Number Conversion

CSE 2300W

Objectives

This assignment is to design a circuit that convert a 4-bit binary number to several representations in different number systems. You will also learn how to use LogicWorks to build and simulate simple digital circuits. More specifically, you will learn the following things.

1. Given logic equations, construct a logic circuit with basic gates in LogicWorks.
2. Provide input to your circuit using binary switches and display circuit outputs on binary probes, and hex and ASCII displays.
3. Exhaustively test a circuit to ensure it works as specified.

Prerequisites

This probably is your initial foray into the digital design process. A knowledge of basic gates is necessary and you should also understand number systems. Since you have not learned how to design a digital circuit yet, the logic equations are given for conversions from binary to decimal. Later in this course, you will be able to write and optimize similar equations yourself.

You will need LogicWorks software, a web browser, and a PDF reader. We assume you have all these in the lab.

Background

Decimal, binary, hexadecimal, and octal are some number systems you will use a lot. The decimal number system (base 10) is used for dealing with money, measuring, etc. The computer you are using works in binary (base 2). But we do not read binary numbers as well as decimal numbers. Therefore, for programmers, octal (base 8) or hexadecimal (base 16) is commonly used since these systems are easy to convert to/from binary and also easier to read. For example, considering 255\textsubscript{10}, which of ‘FF’ and ‘11111111’ would you like to use? Both of them represent the decimal number 255. The first is in hexadecimal and the second is in binary.

This lab helps you understand how numbers can be represented in one format and then converted to another. Although your digital circuit may work in binary, very often, you need to display values in other systems. It is important to understand how these conversions are performed with hardware.

We have discussed the theory behind the number conversions in lectures. The major thing to remember is the meaning of a base or radix for the number system being used. There are
methods that convert numbers from any base to any other, which are helpful in cases like binary to decimal and vice versa. But, in cases involving only binary, octal, and hexadecimal, the conversions become much easier because of the power relationship between their bases.

Consider a binary number 100110010001. It can be easily converted into an octal by breaking the bits into groups of three.

\[ 100 \quad 110 \quad 010 \quad 001 \]
\[ 4 \quad 6 \quad 2 \quad 1 \]

To perform the hex version you can break the binary bits into groups of four.

\[ 1001 \quad 1001 \quad 0001 \]
\[ 9 \quad 9 \quad 1 \]

One of the important aspects of number systems (considering integers only) is the use of preceding zeros. For example, a 7-bit binary number, 1010011 does not convert to hexadecimal by taking the first (most significant) bits 1010, converting to \( A_{16} \), and then converting the remaining 3 binary bits to \( 3_{16} \). The correct way is to add a preceding zero to make the number of binary bits a multiple of 4. In the previous example, the bits should be put in groups this way: 0101 0011. Converting each group to a hex digit gives you \( 53_{16} \).

### Specifications

You will build a circuit that takes a 4-bit binary input and displays it in four different number systems: binary, octal, decimal, and hexadecimal. The circuit also displays the decimal number on ASCII displays.

Using LogicWorks, you are to build a circuit with the inputs and outputs arranged as shown in the following figure.

1. Four switches (far left) are used to input a 4-bit binary number.
2. The four probes (next to the switches) display the binary value of the input.
3. A hexadecimal display outputs the number as a hexadecimal digit.
4. The two pairs of Hexadecimal outputs will display the number in octal and decimal, respectively.
5. The ASCII displays (far right) also show the input value in decimal. But the input to an ASCII display has eight bits, instead of four bits going to a hexadecimal display.

After placing the inputs and outputs, you can follow these steps.

**Connect each input bit to its appropriate binary output probe** Since there is no conversion, the outputs can be connected directly to the inputs. After this step, you are able to read the inputs from the binary probes and observe the values changing when you click the switches.
Connect the inputs to the hexadecimal displays. The 4 pins on the hexadecimal display correspond to the 4 bits which make up the hexadecimal digit. The dot on the display indicates the pin corresponding to the least significant bit (LSB).

Connect the inputs to the octal displays. Some values represented by a 4-bit binary number need two octal digits, so we use two hexadecimal displays. Although we still use hex displays, you want to make sure only valid octal digits are displayed. It is not difficult to figure how to connect inputs to pins. Think about the conversion process from base 2 to base 8 and the values you want to see on the displays.

