Lecture 1: Digital Systems and Number Systems

Matthew Shuman

September 27th, 2010

The Digital Abstraction — 1.3 in Text

Analog Systems

Analog systems are continuous. Look at the analog clock in figure[1] The second hand on the clock rotates continuously around on the clock. Notice how the fraction of a second can be estimated by the distance of the hand between the second divisions. The fractional value is an indicator that the system is analog. What is the limiting factor of how precise an analog system can be read?

![Figure 1: This is a typical analog clock.](image)

Digital Systems

Digital systems are discrete. Look at the digital clock in figure[2] The second digit moves instantly from a 1 to a 2. There is no partial value for seconds and no way to zoom in to gain more precision. Precision could be increased by adding fractional digits that measure smaller amounts of time (tenths or hundredths of seconds).

Examples

Are for following items digital or analog?

1. The clock in the classroom?
2. The number of people in a room?
3. The voltage of a AA battery?
4. The DMM (Digital Multi Meter) measured voltage of a AA battery?
5. The time it takes to read this question?
6. The measured time it takes to read the previous question?

These examples should show that the physical properties (distance, time, voltage, and many others) are analog in nature, but after being measured they become digital information. The precision of this digital information depends directly on the quality of the measurement device.

**Number Systems — 1.4 in Text**

**Decimal Numbers — 1.4.1 in Text**

The standard numbers used in the US are base ten, this is the decimal number system. This system uses ten different symbols to represent numbers. These symbols are 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9. Each digit in a decimal number has a specific value. The number 1982 is represented by 1 thousand, 9 hundreds, 8 tens, and 2 ones.

**Binary Numbers — 1.4.2 in Text**

The number system typically used in digital logic will be base 2, this is the binary number system. This system uses two different symbols to represent numbers, 0 and 1.

The table below shows how the binary system progresses to more than three digits to represent even one decimal digit.

<table>
<thead>
<tr>
<th>Number System</th>
<th>Decimal</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zero</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>One</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Two</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>Three</td>
<td>3</td>
<td>11</td>
</tr>
<tr>
<td>Four</td>
<td>4</td>
<td>100</td>
</tr>
<tr>
<td>Five</td>
<td>5</td>
<td>101</td>
</tr>
<tr>
<td>Six</td>
<td>6</td>
<td>110</td>
</tr>
<tr>
<td>Seven</td>
<td>7</td>
<td>111</td>
</tr>
<tr>
<td>Eight</td>
<td>8</td>
<td>1000</td>
</tr>
<tr>
<td>Nine</td>
<td>9</td>
<td>1001</td>
</tr>
<tr>
<td>Ten</td>
<td>10</td>
<td>1010</td>
</tr>
</tbody>
</table>
The standard notation to avoid confusion of 10 in decimal with 10 in binary is to add the base of the system being used as a subscript.

\((10)_2\) indicates two, while \((10)_{10}\) indicates ten.

If no subscript is included then the number is assumed to be decimal.

**Octal and Hexadecimal Number Systems — 1.4.3 in Text**

Other useful number systems used in digital logic are octal and hexadecimal.

Octal uses 8 symbols to represent numbers. These symbols are 0, 1, 2, 3, 4, 5, 6, and 7.

Hexadecimal uses 16 symbols to represent numbers. These symbols are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F.

Octal is not used very frequently, but hexadecimal is very useful. It can be hard to read 8 bits of binary. The 1’s and 0’s tend to blend together and it can cause mistakes. The main use of hexadecimal is to condense long binary strings into shorter hexadecimal values. The table below shows the progression used for octal and hexadecimal.

<table>
<thead>
<tr>
<th>Number System</th>
<th>Decimal</th>
<th>Binary</th>
<th>Octal</th>
<th>Hexadecimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zero</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>One</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Two</td>
<td>2</td>
<td>10</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Three</td>
<td>3</td>
<td>11</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Four</td>
<td>4</td>
<td>100</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Five</td>
<td>5</td>
<td>101</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Six</td>
<td>6</td>
<td>110</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>Seven</td>
<td>7</td>
<td>111</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>Eight</td>
<td>8</td>
<td>1000</td>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td>Nine</td>
<td>9</td>
<td>1001</td>
<td>11</td>
<td>9</td>
</tr>
<tr>
<td>Ten</td>
<td>10</td>
<td>1010</td>
<td>12</td>
<td>A</td>
</tr>
<tr>
<td>Eleven</td>
<td>11</td>
<td>1011</td>
<td>13</td>
<td>B</td>
</tr>
<tr>
<td>Twelve</td>
<td>12</td>
<td>1100</td>
<td>14</td>
<td>C</td>
</tr>
<tr>
<td>Thirteen</td>
<td>13</td>
<td>1101</td>
<td>15</td>
<td>D</td>
</tr>
<tr>
<td>Fourteen</td>
<td>14</td>
<td>1110</td>
<td>16</td>
<td>E</td>
</tr>
<tr>
<td>Fifteen</td>
<td>15</td>
<td>1111</td>
<td>17</td>
<td>F</td>
</tr>
<tr>
<td>Sixteen</td>
<td>16</td>
<td>10000</td>
<td>20</td>
<td>10</td>
</tr>
</tbody>
</table>

The standard notation of using the subscript to indicate the base of the number still applies.

