Laboratory 3-4:
Arithmetic Logic Unit (ALU) Design
EE 203 Digital Systems Design

Date Performed:	November 8, November 15 2024
Teaching Assistant:	Ayşenaz Ezgi Ergin
Teaching Assistant:	Bora Kayaoglu
Instructor:	Dr. Mustafa Tanıs Institution:
MEF University	
Filiz Köse	**042301071**
Kerim Ercan	**042401167**

1 Objective

Objective of this laboratory session is the hierarchical design of a 4-bit Arithmetic Logic Unit (ALU). Additionally, you shall learn about testing and demonstrating how an ordinary ALU works. A set of rough hierarchical design steps shall be provided with this document, however, if you find more efficient or more elegant ways to implement the subcomponents of the ALU, please go ahead and use them. Make sure you justify your design steps and explain them clearly when you write your lab report with your group member/s.

Your designs should be as least gate-consuming design as possible. Make sure also that you use minimum wiring for circuit efficiency and save yourself from complicating your job. Each question has its points written down next to it. Although point-wise percentage of the Prelab. work is relatively low, it is quite essential to the overall lab experience, particularly to be able to get the experiments done correctly.

Teaching assistants are free to give extra bonus points (up to 10) depending on your performance in the lab., how neat you work with electronic components, actively engage in or collaborate with your group member/s.

2 Prelab work

An *n*-bit ALU is a combinational circuit that can perform logic and arithmetic micro-operations on a pair of *n*-bit operands. The operations performed by an ALU are controlled by a set of function-select inputs. In this lab work, you will implement a design (out of many options) of 4-bit ALU with 2 function-select inputs: Select S1 and S0 inputs. To simplify the ALU circuit, only the arithmetic part will be covered in this lab and the logic part will be left as an exercise (Do it!, you will need it anyway in the multi-cycle CPU project). The functions performed by the ALU are specified in Table 1. You will be adding more functionality to your current design of ALU such as multiplication or comparisons later but for now we will stick to this table.

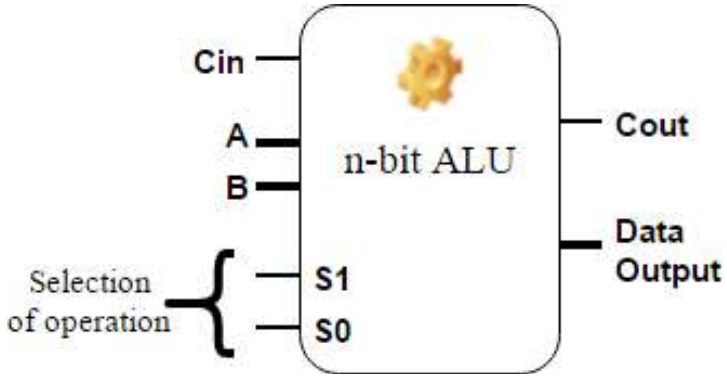
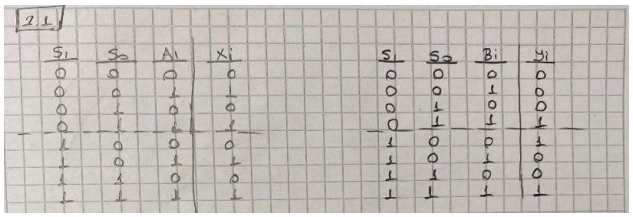


Figure 1: A simplified top-level block diagram of the n-bit ALU where each input (A and B) and the output buses are of n-bit long.

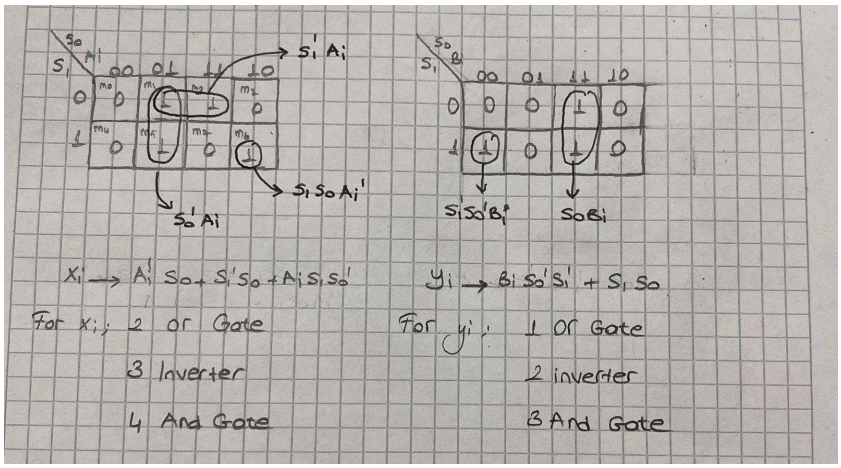
Since we define "subtraction" operation for the ALU, we need to encode negative numbers. There are various ways of encoding negative numbers into our number system. For instance, the magnitude and the sign can be encoded separately. In this lab experiment however, we will use 2's complement representation of numbers. So if we use *n*-bit ALU, the *n*-bit inputs *A* and *B* shall lie within the range -2^{n-1} to 2^{n-1}. For example, when $n = 4$, we have the range [8, +7]. The list of the corresponding codewords for 4-bit 2's complement number system are listed in the following table. Note that operations like +5 + (+6) or (–7) (+4) cannot be represented in our number system, thus our ALU design must have *overflow* and *underflow* detection mechanism, otherwise the result shall be interpreted wrong.

