EXPERIMENT 10

10. Design of a 4-bit Arithmetic Unit

10.1 Objectives:

The objectives of this lab are:

- To design a 4-bit Arithmetic Unit (AU)
- To experimentally check the operation of the AU

10.2 Overview

An Arithmetic Unit is a combinational circuit that performs arithmetic micro-operations on a pair of n-bit operands (ex. A[3:0] and B[3:0]). The operations performed by an AU are controlled by a set of function-select inputs. In this lab you will design a 4-bit AU with 2 function-select inputs: Select S1 and S0 inputs. The functions performed by the AU are specified in Table 10.1. A block diagram is given in Figure 10.1

Table 10.1: Functions of AU.

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)

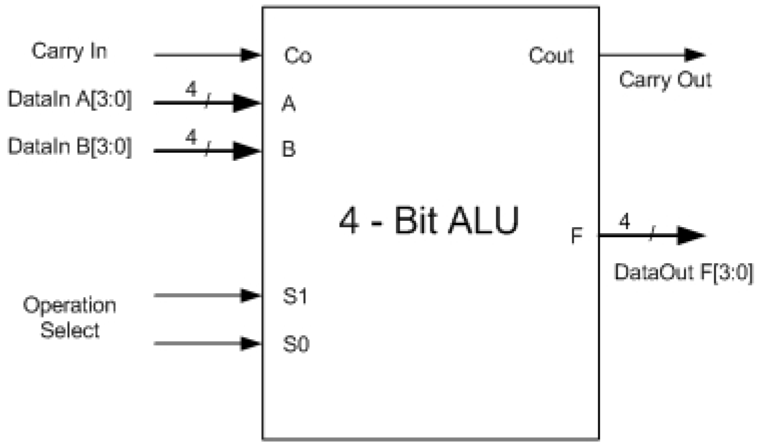

Figure 10.1: Block diagram of the 4-bit AU.

We will be using two’s complement system of notation while dealing with arithmetic operations in our AU. 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:

$$\mathbf{2^n - N = (2^n - 1 - N) + 1}$$

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,

$$\begin{array}{llrl} 5 & 0\,1\,0\,1 \text{ -->} & 1\,0\,1\,0 & \text{(bit wise complement)} \\ & & \underline{+\,1} & \\ & & 1\,0\,1\,1 & \text{(two's complement)} \end{array}$$

Numbers represented in two’s complement lie within the range $\mathbf{-(2^{n-1})}$ to + $\mathbf{(2^{n-1} - 1)}$. 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,

$$\begin{array}{llrl} & & \mathbf{0}\;\mathbf{1}\;0\;0 & \text{(=carry Ci)} \\ +5 & & 0\;1\;0\;1 & \\ \underline{+4} & \underline{+} & \underline{\quad 0\;1\;0\;0} & \\ +9 & & \cancel{0}\;1\;0\;0\;1 & = -7! \end{array}$$

Also,

$$\begin{array}{llrl} & & \mathbf{1}\;\mathbf{0}\;0\;0 & \text{(=carry Ci)} \\ -7 & & 1\;0\;0\;1 & \\ \underline{-2} & \underline{+} & \underline{\quad 1\;1\;1\;0} & \\ -9 & & \cancel{1}\;0\;1\;1\;1 & = +7! \end{array}$$

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 AU***.

10.3 Design strategies

When designing the AU 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 AU as one circuit we will first design a one-bit AU, also called a *bit-slice*. These bit-slices can then be put together to make a 4-bit AU.

10.4 Pre-Lab

- Read section 7-7 (the arithmetic/logic unit) of you text book (pp. 360-365).
- Give the truth tables for the X_i and Y_i functions with inputs S_1, S_0 and A_i, and S_1, S_0 and B_i, respectively. Fill out the Table 10.2. Notice that in definition Table 10.1 of the AU, 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 10.2: Truth Tables for the A and B Logic Circuits.

S_1	S_0	A_i	X_i (A Logic)	S_1	S_0	B_i	Y_i (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	

- Give the K-map for X_i and Y_i functions. Find the minimum realization for X_i and Y_i.
- Draw the logic diagram for X_i and Y_i.
- Design the circuit that detects over- or underflow.
- Bring the *multiplexed seven-segment display* macro design to the lab.

10.5 In-lab:

1. Build a macro for logics A and B. this macro should have four inputs (S_1, S_0, A_i, and B_i) and two outputs (X_i and Y_i). Call it *bit-slice*.
2. You need two 4-bit (FD4CE) registers to store the values A and B.
3. You need four instances of the macro you built in step 1.
4. You need four full adders. You can build your own or fined one in the Xilinx library.
5. Connect four switched (SW1, SW2, SW3, and SW4) to the inputs of the two 4-bit registers (D3, D2, D1, and D0) respectively.
6. Use SW5 to enable one of the registers and disable the other.
7. Use a button switch (BTN1) as a clock for the two registers.
8. Connect the outputs of the registers to the proper inputs of the *bit-slice* macro.
9. Connect the outputs of the *bit-slice* macros to the full adder as in Figure 10.2.

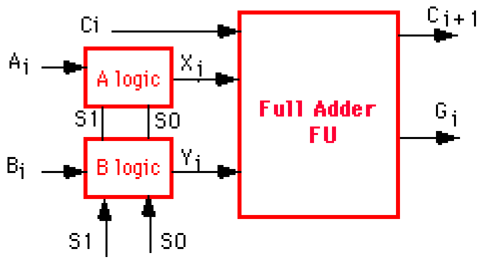

Figure 10.2: Schematic Block Diagram of the Arithmetic Unit.

10. Use the switches (SW6, SW7, and SW8) for the control signals S_1, S_0, and C_{in} respectively.
11. Import the *multiplexed seven-segment display* macro you built earlier and connect the outputs of the register A to the most left display and the output of register B to the second left display. Connect the outputs of the adder to the right most display as in Figure 10.3.
12. Use three LEDs for over-flow, carry-out, and sing signals.
13. Download you design and demonstrate it to the lab instructor.

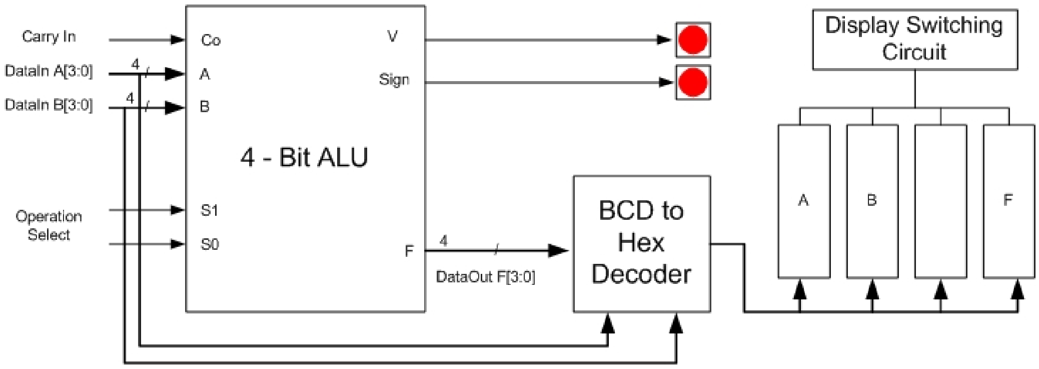

Figure 10.3: Overall System, Including the 4-Bit AU and Display Units.