\((10)_8\) indicates eight, while \((10)_{16}\) indicates sixteen.

**Converting Between Bases — Not in Text**

Converting between bases is initially easy, but it can quickly become confusing when using uncommon bases. The main source of confusion is when the conversion requires math in different bases an example of this is below.

\((10)_7 + (6)_8 = (1101)_2\)

This demonstrates the difficulty of even adding two small numbers when using different bases.

A useful method is to convert the entire problem to base 10, and after the answer has been found, convert the answer to the desired base.
Converting Any Base to Decimal

A useful word to describe the base of a number system is **radix**. The radix of a decimal number is ten, and the radix of a binary number is two.

In order to understand how to convert numbers to base ten is it helpful to understand how decimal numbers are really organized. The number 365.24 is elaborated in the equation below.

\[
365.24_{10} = 3 \times 10^2 + 6 \times 10^1 + 5 \times 10^0 + 2 \times 1.1 + 4 \times 0.01
\]

This progression leads to the more general equation listed below.

\[
365.24_{10} = 3 \times 10^2 + 6 \times 10^1 + 5 \times 10^0 + 2 \times 10^{-1} + 4 \times 10^{-2}
\]

This form shows each digit has a different value. The 3 is in the 100’s place and the 6 is in the 10’s place.

The following equation converts a binary number into a decimal number.

\[
101.101_2 = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} = 4 + 1 + .5 + .125 = 5.625
\]

The final generalized form for this equation is shown below.

\[
d_2d_1d_0.d_{-1}r = d_2 \times r^2 + d_1 \times r^1 + d_0 \times r^0 + d_{-1} \times r^{-1} = \text{DecimalValue}
\]

Converting Decimal to Any Base

**Successive Quotients**

The method used to convert a decimal integer number to any other radix is called **Successive Quotients**. This method uses a recursive algorithm. Consider the example of conversion to binary below.

<table>
<thead>
<tr>
<th>Integer Quotient</th>
<th>Remainder</th>
</tr>
</thead>
<tbody>
<tr>
<td>(13)\text{$_{10}$}</td>
<td>$-$</td>
</tr>
<tr>
<td>(6)\text{$_{10}$}</td>
<td>1</td>
</tr>
<tr>
<td>(3)\text{$_{10}$}</td>
<td>0</td>
</tr>
<tr>
<td>(1)\text{$_{10}$}</td>
<td>1</td>
</tr>
<tr>
<td>(0)\text{$_{10}$}</td>
<td>1</td>
</tr>
</tbody>
</table>

The integer column begins with the decimal number being converted. The second row is the first row divided by the radix, the quotient on the left and the remainder on the right. This pattern is continued until the quotient returns 0. The conversion is read **least significant bit, LSB** on the top. The algorithm for this process is itemized below.

1. Divide the initial decimal number by the radix.
2. Place the remainder into the LSB (Least Significant Bit) digit of the converted result.
3. Divide the current quotient by the radix.
4. Place the remainder into the next LSB digit of the converted result.
5. Repeat Step 3 and 4 until the quotient is 0.
Successive Products

The process of converting decimal fractions is a similar recursive algorithm process. Consider the example of converting .375 into binary.

\[
\begin{array}{c|c}
\text{Fractional Product} & \text{Integer Component} \\
(0.375)_{10} & \text{--} \\
(0.750)_{10} & 0 \\
(0.5)_{10} & 1 \\
(0)_{10} & 1 \\
\end{array}
\]

\[.375_{10} \rightarrow .011_2\]

Below is the summary of this recursive process.

1. Multiply the initial decimal fraction by the radix.
2. Place the integer into the MSB (Most Significant Bit) digit of the converted result.
3. Multiply the current fraction by the radix.
4. Place the integer into the next MSB digit of the converted result.
5. Repeat Step 3 and 4 until the fraction is 0 or the required number of bits have been converted.

