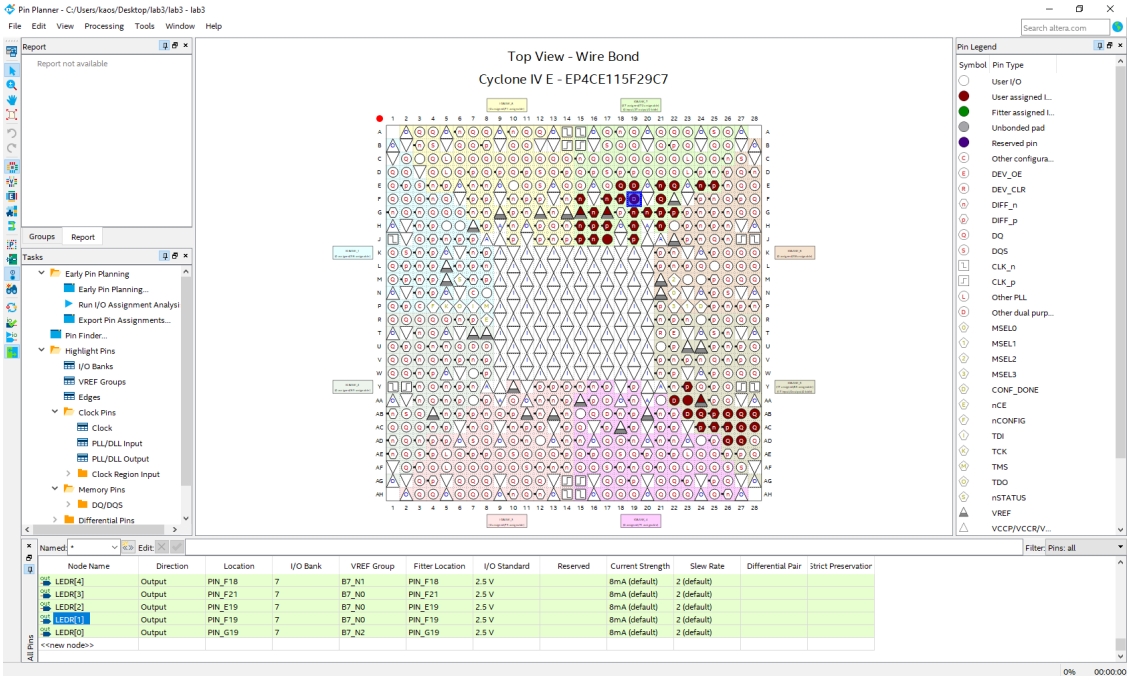
2’s complement adder-subtracter circuit design



