Name: ________________________________	ID# ____________________
Date Submitted: __________________	Time Submitted __________________

CSE 4357/5357 Advanced Digital Logic Design Spring Semester 2021

Assignment #1 – Ripple-Carry Adder/Subtractor

Due Date – January 30, 2021 (11:59 PM)

Submit on Canvas Assignments

CSE 4357/5357 ASSIGNMENT #1 SPRING 2021

RIPPLE-CARRY ADDER/SUBTRACTOR

(100 POINTS)

PURPOSE/OUTCOMES

To design in Verilog, implement on the DE1-SoC, and test a basic ripple-carry adder/subtractor circuit. You will also review how to represent negative numbers using 2's complement, how to detect overflow, and how to display decimal numbers on seven-segment displays. Once you complete this assignment, you will have demonstrated an ability to design and model combinational logic circuits in Verilog that meet specified requirements and to implement the design using a Field Programmable Gate Array (FPGA).

BACKGROUND

Four full adders can be interconnected as shown in Figure 1 to form a ripple-carry adder for adding two four-bit binary numbers. If *A* and *B* are signed binary numbers, then *A*3, *B*3, and *S*3 are the sign bits. An *n*-bit ripple-carry adder can be built using *n* full adders.

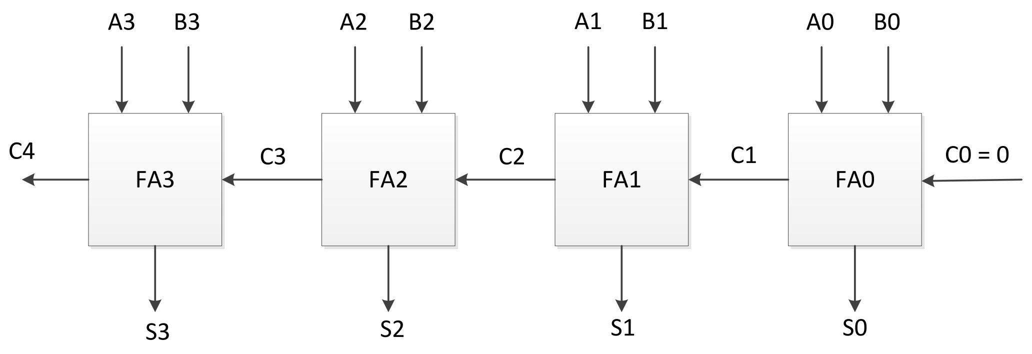


Figure 1. Four-Bit Ripple-Carry Adder (A + B)

Recall that if *A* and *B* are binary numbers then $A – B = A + (-B) = A + [B]_2 = A + [B]_1 + 1$ where $[B]_2$ is the *2's* complement of *B and* $[B]_1$ is the 1's complement. Example, if $A = 0101$ and $B = 0010$, then $A – B = 0101 + (-0010) = 0101 + [0010]_2 = 0101 + [0010]_1 + 1 = 0101 + 1101 + 1 = \cancel{1}0011.$
The following modified four-bit ripple-carry adder performs all of this in hardware.

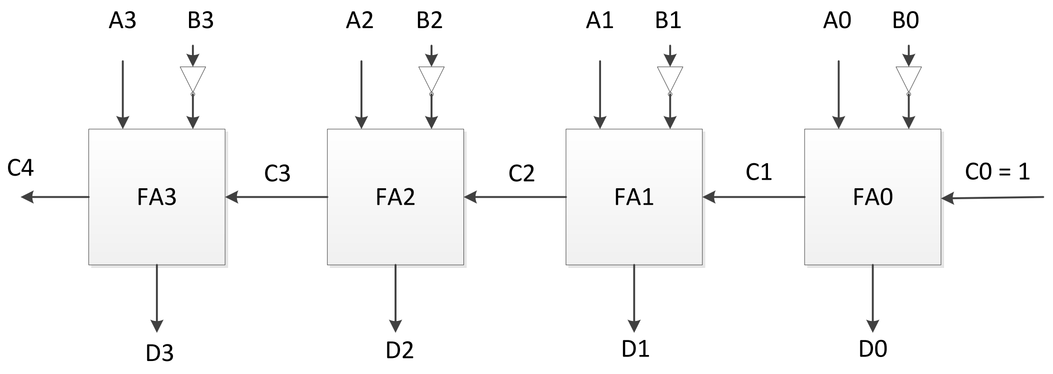


Figure 2. Four-Bit Two's Complement Subtractor (A - B)

Replacing the NOT gates in Figure 2 with XORs as shown in Figure 3 results in a circuit that adds or subtracts depending upon value of *C0* (*C0*=0 to add, *C0*=1 to subtract).

If C0 = 0, then $(C_4 R_3 R_2 R_1 R_0)_2 = (A_3 A_2 A_1 A_0)_2 + (B_3 B_2 B_1 B_0)_2$. So, R = A + B.

If C0 = 1, then $(C_4 R_3 R_2 R_1 R_0)_2 = (A_3 A_2 A_1 A_0)_2 + [B_3 B_2 B_1 B_0]_{1ns} + 1$. So, R = A – B.