It is important to note that some common decimal numbers cannot be represented in binary. Look at the example converting .6 to 8 bits below.

\[
\begin{array}{c|c}
\text{Fractional Product} & \text{Integer Component} \\
(0.6)_{10} & \text{--} \\
(0.2)_{10} & 1 \\
(0.4)_{10} & 0 \\
(0.8)_{10} & 0 \\
(0.6)_{10} & 1 \\
(0.2)_{10} & 0 \\
(0.4)_{10} & 0 \\
(0.8)_{10} & 0 \\
(0.6)_{10} & 1 \\
(0.2)_{10} & 0 \\
\end{array}
\]

\[.6_{10} \rightarrow .100100010_2\]

Quickly Converting Between Binary and Hexadecimal — 1.4.4 in Text

Binary and hexadecimal numbers are often used in digital logic or in computer programming. It would be possible to convert between these two bases by using a decimal number as an intermediate step, but there is a faster and easier method. This method uses nibbles to divide a larger binary or hexadecimal into smaller conversions. There are some examples below:

<table>
<thead>
<tr>
<th>Binary</th>
<th>Hexadecimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1001 1100)_2</td>
<td>9 C_{16}</td>
</tr>
<tr>
<td>(0101 1010)_2</td>
<td>5 A_{16}</td>
</tr>
<tr>
<td>(1010 1011 0111)_2</td>
<td>A B 7_{16}</td>
</tr>
</tbody>
</table>

Each group of 4 binary digits is called a nibble and can be represented by one hexadecimal digit. This method only works when the bases in the conversion are a power of each other, in this instance \(2^4 = 16\) so it is valid to use the nibble short cut. It also works for binary to octal, \(2^3 = 8\).
Complements — Not really in Text

The meaning of complement is something required to make a thing complete. For example, salsa complements tortilla chips, beer complements pizza, an ice cream cone complements a hot summer day, and apple sauce complements pork chops. A key concept to explore is how two things complement each other. For example, when a piece of pizza is removed from a whole pizza the piece complements what is left behind and vice versa. Each of the 4 following complements use the same concept except in different bases and what is considered a complete number in that base.

Diminished Radix Complement

The diminished radix complements are called by the \( \text{radix} - 1 \). The diminished complement for a decimal number is the 9’s complement and 1’s complement for a binary number.

9’s Complement

The 9’s complement finds whatever is needed to make an entire set of 9’s. This is shown in the example below.

\[
\begin{array}{c|c}
\text{Finding the 5 digit 9’s complement of 1357} & \\
\hline
\text{All 9’s} & 99999 \\
\text{Initial Value} & -01357 \\
\text{9’s Complement} & 98642 \\
\end{array}
\]

The 5 digit 9’s complement of 1357 is 98642

1’s Complement

The 1’s complement finds whatever is needed to make an entire set of 1’s. This is shown in the example below.

\[
\begin{array}{c|c}
\text{Finding the 8 digit 1’s complement of 01101100} & \\
\hline
\text{All 1’s} & 11111111 \\
\text{Initial Value} & -01101100 \\
\text{9’s Complement} & 10010011 \\
\end{array}
\]

The 8 digit 1’s complement of 01101100 is 10010011

Radix Complement — 1.4.6 in Text

The radix complements are called by their \( \text{radix} \). The radix complement for a decimal number is the 10’s complement and 2’s complement for a binary number. The value that is considered the whole part is \( \text{radix}^{\text{digit}} \).

10’s Complement

\[
\begin{array}{c|c}
\text{Finding the 5 digit 10’s complement of 1357} & \\
\hline
\text{radix^5} & 100000 \\
\text{Initial Value} & -01357 \\
\text{10’s Complement} & 98643 \\
\end{array}
\]

The 5 digit 10’s complement of 1357 is 98643
2's Complement

Finding the 8 digit 2's complement of 01101100

| radix^8 | 100000000 |
| Initial Value | −01101100 |
| 2's Complement | 10010100 |

The 8 digit 2's complement of 01101100 is 10010100

Subtracting by Adding — 1.4.6 in Text

A key use of complements is to do subtraction. Building an adder in hardware is fairly easy, but a subtracter is much more difficult. Using the following mathematical property subtraction can be avoided. \( A - B = A + (-B) \) The following example shows how adding the radix complement can give an identical result as subtraction.

| Showing how to do 72532 - 3250 | Normal Way | Using Complements |
| Initial Value | 72532 | 72532 |
| Adding a 10's complement | −3250 | +96750 |
| Difference | 69282 | 169282 |

Note the answer has a positive carry out. This means that the difference is positive. If the carry out was 0, then the difference would be a negative number. Taking the radix complement of this negative number indicates the magnitude of the negative number.