0 = 0000	+4 = 0100	-8 = 1000	-4 = 1100
+1 = 0001	+5 = 0101	-7 = 1001	-3 = 1101
+2 = 0010	+6 = 0110	-6 = 1010	-2 = 1110
+3 = 0011	+7 = 0111	-5 = 1011	-1 = 1111

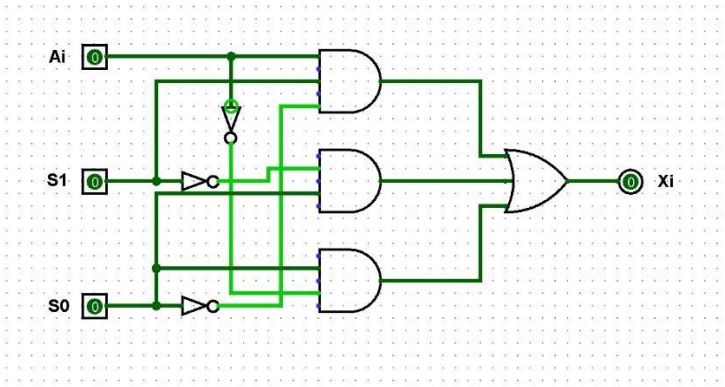
2.1 (7pts) We shall use "divide and conquer" method to hierarchically design the 4-bit ALU by first designing a 1-bit ALU module or so called "bit-slice". Though, there are different ways for the design of this approach, our ALU will have the characteristic architecture as shown in Fig. 2. Draw two truth tables, one admitting inputs S_1, S_0, A_i and giving an output X_i and the other admitting inputs S_1, S_0, B_i and giving an output Y_i in light of the arithmetic operations as presented in Table 1. A sample table is drawn for you to fill out in the Appendix section. Explain your reasoning by referring to Table 1.

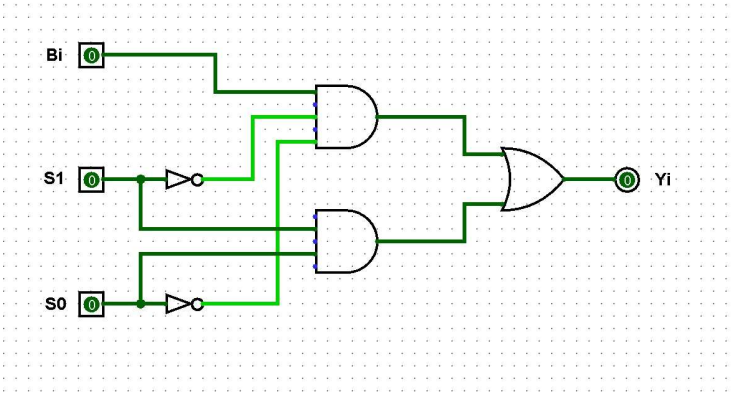


2.2 (8pts) Draw Karnough maps for each output X_i and Y_i and find the minimum gate implementation for logic X and Y.



2.3 (8pts) Draw the logic diagram in Logisim and verify its correctness.





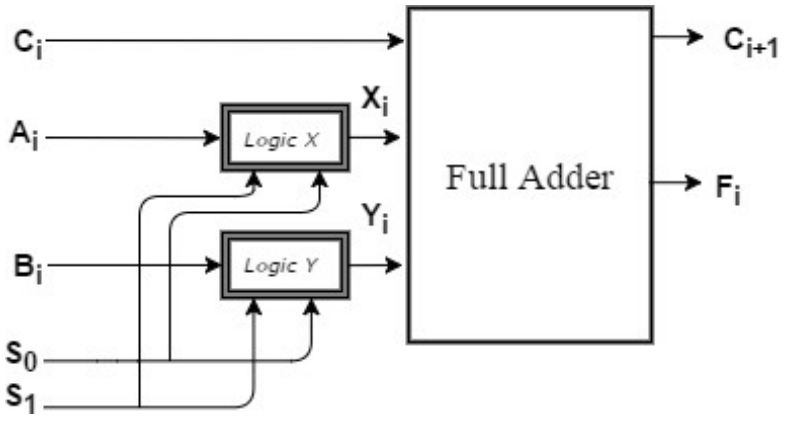


Figure 2: Block Diagram of the 1-bit ALU component.

