Experiment # 11

Design and Implementation of a 4 - bit ALU

Objectives:

The objectives of this lab are:

- To design a 4-bit ALU
- To experimentally check the operation of the ALU

Overview

An Arithmetic Logic Unit (ALU) is a combinational circuit that performs logic and arithmetic micro-operations on a pair of n-bit operands (ex. A[3:0] and B[3:0]). The operations performed by an ALU are controlled by a set of function-select inputs. In this lab you will design a 4-bit ALU with 2 function-select inputs: Select S1 and S0 inputs. The functions performed by the ALU are specified in Table 1.

Table 1: Functions of ALU

S1	S0	C0	FUNCTION	OPERATION
0	0	0	A	Transfer A
0	0	1	A + 1	Increment A by 1
0	1	0	A + B	Add A and B
0	1	1	A + B + 1	Increment the sum of A and B by 1
1	0	0	A + B'	A plus one's complement of B
1	0	1	A – B	Subtract B from A (i.e. B' + A + 1)
1	1	0	A' + B	B plus one's complement of A
1	1	1	B – A	B minus A (or A' + B + 1)

A block diagram is given in Figure 1.

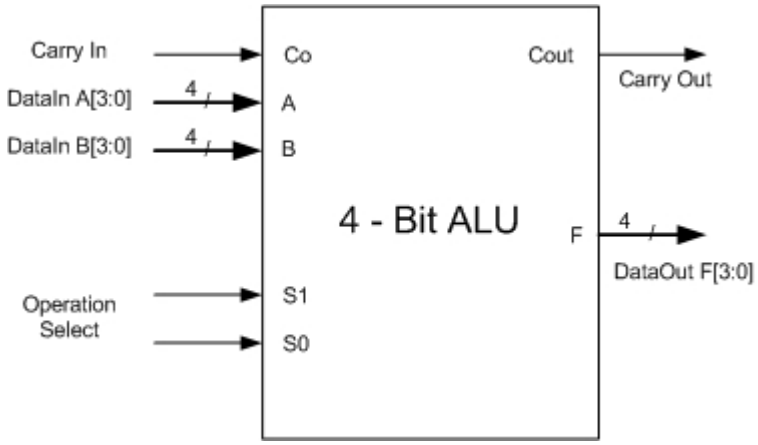

Figure 1: Block diagram of the 4-bit ALU.

We will be using two's complement system of notation while dealing with arithmetic operations in our ALU. This has a number of advantages over the sign and magnitude representation such as easy addition or subtraction of mixed positive and negative numbers. Recall that the two's complement of a n-bit number N is defined as,

$$2^n - N = (2^n - 1 - N) + 1$$

The last representation gives us an easy way to find two's complement: take the bit wise complement of the number and add 1 to it. As an example, to represent the number -5, we take two's complement of 5 (=0101) as follows,

```
5    0 1 0 1    -->      1 0 1 0   (bit wise complement)
                         + 1
                        ---------
                         1 0 1 1   (two's complement)
```

Numbers represented in two's complement lie within the range **- (2^{n-1})** to + **(2^{n-1} - 1)**. For a 4-bit number this means that the number is in the range of -8 to +7. There is a potential problem we still need to be aware of when working with two's complement, namely overflow and underflow as is illustrated in the examples below,

```
          0 1 0 0     (=carry Ci)
+5          0 1 0 1
+4    +     0 1 0 0
-----     ---------
+9        X 1 0 0 1     = -7!
```

also,

```
          1 0 0 0     (=carry Ci)
-7          1 0 0 1
-2    +     1 1 1 0
-----     ---------
-9        1 0 1 1 1     = +7!
```

Both calculations give the wrong results (-7 instead of +9 or +7 instead of -9) which is caused by the fact that the result +9 or -9 is out of the allowable range for a 4-bit

two's complement number. Whenever the result is larger than +7 or smaller than -8 there is an overflow or underflow and the result of the addition or subtraction is wrong. Overflow and underflow can be easily detected when the carry out of the most significant stage (i.e. C_4) is different from the carry out of the previous stage (i.e. C_3). In our lab, the inputs ***A and B have to be presented in two's complement to the inputs of the ALU***.

Design strategies

When designing the ALU we will follow the principle "Divide and Conquer" in order to use a modular design that consists of smaller, more manageable blocks, some of which can be re-used. Instead of designing the 4-bit ALU as one circuit we will first design a one-bit ALU, also called a ***bit-slice***. These bit-slices can then be put together to make a 4-bit ALU.

There are different ways to design a bit-slice of the ALU. One method consists of writing the truth table for the one-bit ALU. This table has 5 inputs (S1, S0, C_0, A_i and B_i) and two outputs F_i and C_{i+1}.

Inputs

- One switch is used to define bits of the first operand (A)
- A second switch is used to define bits of the second operand (B)
- The bits of A & B are shifted one by one into 2 shift registers that will hold the two operands. A push button may be used as a clock for the 2 shift registers. A total of 4 clocks is required to input both operands (***beware of bouncing problem***).
- Selects, S0 and S1, and the C0 inputs will be defined by 3 other switches.

Outputs

- The operands A and B should be displayed on the two leftmost 7-segment display digits as HEX digits. Thus an input of 1010 should be displayed as the character "A", 1011 as "b", 1100 as "C", 1101 as "d", 1110 as "E", and 1111 as F.
- Another LED should display the overflow flag V.
- The result of arithmetic operation (2's complement representation) is to be shown on the rightmost 7-segment display digit.
- An LED should be used to indicate the sign of the result.
- Another LED should display the carryout bit Cout.
- You need to time-multiplex the inputs of the 7-segment display digits such that all three digits (A, B & result) are displayed (in a manner similar to what was done to display the alarm clock time).
- Figure 3 shows a schematic of the overall system, consisting of the ALU, Decoder and Switching circuit, and Displays on the Digilab board.