Implement the binary-to-decimal conversion. The conversion from binary to decimal is a little bit more complicated. Suppose $A$, $B$, $C$, and $D$ are the four binary input bits. Two digits in the decimal number can then be generated with the following equations. You may ask which of the four bits is the right most bit, i.e., LSB. Staring at the equations for a moment, you will figure it out.

- **10's digit**
  - $1's = D \cdot C + D \cdot B$
  - $2's = 0$
  - $4's = 0$
  - $8's = 0$

- **1's digit**
  - $1's = A$
  - $2's = D' \cdot B + D \cdot C \cdot B'$
  - $4's = D' \cdot C + C \cdot B$
  - $8's = D \cdot C' \cdot B'$

Connect the inputs to the ASCII displays. Now you know the value of each decimal digit. You need to generate proper ASCII code for ASCII displays. Notice that the ASCII code (see Additional Reference section) for the numbers 0 - 9 all begins with the same 4 bits. So these four bits can be connected to constant 0 or 1. You can use the “GND” and “5V” components in LogicWorks.

Testing

Since this circuit is designed to convert a 4-bit binary number to a different number system, only 16 possible combinations exist. You should exhaustively try every possible input. A recommendation for systematically testing is to start from ‘0’ (0000$_2$) and count to ‘15’ (1111$_2$). As you input each binary number, make a table as in Table 1.

Verify that there are no mistakes (either in the circuit or as you copy the screen results to paper).

Deliverables

You will work individually in this lab. You do not need to implement the circuit physically with chips and wires. Once you have tested and debugged you circuit thoroughly in LogicWorks,
Table 1: Results

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>D C B A</td>
<td>L R</td>
<td>L R</td>
<td>L R</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0 0 0 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0 0 1 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0 0 1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

write a report, following the guidelines described in the first lab. Make sure your report has a table summarizing the number and type of chips you used, and a table as shown in the testing section. Your report and LogicWorks files must be submitted electronically on HuskyCT. You also need to hand in a hard copy of your report to TA by the due date.

Additional Reference

Table 2: ASCII Table

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>NUL</td>
<td>SOH</td>
<td>STX</td>
<td>ETX</td>
<td>EOT</td>
<td>ENQ</td>
<td>ACK</td>
<td>BEL</td>
<td>BS</td>
<td>HT</td>
<td>LF</td>
<td>VT</td>
<td>FF</td>
<td>CR</td>
<td>SO</td>
<td>SI</td>
</tr>
<tr>
<td>1</td>
<td>DLE</td>
<td>DC1</td>
<td>DC2</td>
<td>DC3</td>
<td>DC4</td>
<td>NAK</td>
<td>SYN</td>
<td>ETB</td>
<td>AN</td>
<td>EM</td>
<td>SUB</td>
<td>ESC</td>
<td>FS</td>
<td>GS</td>
<td>RS</td>
<td>US</td>
</tr>
<tr>
<td>2</td>
<td>!</td>
<td>&quot;</td>
<td>#</td>
<td>$</td>
<td>%</td>
<td>&amp;</td>
<td>'</td>
<td>(</td>
<td>)</td>
<td>*</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>:</td>
<td>;</td>
<td>&lt;</td>
<td>=</td>
<td>&gt;</td>
<td>?</td>
</tr>
<tr>
<td>4</td>
<td>@</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td>E</td>
<td>F</td>
<td>G</td>
<td>H</td>
<td>I</td>
<td>J</td>
<td>K</td>
<td>L</td>
<td>M</td>
<td>N</td>
<td>O</td>
</tr>
<tr>
<td>5</td>
<td>P</td>
<td>Q</td>
<td>R</td>
<td>S</td>
<td>T</td>
<td>U</td>
<td>V</td>
<td>W</td>
<td>X</td>
<td>Y</td>
<td>Z</td>
<td>[</td>
<td>\</td>
<td>]</td>
<td>`</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>a</td>
<td>b</td>
<td>c</td>
<td>d</td>
<td>e</td>
<td>f</td>
<td>g</td>
<td>h</td>
<td>i</td>
<td>j</td>
<td>k</td>
<td>l</td>
<td>m</td>
<td>n</td>
<td>o</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>p</td>
<td>q</td>
<td>r</td>
<td>s</td>
<td>t</td>
<td>u</td>
<td>v</td>
<td>w</td>
<td>x</td>
<td>y</td>
<td>z</td>
<td>{</td>
<td>—</td>
<td>}</td>
<td>~</td>
<td>~</td>
</tr>
</tbody>
</table>