CDA 3201L Thursday Section
Lab number 04 -- Combinational Logic Circuits IV
06/14/14 , John Gangemi & Kyle Peck

PURPOSE AND OBJECTIVES

Design a circuit that implements four arithmetic functions using two 1-bit control signals. The inputs for the functions are two 4-bit binary numbers in two's complement form and a 4-bit binary number in two's complement form as the output. One TTL 74LS83 IC and any number of other components available in the lab can be used to create the functionality of the circuit.

COMPONENTS USED

- Integrated Circuits For Demonstration
    . 74LS04 HEX Inverter
    . 74LS153 Dual 4-to-1 Selector/Multiplexer
    . 74LS83 4-Bit Binary Full Adder
- 5 x 5mm Red LEDs
- 5 x 470 ohm Resistors
- 5 Volt Regulated DC Power Supply
- Assortment of 22 AWG Jumper Wires
- 2250 point Breadboard
- Logisim Program version 2.7.1

DESIGN DESCRIPTION

The decimal range for a 4-bit binary number in two's complement notation is formulated by the equation -2^{N-1} to $2^{N-1}-1$ resulting in a possible range of values from -8 to +7.

Base 10	Base 2
7	0 1 1 1
6	0 1 1 0
5	0 1 0 1
4	0 1 0 0
3	0 0 1 1
2	0 0 1 0
1	0 0 0 1
0	0 0 0 0
-1	1 1 1 1
-2	1 1 1 0
-3	1 1 0 1
-4	1 1 0 0

-5	1 0 1 1
-6	1 0 1 0
-7	1 0 0 1
-8	1 0 0 0

The binary representation for positive numbers remains unchanged from the unsigned binary representation, however the representation for negative numbers in the table above is derived from the addition of constant 1 to one's complement of its positive counterpart.

Two's complement notation avoids the problems produced from other methods of "negative" binary representation, such as zero having two binary forms (-0 and +0). Accordingly, two's complement notation is sufficient when performing arithmetic computations involving negative binary numbers which coincidently a key element to this particular lab.

Per guidelines for lab number 4, the logic circuit has to implement the following functionality using two 1-bit control signals, labeled S_0 and S_1 …

S0	S1	Function
0	0	A + 1
0	1	A - B
1	0	A + B
1	1	A - 1

The inputs A and B for the circuit are 4-bit binary numbers in two's complement form and represented by A_3, A_2, A_1, A_0 and B_3, B_2, B_1, B_0.

The four arithmetic functions can be rewritten as follows, where addition is the only arithmetic computation involved...

S0	S1	Function
0	0	A + 1
0	1	A + (-B)
1	0	A + B
1	1	A + (-1)

Thus the binary representation for input A is not altered and all four bits (A_3, A_2, A_1, A_0) can be connected directly to the 4-bit adder (74LS83). However, the binary representation for input B and the constant 1 have been changed to their negative counterpart in the case of subtraction, this can be accomplished using two's complement form.

In order to select the appropriate arithmetic function and likewise the corresponding binary representation for input B and constant 1 it only seems fitting that a collection of 4-to-1 multiplexers be used to implement such functionality. Each of the four bits for input B (B_3, B_2, B_1, B_0) of the 4-bit adder (74LS83) are associated with an individual 4-to-1 multiplexer sharing common control signals S_0 and S_1 .

To represent the subtraction function selected when $S_0 = 0$ and $S_1 = 1$, input B needs to be in two's complement form. This is done a two step manner, where all the bits for input B are inverted and then a constant 1 is added to the result. The inversion for the bits is done through a single NOT gate applied at the second input to each individual MUX, whereas the addition of a constant 1 is done by sending a "Hi" signal to the carry-input of the 4-bit adder (74LS83). This 2-step procedure gives the two's complement form for input B.

To represent the subtraction function selected when $S_0 = 1$ and $S_1 = 1$ input B is given by the one's complement form of positive 1, 1110 (this is hardwired bitwise to each MUX). The necessary two's complement form is obtained by adding a constant 1 to 1110, or in this case by sending a "Hi" signal to the carry-input of the 4-bit adder (74LS83).