2’s Complement Simulation result



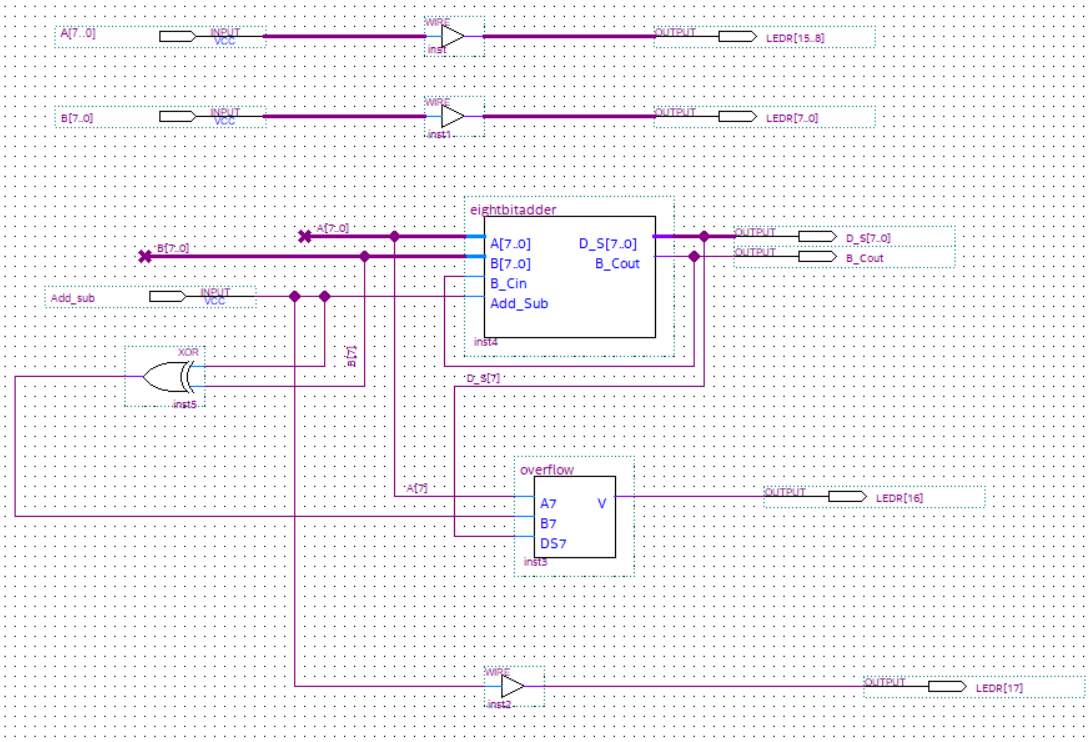
2’s Complement Pin Planner



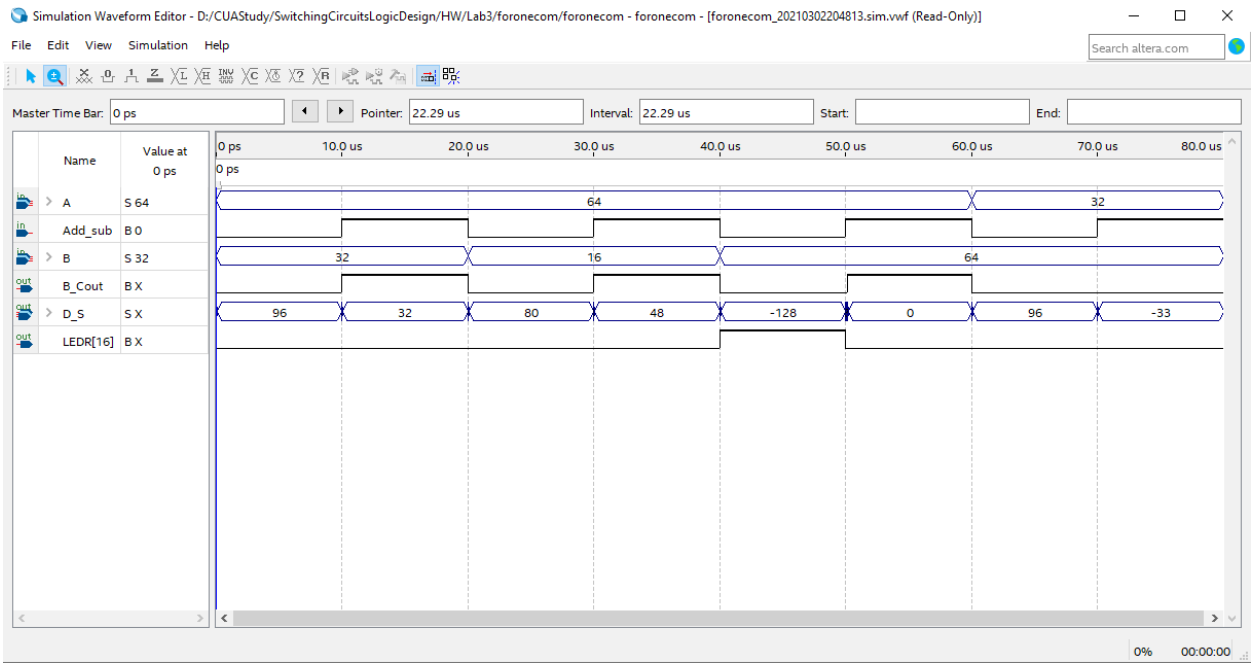
Question on page 5

Answer: The circuit's sign and arithmetic operator are determined by the Add sub that connects to the B Cin. B[7] and Add sub proceed to the Ex-OR gate to determine the final sign, and then to B7 to contribute the overflow detection factor. The DS7 receives the output and checks for overflow.