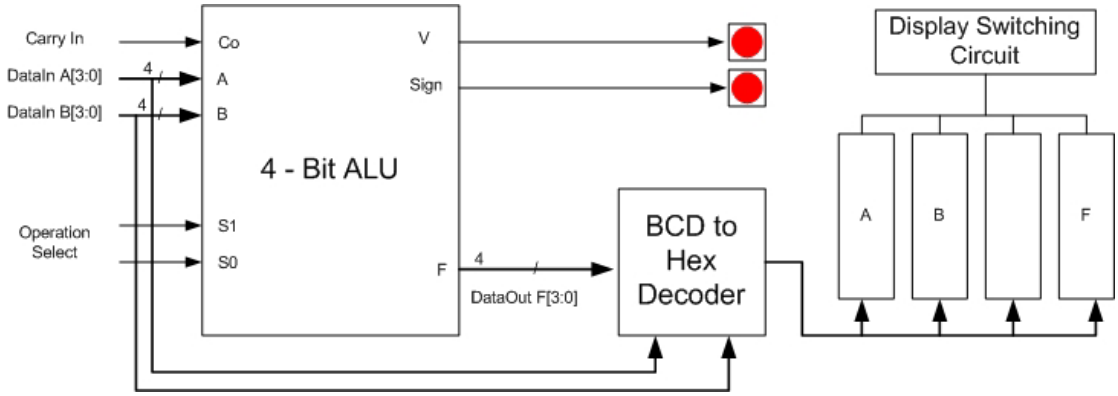

Figure 2: Overall system, including the 4-bit ALU and display units.

Pre Lab

Do the following tasks prior to coming to the lab. Write the answers to all questions on a sheet prior to coming to the lab. You will also need to include answer to the pre-lab questions in your lab report.

- Read section 7-7 (the arithmetic/logic unit) of you text book (pp. 360-365).
- Design the HEX-to 7-segment display decoder.
- Derive the logic for the arithmetic overflow condition.
- *ALU Unit Design.* You have different choices to design and implement the ALU unit. A particularly attractive method is one that makes use of readily available modules such as Full Adders. The arithmetic unit basically performs additions on a set of inputs. By proper choice of the adder inputs, one can perform a range of operations. This approach is shown in Figure 3. The only blocks that need to be designed are the A Logic and B Logic circuits. You can make use of full adders available in the SPARTAN library.

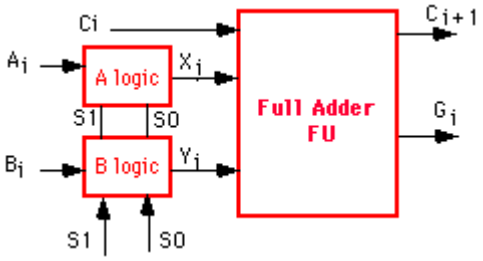

Figure 3: Schematic block diagram of the arithmetic unit.

A. Give the truth tables for the Xi and Yi functions with inputs S1, S0 and Ai, and S1, S0 and Bi, respectively. Fill out the following tables. Notice that in definition table I of the ALU, the variable C_0 acts as the Carry input. Depending on the value of C_0, one performs the function on the odd or even entries of the definition table I. As an example the first entry is "transfer A" (for C_0=0) while the second one is "A+1" (for C_0=1); similarly for A + B and A + B + 1, etc.

Table 2: Truth tables for the A and B logic circuits.

S1	S0	Ai	Xi (A Logic)	S1	S0	Bi	Yi (B Logic)
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. Give the K-map for Xi and Yi functions. Find the minimum realization for Xi and Yi.
C. Draw the logic diagram for Xi and Yi.
D. Design the circuit that detects over- or underflow.
E.

- *BCD to Hex Decoder Design*
- *Design the switching* circuit that is needed to use the seven-segment displays.

In-lab:

Your task is to design and implement the 4-bit ALU. Follow the guidelines of the pre-lab in designing the 4-bit ALU. You will create a project (see tasks below) with a top-level schematic that will correspond to the one as shown in Figure-2. This top-level file will be instantiating several macros which are shown as black boxes in the Figure. These macros will have to be developed separately.

Task 1:

- Design the input circuit using two 4 bits shift registers in order to shift-in inputs A and B serially.

Task 2:

- Implement the 4-bit ALU.
- Design and add the circuit that detects overflow, *'V'* to your schematic.
- Simulate the full 4-bit ALU and verify that all operations function properly.

Task 3: Inputs and outputs

- For the A and B inputs, use the two 4 bits shift register for inputs A and B and use switches SW1 and SW2

- For the Select and CarryIn inputs (S1, S0 and Cin), you will need to use the general purpose switches SW3-SW5.
- The overflow and sign of the result is to be displayed using LEDs.
- Display the Hexadecimal values of the inputs A and B on the two leftmost 7-segment display digits and the result of the arithmetic operation on the rightmost 7-segment display digit.
- Specify the pin locations (using the constraint editor or on the schematic), implement and test the ALU.
- It will be instructive to do a timing simulation to get an idea about the overall speed of the circuit.
- Give the working demo to your lab instructor.

Post Lab

You have to hand in a lab report that contains the following:

- Section on the Pre-lab explaining the design of each block and giving the answers to each task.
- Section on the lab experiment:
  - Brief description of the goals.
    b. Brief explanation of the design approach, the overall schematic and of each macro.
  - Discussion of the results indicating that the circuit functions properly.

- Conclusion and discussion.