A truth table shows the single bit inputs of B and the corresponding output for each individual MUX (Q_3, Q_2, Q_1, Q) based on the control signals S_0 and S_1 …

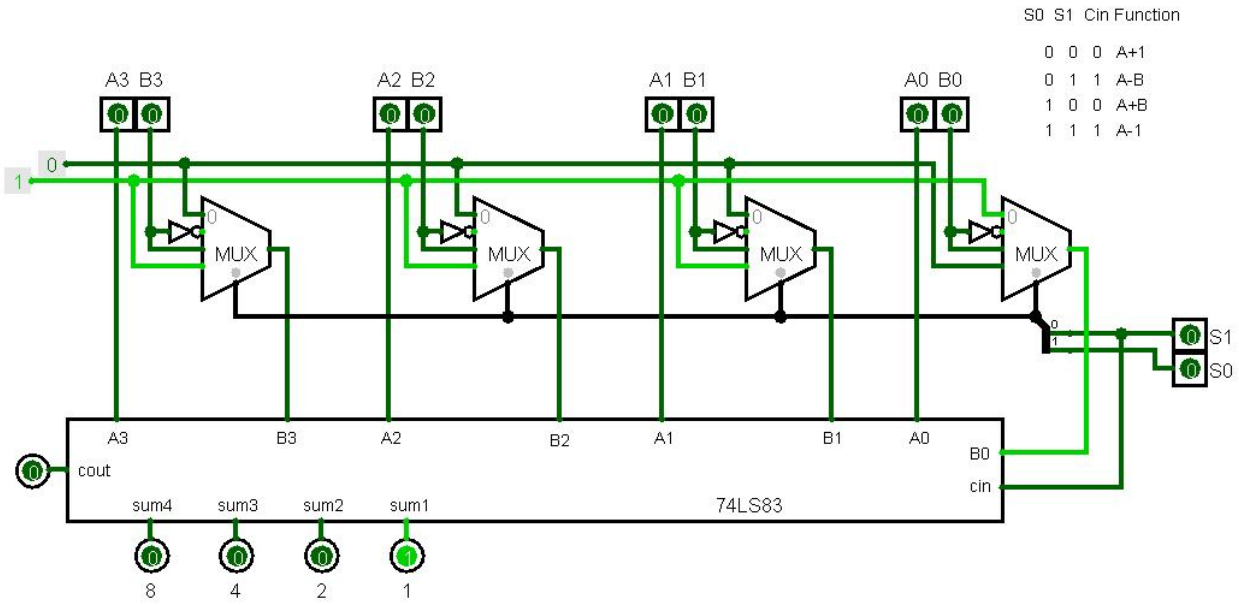
B3	B2	B1	B0	S0	S1	Q3	Q2	Q1	Q
0	0	0	1	0	0	0	0	0	1
0/1	0/1	0/1	0/1	0	1	B3'	B2'	B1'	B0'
0/1	0/1	0/1	0/1	1	0	B3	B2	B1	B0
1	1	1	0	1	1	1	1	1	0

The correlation between the control signals and the carry-input for the 4 bit adder is shown below (notice that the carry-in is "Hi" when the arithmetic function is subtraction, the result of adding a constant 1 to one's complement and acquiring two's complement form)...

S0	S1	Carry-In	Function
0	0	0	A + 1
0	1	1	A - B
1	0	0	A + B
1	1	1	A - 1

Clearly the carry-input for the 4-bit adder (74LS83) is driven by the control signal S_1; no further logic is needed to implement the carry-input functionality and each individual MUX has been defined in terms of its inputs/outputs.