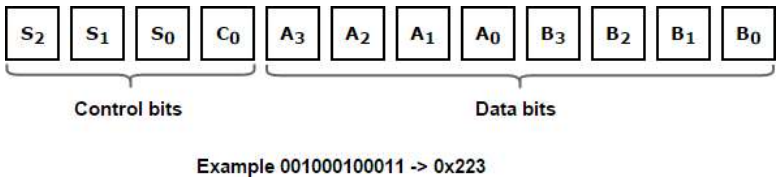


Figure 3: The structure of the instruction as well as data bits.

To test our ALU, we need to define our own *instruction* structure. In this way, a combined data and control bit sequences would be correctly understood by the ALU and the corresponding operations shall be executed. Our structure consists of three 4-bit HEX letters per instruction as shown in Fig. 3. An example could be 0*x*223 meaning that we add 3 to 2 while the literal S_3 is redundant and set to 0 for all instructions. As we add more operations to the instruction set, we shall use this extra bit to select among the multiple ALU functionalities.

2.4 (7pts) Find the instructions in HEX number system to perform the following two operations. *Operation index 1* is defined by ((4 + 3) —1) + 2 and the *operation index 2* is defined by − (6 + 2) − 4. In the Appendix section, a table is given for you to fill out.

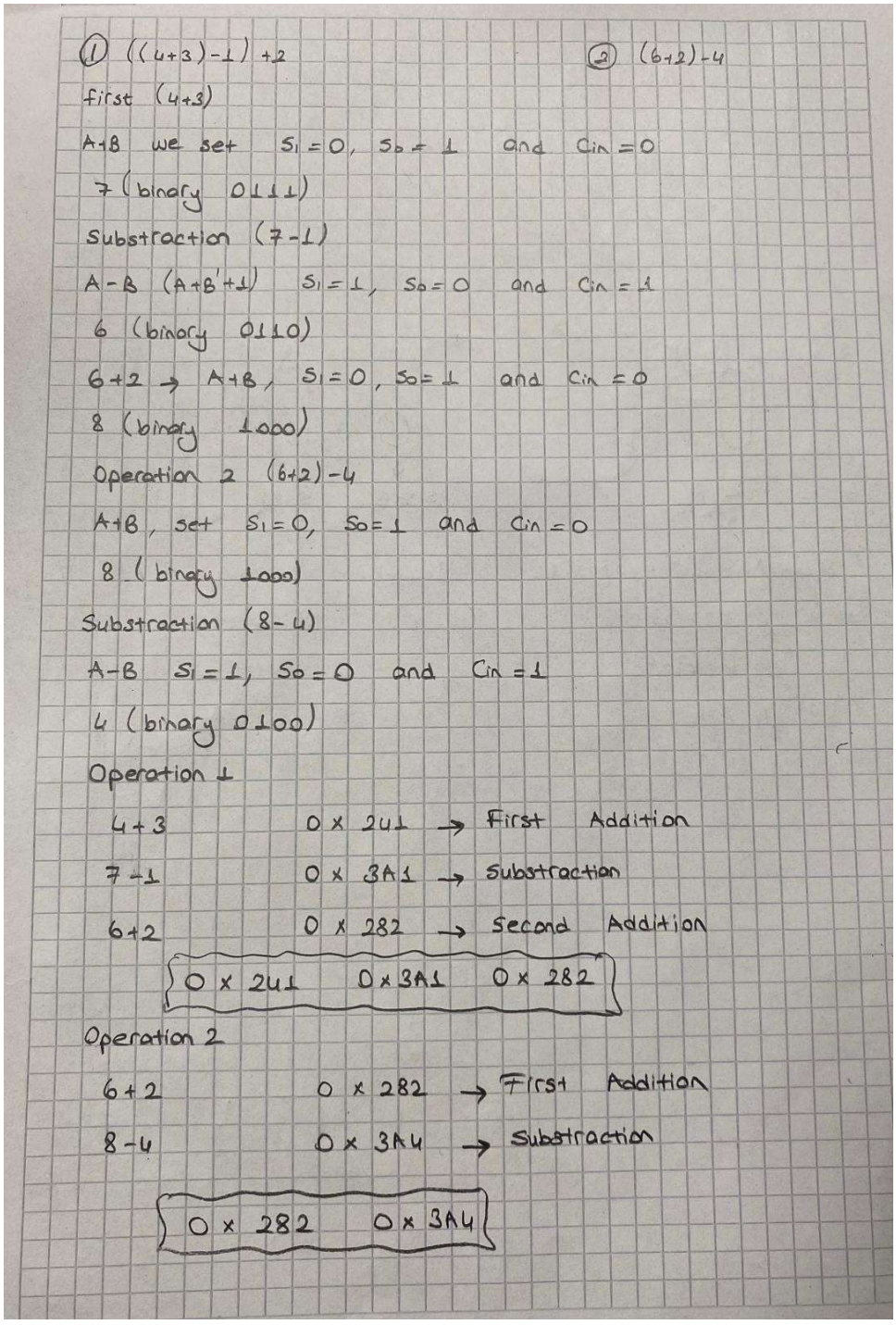


Table 1: The set of functions by ALU.

S_1	S_0	C_{in}	function	operation
0	0	0	A	Transfer A as is
0	0	1	A + 1	Increment A
0	1	0	A + B	Add B to A
0	1	1	A + B + 1	Increment the sum A + B
1	0	0	A + B'	A + 1's complement of B
1	0	1	A - B = A + B' + 1	Subtract B from A
1	1	0	B + A'	B + 1's complement of A
1	1	1	B - A = B + A' + 1	Subtract A from B

3 Experimental work

Caution: In our experiment, we will only implement a 1-bit ALU on breadboard due to space constraints.

3.1 (15pts) Implement the logic X and logic Y using multiplexers. How many multiplexers do you need? Draw the corresponding logic diagram in Logisim and verify its correctness then implement on breadboard. Compare this circuit with the implementation of the previous section using only AOI gates that you implement at 2.3 .

3.2 (20pts) First implement the 1-bit ALU slice in Logisim.
Then determine the model names of the IC components you need to implement it using conventional AOI gates.
Implement the 1-bit ALU slice model with IC gates on breadboard.
How many of these IC components you would need to implement a 2-bit ALU?

4 Postlab

4.1 (15pts) Answer the same question in 3.2 only using multiplexers in and implement the 2-bit ALU in Logisim . Let us define

$$S_3, S_2, S_1, C_0, A_1, A_0, B_1, B_0 \tag{1}$$

