LAB 4: MSI Circuits & ALU

Objectives
This lab is designed to introduce the student to the functioning and design of an encoder (a common MSI element) and an arithmetic logic unit (ALU). The ALU will be a building block used in future labs.

Materials
- Your entire lab kit including your UF-3701 PCB and USB Blaster
- Read UF’s DAD Tutorial on our website
- Suggested Quartus Components:
  - In “others | maxplus2” library
    - 74153: Dual 4-Input MUX
    - 74283: 4-bit Adder
  - In “primitives | logic” library
    - nand2, nor2, etc.
  - In “primitives | pin” library
    - input, output
  - In “primitives | other” library
    - vcc, gnd
- Possible, but NOT recommended:
  - In “megafun|c|gates” library
    - mux (although I used the 74153)
- Read the following from the lab or Software/Docs links on our website. Printout and bring the first two of these documents to lab.
  - UF-3701 Programming WARNING!
  - 3064 Pin Definitions
  - MAX 3000A specification sheet
- You must adhere to the Lab Rules and Policies document for every lab. Re-read, if necessary. Documents must be submitted through Canvas and on paper for every lab. All pre-lab files must submitted BEFORE the beginning of your lab.

Part 1. The Priority Encoder
In this part of the lab, you will design a priority encoder. The encoder has four input lines (I1-I4) and three output lines (C0:C2) as shown in Figure 1.

![Figure 1a: Priority Encoder](image1)

![Figure 1b: Priority Encoder](image2)

Note: Two of the inputs and two of the outputs are active-low and the other signals are active-high. The notation in Figure 1a (with no bubbles for active-low inputs or outputs) is common. Sometimes the /I4 will appear as I4 with a bar over it. I suggest you use the drawing of Figure 1b for your designs in this course and in the future.

When one of the four inputs is asserted (i.e., true), the encoder outputs a binary code corresponding to the active input. For example, if I3 is the only input asserted, the output code will be “011”, or a binary “3”. If no inputs are asserted (i.e., true), the encoder will output “000”. If more than one input is true at the same time, the encoder will output a code corresponding to the highest numbered true input. For example, if both I2 and I4 are true, the encoder will output a “100”. In this case, the code for I4 is output because it has the highest priority.

Pre-Lab Requirements (Encoder)
1. Write the truth table for the priority encoder.
2. Derive both MSOP and MPOS logic expressions for each of the encoder outputs using K-maps.
3. Design the encoder circuit using mixed-logic and only gates available on 7400 and 7402 chips. (It is usually easier to design first on paper.) Design the circuit using either MSOP or MPOS equations.
4. If you designed the circuit in part 3 on paper, now design it using Quartus (Lab4_Enc_74) Sand then simulate the complete mixed-logic circuit in Quartus. Annotate the simulation. Verify its operation with a truth table, a voltage table, and your Quartus simulation.
5. When you use Quartus to design with a 74’153 MUX, you will find that the select lines are labeled A and B. Which select inputs correspond to A and B? Which inputs correspond to S0 and S1? By creating a new Quartus circuit (in a new file, Lab4_MUX) using only this chip (available in the library “others | maxplus2”) and inputs and outputs. Simulate this design using enough different input combinations to prove that the 74’153 works as you suspect it should. Note: For every “new” part you use (like the 74’153), if you are not sure of a pins function, you should verify its operation before using it in a circuit. Include this simulation in your pre-lab document.
   - We can use the DAD to verify the operation of a 74’153 MUX chip as follows. After reading the DAD tutorial, do the following.
     - Generate a 2.0 kHz (= 2000 Hz) 7-bit pattern for the inputs of the MUX using the signal generator (as discussed in the Digital Pattern Generator section of the DAD Tutorial).
     - Use the generated signal from the DAD as inputs to your MUX circuit (with NO switches connected). Connect the DAD’s DIO6 to one of the MUX’s select lines, DIO5 and DIO4 to the same MUX’s select lines (preferably DIO5 to S1 and DIO4 to S0, if you know which is which), and DIO3-DIO0 to the MUX’s normal inputs (which may be named X3-X0 or similar). Now use the Logic Analyzer with the corresponding signal lines (e.g., DIO6 – DIO0) and the MUX’s output. Attach Vcc and Gnd to your 74’153, connect the DAD to the appropriate 74’153 pins, and view the DAD’s logic analyzer waveforms. Do they match the voltage waveform generated in the Quartus simulation for the MUX?
LAB 4: MSI Circuits & ALU

Include a screen shot of the DAD’s output in your pre-lab document.

6. Now design a circuit (Lab4_Enc_MUX) to create the encoder with a single 74’153 chip and, only if necessary, the gates available on 74’00 and/or 74’02 ICs. Label all gates and show device pin numbers. Your design must (non-trivially) use a 74’153. Note that the 74’153 includes two 4-input MUX’s. Simulate your designs using Quartus to verify that your design works.

7. Simulate the designs of parts 4 and 6 together (Lab4_Enc_both), in the same project in a single block diagram file (bdf). To do this you must use the same 4 inputs but different outputs. This simulation will test the logical and voltage equivalence of your two solutions. Display all above materials to your TA at lab time.