Below is a logic design diagram for the circuit…

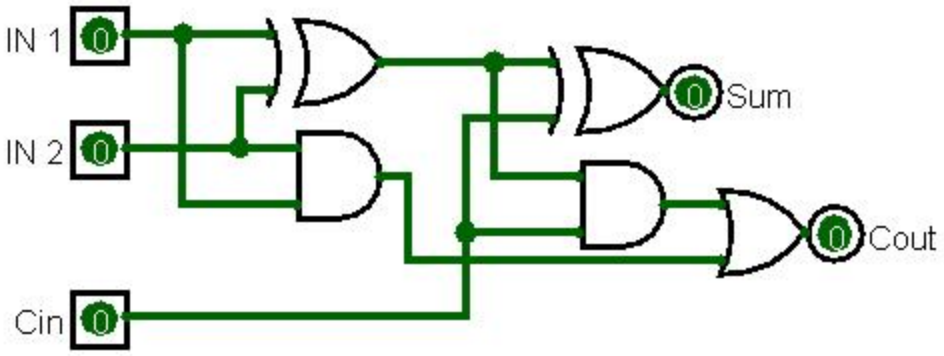


OBSERVATIONS AND DATA ANALYSIS

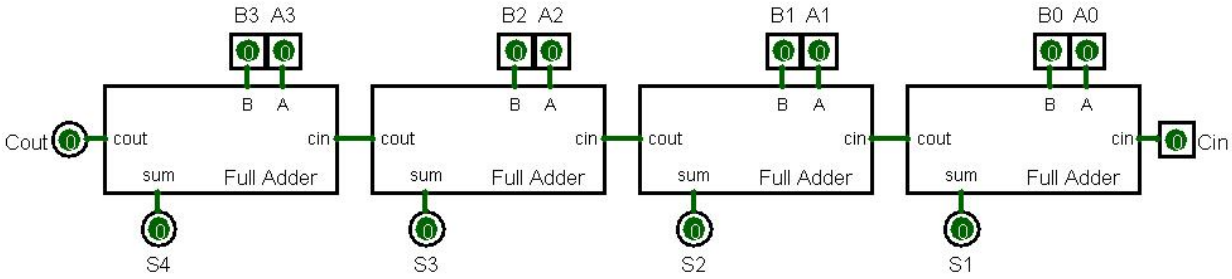
When designing the circuit using the logic simulation program Logisim there were errors encountered with the "adder" units packaged within the program. To sufficiently test the circuit a custom programmed 4-bit adder was created using some basic features of Logisim.

The 4-bit adder mimics the functionality of the TTL 74LS83 IC used in the lab for this project, so that the physical implementation would be similar to the logic simulation. The 4-bit adder is designed using the method of "Ripple-Carry" where the carry-input of the adder is moved through-out each of the four full-adders inside the circuit. In reality the 74LS83 uses a more efficient method called "Fast-Carry Lookahead" but this requires a more time to implement inside Logisim and not beneficial to the overall logic simulation as Logisim does not account for real-world propagation delay in circuits.

The design for the custom 4-bit adder began with the creation of a full-adder circuit. Full-adders can calculate a proper 2 bit output (sum and carry-out) for three 1-bit inputs (A, B, and carry-in). A logic diagram is shown below…



The next step was to chain four full-adders together using the carry-out of one to drive the carry-in of an adjacent full-adder. There are four full-adders to handle the 4-bits of each input correctly. This circuit is an essential 4-bit Ripple Carry Adder and is shown below…



DISCUSSION AND CONCLUSION

This lab introduce the basic concepts of arithmetic functions using combinational logic and required that knowledge gained from previous projects was applied to the overall design of the circuit for lab number 4.

By simplifying the logic for the four arithmetic functions using two's complement notation it became fairly easy to implement the circuit on the breadboard, especially since the use of a 4-bit adder (74LS83) condensed most of the logic that would have otherwise been extremely tedious to implement physically.

QUESTIONS AND ANSWERS

1. Is the output valid for the following combinations:

a. S0 = 0, S1 = 0, A = 7, B = 3?

From the values of the control signals this is the arithmetic function of *A+1*, where the value of B is ignored and constant 1 is used. The result should be *7+1=8*, but the output is a 4-bit binary number represented in two's complement and according to the equation -2^{N-1} to $2^{N-1}-1$ the range of values is from -8 to 7, thus the output is NOT valid for this input combination.

b. S0 = 0, S1 = 1, A = 7, B = -3?

From the values of the control signals this is the arithmetic function of *A-B*, where the value of B is represented using two's complement. The result should be *7-(-3)=10*, but the output is a 4-bit binary number represented in two's complement and according to the equation -2^{N-1} to $2^{N-1}-1$ the range of values is from -8 to 7, thus the output is NOT valid for this input combination.

c. S0 = 1, S1 = 0, A = -4, B = -5?

From the values of the control signals this is the arithmetic function of *A+B*. The result should be *-4+(-5)=-9*, but the output is a 4-bit binary number represented in two's complement and according to the equation -2^{N-1} to $2^{N-1}-1$ the range of values is from -8 to 7, thus the output is NOT valid for this input combination.

2. What is the range of inputs that will produce the valid output for all the functions?

Since the output is a 4-bit binary number represented in two's complement and according to the equation -2^{N-1} to $2^{N-1}-1$ the range of values is from -8 to 7.

For the arithmetic function *A+1* the range of inputs for A is -8 to 6.

$A+1 \leq 7$ *then* $A \leq 6$

For the arithmetic function *A-1* the range of inputs for A is -7 to 7.

$A-1 \geq 8$ *then* $A \geq -7$

For the arithmetic function *A+B* the range of inputs for A and B is -4 to 3.

If A= 3 and B= 3 then $A+B \leq 7$

If A= -4 and B= -4 then $A+B \geq -8$

For the arithmetic function *A-B* the range of inputs for A and B is -4 to 3.

If A= 3 and B= -3 then $A-B \leq 7$

If A= -4 and B= -4 then $A-B \geq -8$

To consider a single range of input values for all the functions, a valid output would be dependent upon the numbers from -4 to 3 as the result of these numbers when manipulated by the functions are represented in two's complement notation.