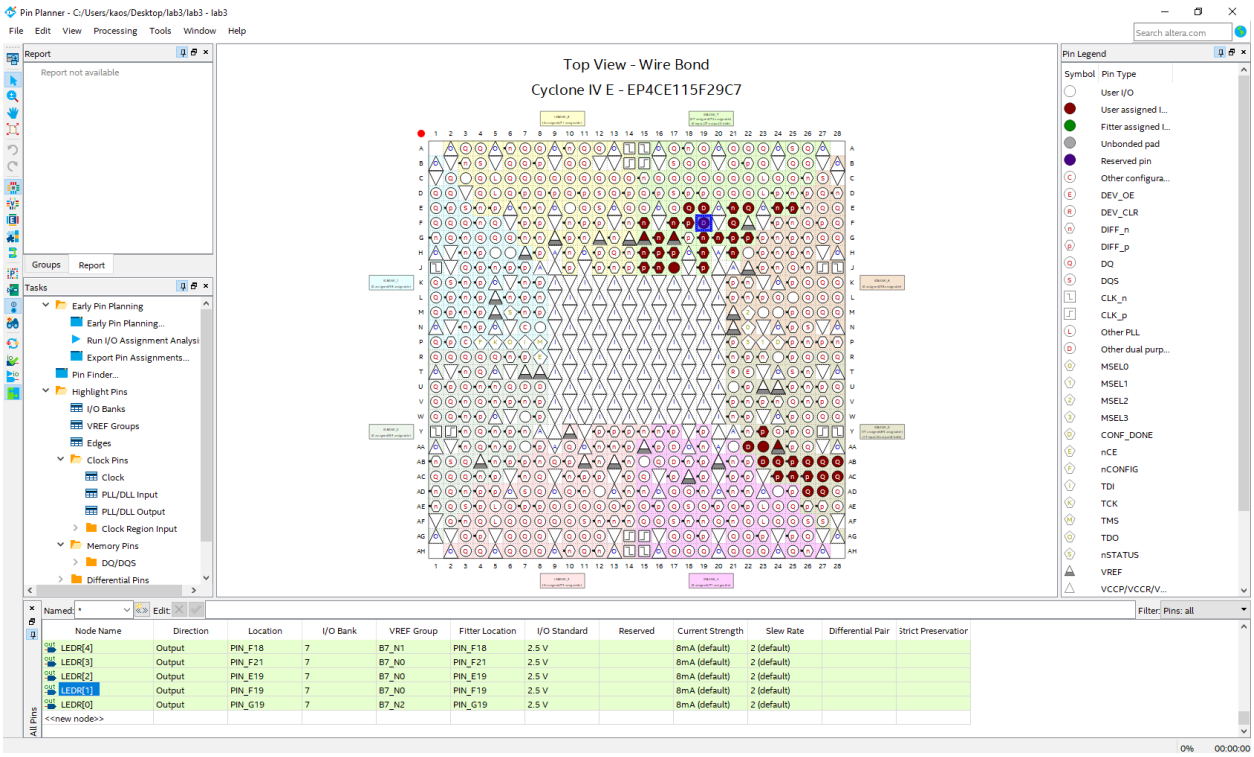
1’s complement circuit design



Simulation result



1’s Complement Pin Planner



Question on page 8:

Ans: A 1's complement is sometimes provided an extra 1 that isn't added to the outcome. This procedure will take some time to transmit and add that 1 to the results.