8. For each of the above designs, include the truth tables, logic equations, voltage tables, circuit schematics (with chip and pin numbers for circuits that you will construct), and annotated simulation results, to your pre-lab document. As usual, all pre-lab material (including your archive files) must be submitted through Canvas prior to the start of your lab.

9. Build the priority encoder circuit (in which you used the MUXs) at home on your breadboard (using parts available in your lab kit, but NOT your UF-3701 CPLD board). Your circuit must be brought (already wired-up) to lab.

Pre-lab Question (Encoder)
How many total input lines could we have on the priority encoder (i.e., how many more can we add) without increasing the number of lines used for the output binary code? Answer the question in part 5 above.

In-Lab Requirements (Encoder)
1. Demonstrate your functioning priority encoder to your TA using switches for inputs and LED’s for outputs.
2. After demonstrating your circuit to your TA, disassemble your circuit. Leave your switch and LED’s in order to demo parts 2 and 3 of the lab.

Part 2. UF-3701 board - PROGRAMMING YOUR 3064 CPLD

Warning 1: Tri-state all unused CPLD pins, as described in the Quartus Tutorial.

Warning 2: Do not program the board while the board is connected to anything (other than power and ground) on your breadboard. I suggest that you keep a corner of your breadboard available for programming your CPLD (with the chip dangling off the end of the breadboard, thus efficiently minimizing the utilized breadboard area). Create your circuit design for a particular project elsewhere on your breadboard. Circuit designs for the remainder of the semester will consist of only switch circuit, LED circuits, and your UF-3701 boards. After the CPLD on your UF-3701 board is programmed (on a corner of your breadboard), remove the UF-3701 board from the programming location, and insert it into your already designed circuit. For efficiency, again design with your UF-3701 board located with the chip dangling off the edge of the breadboard.

The ground connection is the most important one for the UF-3701 board. Connect BOTH of the pins labeled GND on your UF-3701 board (see Figure 4) to the ground on your breadboard. Similarly, connect BOTH power pins, labeled +3.3V on your UF-3701 board (see Figure 4), to the power on your breadboard. When using the DAD, you must also connect the DAD’s ground (black wire) to the ground on your breadboard.

Pre-Lab Requirements (Encoder Designed with CPLD)
1. Download your Priority Encoder design to the 3064 CPLD on the UF-3701 board. If you are not sure how to program your UF-3701 board, please reread the Quartus Tutorial. If you are still not sure, go to a TA office hour before your lab to get help.
2. Connect switches and LEDs to the UF-3701 board to verify that it functions as required. (Note: Inputs from the non-CPLD design can ALSO be used for this design.) Even if you can’t program your board, you can still wire up your switches and LEDs. Although you can have your input switch circuits going to both your discrete element encoder design AND the CPLD encoder design, only ONE of the sets of outputs should be connected to the LED circuits at a time. Do NOT put power to these switches and LEDs while your un-programmed CPLD is connected.
3. Take a screen shot of your CPLD bdf with pin numbers and include this in your pre-lab report.
4. You will NOT demo this circuit in lab, so after proving to yourself that it works, remove it from your breadboard.

In-Lab Requirements (Encoder Designed with CPLD) None. Download and demonstrate your functioning CPLD-designed priority encoder.

Part 3. The Arithmetic Logic Unit (ALU)
In this part of the lab, you will design a 4-bit ALU (Lab4_ALU). The ALU has two 4-bit inputs, two function-select inputs, and a carry input. It has a 4-bit function output and a carry output as described Table 1 and shown in Figure 2.

<table>
<thead>
<tr>
<th>S1:S0</th>
<th>Action</th>
<th>Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>bit-wise AND</td>
<td>F = A and B</td>
</tr>
<tr>
<td>01</td>
<td>sum</td>
<td>F = A + B + Cin, Cout</td>
</tr>
<tr>
<td>10</td>
<td>complement of A</td>
<td>F = /A</td>
</tr>
<tr>
<td>11</td>
<td>bit-wise OR</td>
<td>F = A or B</td>
</tr>
</tbody>
</table>

Table 1. ALU functions.
Independent circuits must be designed for each of these actions. The function-select inputs (S) determine which F is used for the ALU output (see Figure 3). The carry output should be set only when an addition causes a carry output. In all other operations the carry output should be false.

PROGRAMMING YOUR 3064 CPLD

Debug Outputs
It is often helpful to create extra output signals so that your circuit design is easier to debug (in simulation and in hardware). I suggest adding outputs for each of the functional blocks (F_{\text{and}}, F_{\text{sum}}, F_{\text{not}}, and F_{\text{or}}). If your design does NOT fit with these extra outputs, using a functional compilation/simulation or pick a large device in Quartus (like the last device in the MAX3000A family, EPM3512----). After you have thoroughly tested your circuit through simulation, you can remove the unneeded outputs if it is necessary to make the design fit in your EPM3064ALC44-10.

Pre-Lab Questions (ALU)
If you were to make a complete (un-abbreviated) truth table for your ALU, how many rows would it need? (The answer should tell you why I did not want a simulation including ALL input combinations.)

What changes would be necessary to the design already made if I wanted to add two more functions, F=A XOR B, and F=A EQ B, where XOR is exclusive-or and EQ is equivalence?

In-Lab Requirements (ALU)
Download and demonstrate your ALU design using your 3064 CPLD on the UF-3701 board, switch circuits, and LED circuits. Verify that it operates correctly.