to be our codeword. Determine the output by testing the input sequences 0*x*13, 0*x*14, 0*x*5*D* on your design. Create a table with all the input numbers and the corresponding output numbers. When do you have an error or inconsistency at the output of the ALU? Indicate why.

5 Discussion of Experiment Results

5.1 (5pts) Please discuss the results of the experiments. Generate three questions and responses.

5.2 (5pts) A bill of materials (sometimes bill of material, BOM or associated list) is a list of the raw materials, sub-assemblies, intermediate assemblies, sub-components, extra parts and the quantities of each needed to manufacture/implement a certain functionality or a product. Generate and discuss the BOM for this laboratory.

5.3 (5pts) Suggest and discuss one more alternative design methods for your 4-bit ALU. Can you get rid of the delay due to the arithmetic operations based on the ripple logic?. This actually means there might be better designs for the 4-bit ALU of this experiment. Do a quick search to find out these designs. Share your experience.

5.4 (5pts) As suggested by the previous question, there are better design approaches that can decrease the execution speed as well as the silicon area used. Despite that, why did we adapt such a design approach? What is the advantage of our approach? (Hint: Think about designing 8-bit ALU or 16-bit ALU ...)

6 Appendix

This appendix is provided for filling out the truth tables or extra lab sheets that you need to complete before you turn in your lab report. Please refer to the lab report preparation guidelines document for details about writing the lab report write-up. It is important that you follow those guidelines otherwise our evaluation work will be darn hard to manage! Thank you for your cooperation.

a. The truth tables for logic X and logic Y (Q2.1)

S_1	S_0	A_i	X_i (Logic X)	S_1	S_0	B_i	Y_i (logic Y)
0	0	0		0	0	0	
0	0	1		0	0	1	
0	1	0		0	1	0	
0	1	1		0	1	1	
1	0	0		1	0	0	
1	0	1		1	0	1	
1	1	0		1	1	0	
1	1	1		1	1	1	

b. Set of instructions for the 4-bit ALU design (Q2.4)

#	A	B	operation	result	underflow	overflow
1						
1						
1						
2						
2						

Please hand in all the tables you fill out in this lab document when you hand in your report. Also, please keep your logism designs at a secure place so that when we learn about memory elements and simple processor design later in the class, you can pull your design off the shelf and use it. Contact me or the TA with any questions you might have.