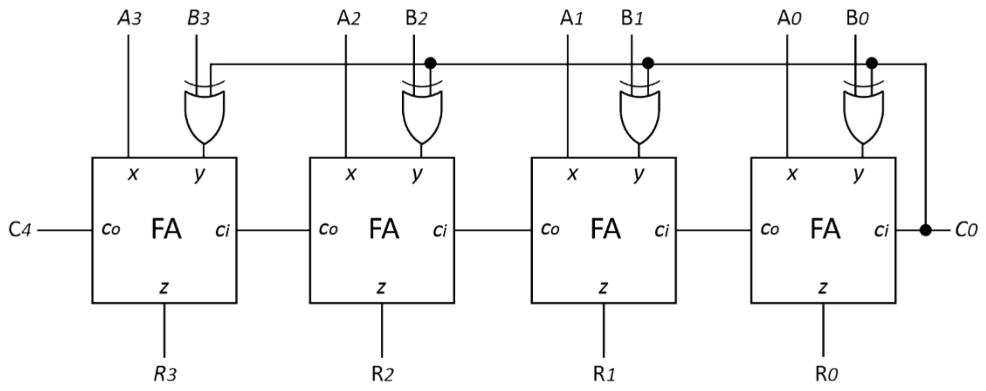


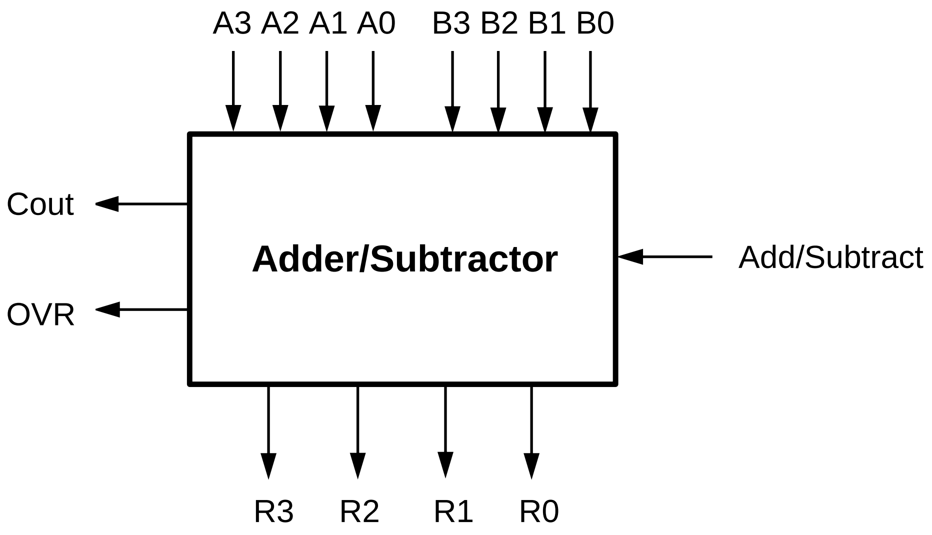
Figure 3. Four-Bit Ripple-Carry Adder/Subtractor

Overflows (OVR) can be detected using an XOR gate since

$$OVR = C4 \oplus C3$$

DESIGN REQUIREMENTS

Your assignment is to design, implement, and test a four-bit ripple-carry adder/subtractor with carry out and overflow outputs as shown below. Assume A, B, and R are four-bit signed binary numbers and Cout and OVR are carry out and overflow functions. Assume a two's complement number system is used for signed number representation.



If Add/Subtract = 0, then R = A + B; else R = A – B.

Figure 4. Ripple-Carry Adder/Subtractor

DESIGN PROCESS (20 points)

1. Design your Adder/Subtractor using full-adders and XOR gates as components. Draw a schematic diagram.
2. Write a Verilog model of your design.
3. Compile your Verilog code.
4. Correct any syntax errors.

DESIGN VERIFICATION (20 points)

1. Simulate your design to verify its correctness. Use the following values of A and B in your simulation.

   (a) 0101 +/- 0001 *
   (b) 0111 +/- 0001 *
   (c) 0111 +/- 1111
   (d) 0110 +/- 1101
   (e) 1010 +/- 0011 *
   (f) 1001 +/- 1110
   (g) 1010 +/- 1110
   (h) 1101 +/- 1100 *

2. Record the simulation results in a table for your report.

DESIGN ENHANCEMENT (20 POINTS)

1. Write a Verilog 2's-complement to decimal decoder that displays signed decimal numbers on two seven-segment displays. Positive numbers should be dislayed as a blank + the magnitude, and positive numbers should be displayed as a minus sign + the magnitude.
2. Instantiate the decoder in to your Adder/Subtractor Verilog code to display *A*, *B*, and *R* on the DE1-SoC (HEX5, HEX4, HEX3, HEX2, HEX1, HEX0)
3. Compile your code.
4. Record your Verilog lisitng and compilation summary for your report.

DE1-SoC iMPLEMENTATION (20 points)

1. Implement your design on the DE1-SoC using the following pin assignments. The DE1-SoC pin assignments can be found in the attached table.

   Input *A*: SW9, SW8, SW7, SW6 | Input *B*: SW5, SW4, SW3, SW2
   Input *Add/Subtract*: SW0
   Output *A*: HEX5, HEX4 | Output *B*: HEX3, HEX2
   Output *R*: HEX1, HEX0
   Output *Cout*: LEDR1 | Output *OVR*: LEDR0

2. Program the DE1-SoC with your design.

DE1-SoC TESTING (20 points)

1. Test your implementation by applying the same patterns as above.
2. Record the test results in a table for your report.
3. Take a picture of your results for cases marked with *

LAB REPORT REQUIREMENTS

1. Cover sheet
2. Design requirements
3. Schematic diagram
4. Verilog code
5. Simulation Results
6. Pin Assignments
7. Test results
8. Photos of marked test results