ELEC 204
Digital System Design
LABORATORY MANUAL

Experiment 3:
4-bit hexadecimal Decoder
&
4-bit Increment by N Circuit

College of Engineering
Koç University

**Important Note:** In order to effectively utilize the laboratory sessions, you should read the manual and prepare the experiments before the sessions.
1. **Objectives:**
   - To design and simulate a 4-bit hexadecimal decoder
   - To design and simulate a 4-bit increment by N circuit.
   - To implement the both circuits on FPGA
   - To test the decoder and increment circuits

2. **Equipments:**
   - Digilent Basys2 board
   - Pentium PC

3. **Procedure:**
   
   i. **Read the Background parts**
   
   ii. **Perform preliminary work before coming to lab**
   - Derive the truth table of 4-bit decoder
   - Result is to be displayed on 7-segment
   - Designing the 4-bit decoder
   - Designing the 4-bit increment by N circuit

   iii. **Perform the experimental work during the lab**
   - Implement your circuits on Foundation Software
   - Simulate the circuits
   - Verify the correctness of your circuits
   - Implement one of the circuit in the FPGA
   - Test the circuit

4. **Part1 - Designing a $4 \times 7$ decoder**

   **Background**

   In this part, we want to display each hexadecimal digit (0, 1, ..., 10, A, b, C, d, E) on the 7-segment display.

   ![7-segment display diagram](image)

   So in order to display, for example, the hexadecimal digit ‘A’ (10 in decimal), the LEDs with labels $S_2, S_3, S_6, S_4, S_1, S_3$ should be ON, and the other LED $S_0$ should be OFF. This means that the corresponding pins $[P_{24}, P_{18}, P_{19}, P_{23}, P_{26}, P_{20}]$ should have logic-1 (5Volts).

   Since we can represent each hexadecimal digit by means of 4 bits, we’ll implement a $4 \times 7$ decoder.
Preliminary Work - 1:

Design a decoder that maps each 4-bit hexadecimal digit (0,1,...,A,...,E) into appropriate pins (which control the LEDs of 7-segment display) in a way that we can view the digit on the 7-segment display. To do this, first you have to learn about how the leds of 7-segment display becomes ON and OFF.

Our Basys2 board has cathodes for controlling the LEDs of 7-segment display:

Anodes are connected via transistors for greater current

Cathodes are connected to Xilinx device via 100Ω resistors

Figure 4. Common anode 7seg display

As it can be seen from the figure LEDs turn ON only when the corresponding cathode signals are LOW.

We will derive the truth table according to this fact as seen in below figure for 1 and 2.

- Derive the truth table of the decoder (which will be a $16 \times 11$ table)
- Considering the truth table, design an optimum circuit

<table>
<thead>
<tr>
<th>Digit Shown</th>
<th>Cathode Signals</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>a b c d e f g</td>
</tr>
<tr>
<td>0</td>
<td>0 0 0 0 0 0 1</td>
</tr>
<tr>
<td>1</td>
<td>1 0 0 1 1 1 1</td>
</tr>
<tr>
<td>2</td>
<td>0 0 1 0 0 1 0</td>
</tr>
</tbody>
</table>

truth table with LOW signals meaning ON

7- segment display of PEGASUS

5. **Part 2 - Designing a 4-bit Increment by N Circuit**

**Background:**

The Half Adder adds two bits and generates a *sum* and a *carry-out* output. However, to be useful for adding binary *words*, one needs a Full Adder which has three inputs: the *augend*, the *addend*, and a *carry-in*. The following example illustrates the addition of two 4-bit words $A = [A_3, A_2, A_1, A_0]$ and $B = [B_3, B_2, B_1, B_0]$ where $A_3$ is the MSB of the 4-bit-word ‘A’ and $A_0$ is the the LSB of the 4-bit-word $A$. In the same way, $B_3$ and $B_0$ are the MSB and LSB bits of the 4-bit-word $B$, respectively.

![Figure 1: Addition of binary numbers](image)

The addition can be split-up in bit slices. Each slice performs the addition of the bits $A_i$, $B_i$ and the carry-in bit $C_i$ (i.e. carry-out bit of the previous slice). Each slice consists of a full adder, illustrated below.

![Figure 2: Block diagram of a full adder (FA)](image)

A circuit that implements a full adder is given in Figure 3 below.
The circuit consists of two XOR gates, two AND gates and one OR gate.

**Preliminary Work - 2:**

Starting with a 4-bit full adder circuit use *contraction* method, that is removal of redundancy from circuit to which input fixing has been applied, to design a 4-bit increment by N circuit. The 4-bit increment value N will be

\[
N = \begin{cases} 
  d_0 - 2 & \text{if } d_0 \geq 8 \\
  d_0 + 2 & \text{if } d_0 < 8 
\end{cases}
\]

where \(d_0\) is the least significant hexadecimal digit of your student ID. For example, if your student ID is 20060222 = 0x132183e, then \(d_0 = \text{e}\), and \(N = \text{e}-2 = \text{c} = 1100\).

**Experimental Work - 1:**

- Implement your 7-segment decoder circuit using VHDL.
- Simulate the circuit.
- Verify the correctness of your circuit.
- If testing on the FPGA, download your *.bit file into Basys2 board and test it. While testing use 4 logic-switches SW3, SW2, SW1 and SW0 as inputs and the pins L14, H12, N14, N11, P12, L13, M12 that control the 7-LEDs as output. Then, try all the possible 16 combinations and see the result on the 7-segment display.

**Experimental Work - 2:**

- Create a new VHDL module called ‘Lab3’. In this schematic you will be using buses. A bus is an array of signals, and is displayed as a thicker line than a regular wire. When using a bus, the naming convention is to give the bus a name in the form of `bus_name[X:Y]`, where `bus_name` is the variable name and X is the most significant bit and Y is the least significant bit. Individual signals in the bus are named `bus_nameN` where N is between X and Y. Individual signals can be accessed by wiring a regular wire from a bus to a terminal and giving the internal node the name corresponding to the desired signal (or by using the ‘Draw Bus Taps’ tool, try it to see what it does).
- After creating the ‘IncN’ schematic that implements your 4-bit increment by N design in the preliminary work, simulate the increment by N for a few different inputs using both hexadecimal and decimal notation (functional mode simulation). Again, only 1 schematic should be listed in the main project window. At this point, Lab2.sch is the main top-level schematic and thus should be the only schematic file listed. In the Foundation Logic Simulator, the state of a bus signal can be set to a hexadecimal number using the ‘Bus’ button within the states selection window. To change the base the number is displayed in, select the bus you want to change then go to ‘Signal:Bus’ and select the desired base.
- If testing your circuit, use the logic switches of the Basys2 board as input to the increment by N circuit. Your inputs will be a 4-bit number. You should display the 4-bit result of the increment by N on 7-segment display by using the 4x7 decoder implemented in **Part-1**, the carry out bit \(C_4\) and overflow flag can be displayed using two leds on your Basys2 board.
6. **Assessment (Lab Report)**

1. Brief description of the lab experiment including the goals and discussion on the theory of operation.
2. Design and schematics of your experiment circuit.
3. Present simulation waveforms.
4. Review of the results indicating that the circuit functions properly. You can for instance give a truth table and indicate that for each entry the logic simulator give the right results. Feel free to label the waveforms to indicate the proper operation.
5. Conclusions and discussions.