LAB 6: Arithmetic Logic Unit (Element of a CPU)

OBJECTIVES
The objective of this lab is to design and build an arithmetic logic unit (ALU) to study how it can be used with registers and buses to form the components of a basic central processing unit (CPU) in a computer. In addition, an introduction is given in using a hardware definition language (VHDL) to specify a digital component.

INTRODUCTION
The ALU you will design consists of the following:

1. The device has one 4-bit wide INPUT bus and one 4-bit wide OUTPUT bus. These buses are used to bring data to and from the ALU.
2. REGA and REGB are 4-bit wide registers (i.e. 4 D Flip-Flops) that are used to hold data originating from MUX A and MUX B. MUX A & B (each containing four 4-input multiplexers) are used to connect a particular bus to REG A and REG B. A bus is connected in the following manner:

<table>
<thead>
<tr>
<th>MSA1</th>
<th>MSA0</th>
<th>MSB1</th>
<th>MSB0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

   - Bus Selected as Input to REGA/B

3. The outputs of REGA and REGB are thus fed back to MUX A and MUX B inputs as well as to a combinatorial logic block. The combinatorial logic block is used to perform data complement, addition, ANDing, ORing and shifts.
4. MUX C consists of four 8-input multiplexers. They are used to select a particular operation for outputs. The three select lines MSC2:0 function as follows:

   - MSC2:0 (Most Significant Bit is on the left)
     - 111 => complement of REGA Bus to OUTPUT Bus
     - 110 => REGA Bus AND REGB Bus to OUTPUT Bus
     - 101 => REGA Bus OR REGB Bus to OUTPUT Bus
     - 100 => REGA Bus to OUTPUT Bus
     - 011 => REGB Bus to OUTPUT Bus
     - 100 => shift REGA Bus right one bit to OUTPUT Bus
     - 001 => shift REGA Bus left one bit to OUTPUT Bus
     - 000 => REGA Bus Plus REGB Bus Plus Cin to OUTPUT Bus & Cout (There is an external Cin and an external Cout for the ALU)

CONTROL BYTE and PROGRAMMING
A Control Byte can now be defined as the bit pattern on: X, MSA1:0, MSB1:0, MSC2:0 (Note the first bit “X” is a “don’t care” to complete 8 bits (byte).

By setting the appropriate control byte bit pattern, it is now possible to load data into the system and perform all the ALU functions.

PRE-LAB REQUIREMENTS

Part I. VHDL specification of a 4-to-1 MUX
1. Enter the design of the 4-to-1 MUX using behavioral VHDL as a dataflow description (i.e., simple signal assignment statements to specify its behavior).
2. Compile the design using the Quartus II compiler.
3. Verify its operation by perform a timing simulation using Quartus II. (You can use the same .vwf file from your previous lab.)

Part II. Graphical specification of the ALU
Use the following components to enter the design of the ALU using the Quartus II graphical editor:

- Use the MUX designed in Part I as components for the four MUX A’s and four MUX B’s:
- REGA and REGB are 4-bit wide registers, using (4) D Flip-Flops.
- The Combinatorial Logic contains primitive gates and the 4-bit ripple adder designed in Lab 4.
- Use the “81mux” from the “others.maxplus2” library for the four instances of MUX C’s.

Special Note: To simplify debugging, minimize the drawing of lines for connections. Instead, use signal labels to avoid “spaghetti-like” connections.
LAB 6: Arithmetic Logic Unit (Element of a CPU)

Part III. Simulation of the ALU
1. Derive the control bytes required to perform each of the following (Note that this is equivalent to writing simple “programs”):
   a) Load register A with 1001, complement the A data and then store it into B. Preserve the data in A during the complement operation.
   b) Load A with 1011 and B with 0110, sum them and then store the result in A. Preserve B during the add operation.
   c) Load A & B with the same values as part b, bit wise AND the registers and then store the result in B. Preserve the contents of register A during this operation.
   d) Load A & B with the same values as part b, bit wise OR the registers and then store the result in A. Preserve the contents of register B during this operation.
   e) Load A with 1001, shift it left 1 bits and then store it in B. Preserve the contents of register A during this operation.
   f) Now write a program to do the following:
      • Load A with the number 2
      • Multiply the number by 4
      • Add 4 to it
      • Divide the number by 4

2. Perform the simulation in Quartus II for steps “a” through “f” above.

IN-LAB REQUIREMENTS
1. Download your design to the breakout board.
2. Connect LEDs and switches to the breakout board; verify it functions as specified in the Pre-Lab Part III.

QUESTIONS
1. Can you load REG A’s contents to REG B and REG B’s contents to REG A in the same clock cycle? Explain the basis for your answer.
2. Can you load REG A from the INPUT bus and REG B from any other bus other than the INPUT bus (i.e. REGA, REGB or OUTPUT) in the same clock cycle? Explain your answer.
3. Suppose you’re not allowed to use a flip-flop that has an asynchronous CLR or SET, how can you add a function that clears the contents of either A or B?