Digital Logic Design

Introduction

A digital computer stores data in terms of digits (numbers) and proceeds in discrete steps from one state to the next. The states of a digital computer typically involve binary digits which may take the form of the presence or absence of magnetic markers in a storage medium , on-off switches or relays. In digital computers, even letters, words and whole texts are represented digitally.

Digital Logic is the basis of electronic systems, such as computers and cell phones. Digital Logic is rooted in binary code, a series of zeroes and ones each having an opposite value. This system facilitates the design of electronic circuits that convey information, including logic gates. Digital Logic gate functions include and, or and not. The value system translates input signals into specific output. Digital Logic facilitates computing, robotics and other electronic applications.

Digital Logic Design is foundational to the fields of electrical engineering and computer engineering. Digital Logic designers build complex electronic components that use both electrical and computational characteristics. These characteristics may involve power, current, logical function, protocol and user input. Digital Logic Design is used to develop hardware, such as circuit boards and microchip processors. This hardware processes user input, system protocol and other data in computers, navigational systems, cell phones or other high-tech systems.

Data Representation and Number system

Numeric systems

The numeric system we use daily is the decimal system, but this system is not convenient for machines since the information is handled codified in the shape of on or off bits; this way of codifying takes us to the necessity of knowing the positional calculation which will allow us to express a number in any base where we need it.

Radix number systems

The numeric system we use daily is the decimal system, but this system is not convenient for machines since the information is handled codified in the shape of on or off bits; this way of codifying takes us to the necessity of knowing the positional calculation which will allow us to express a number in any base where we need it.

A base of a number system or radix defines the range of values that a digit may have.

In the binary system or base 2, there can be only two values for each digit of a number, either a "0" or a "1".

In the octal system or base 8, there can be eight choices for each digit of a number:

"0", "1", "2", "3", "4", "5", "6", "7".

In the decimal system or base 10, there are ten different values for each digit of a number:

"0", "1", "2", "3", "4", "5", "6", "7", "8", "9".

In the hexadecimal system, we allow 16 values for each digit of a number:

"0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "A", "B", "C", "D", "E", and "F".

Where “A” stands for 10, “B” for 11 and so on.

Conversion among radices

- Convert from Decimal to Any Base

Let’s think about what you do to obtain each digit. As an example, let's start with a decimal number 1234 and convert it to decimal notation. To extract the last digit, you move the decimal point left by one digit, which means that you divide the given number by its base 10.

1234/10 = 123 + 4/10

The remainder of 4 is the last digit. To extract the next last digit, you again move the decimal point left by one digit and see what drops out.

123/10 = 12 + 3/10

The remainder of 3 is the next last digit. You repeat this process until there is nothing left. Then you stop. In summary, you do the following:

```
Quotient  Remainder

     -----------------------------
     1234/10 =   123        4 --------+
      123/10 =    12        3 ------+ |
       12/10 =     1        2 ----+ | |
        1/10 =     0        1 --+ | | |(Stop when the quotient is 0)
                                | | | |
                                1 2 3 4   (Base 10)
```

Now, let's try a nontrivial example. Let's express a decimal number 1341 in binary notation. Note that the desired base is 2, so we repeatedly divide the given decimal number by **2**.

```
               Quotient  Remainder
     -----------------------------
     1341/2  =   670        1 ----------------------+
      670/2  =   335        0 --------------------+ |
      335/2  =   167        1 ------------------+ | |
      167/2  =    83        1 ----------------+ | | |
       83/2  =    41        1 --------------+ | | | |
       41/2  =    20        1 ------------+ | | | | |
       20/2  =    10        0 ----------+ | | | | | |
       10/2  =     5        0 --------+ | | | | | | |
        5/2  =     2        1 ------+ | | | | | | | |
        2/2  =     1        0 ----+ | | | | | | | | |
        1/2  =     0        1 --+ | | | | | | | | | |(Stop when the
                                | | | | | | | | | | | quotient is 0)
                                1 0 1 0 0 1 1 1 1 0 1  (BIN; Base 2)
```

Let's express the same decimal number 1341 in octal notation.

```
               Quotient  Remainder
     -----------------------------
     1341/8  =   167        5 --------+
      167/8  =    20        7 ------+ |
       20/8  =     2        4 ----+ | |
        2/8  =     0        2 --+ | | |  (Stop when the quotient is 0)
                                | | | |
                                2 4 7 5  (OCT; Base 8)
```

Let's express the same decimal number 1341 in hexadecimal notation.

```
               Quotient  Remainder
     -----------------------------
     1341/16 =    83       13 ------+
       83/16 =     5        3 ----+ |
        5/16 =     0        5 --+ | |  (Stop when the quotient is 0)
                                | | |
                                5 3 D  (HEX; Base 16)
```

In conclusion, the easiest way to convert fixed point numbers to any base is to convert each part separately. We begin by separating the number into its integer and fractional part. The integer part is converted using the remainder method, by using a successive division of the number by the base until a zero is obtained. At each division, the reminder is kept and then the new number in the base r is obtained by reading the remainder from the lat remainder upwards.

The conversion of the fractional part can be obtained by successively multiplying the fraction with the base. If we iterate this process on the remaining fraction, then we will obtain successive significant digit. This methods form the basis of the multiplication methods of converting fractions between bases

Example. Convert the decimal number 3315 to hexadecimal notation. What about the hexadecimal equivalent of the decimal number 3315.3?

Solution:

```
               Quotient  Remainder
     -----------------------------
     3315/16 =   207        3 ------+
      207/16 =    12       15 ----+ |
       12/16 =     0       12 --+ | |   (Stop when the quotient is 0)
                                | | |
                                C F 3   (HEX; Base 16)


                                          (HEX; Base 16)
                Product  Integer Part    0.4 C C C ...
     ------------------------------         | | | |
     0.3*16    =   4.8         4        ----+ | | | | |
     0.8*16    =  12.8        12        ------+ | | | |
     0.8*16    =  12.8        12        --------+ | | |
     0.8*16    =  12.8        12        ----------+ | |
               :                        ---------------------+
               :
   Thus, 3315.3 (DEC) --> CF3.4CCC... (HEX)
```

- Convert From Any Base to Decimal

Let's think more carefully what a decimal number means. For example, 1234 means that there are four boxes (digits); and there are 4 one's in the right-most box (least significant digit), 3 ten's in the next box, 2 hundred's in the next box, and finally 1 thousand's in the left-most box (most significant digit). The total is 1234:

```
     Original Number:       1      2      3     4
                            |      |      |     |
     How Many Tokens:       1      2      3     4
     Digit/Token Value: 1000    100     10     1
     Value:             1000 + 200   + 30   + 4   = 1234
```

```
or simply,  1*1000 + 2*100 + 3*10 + 4*1 = 1234
```

Thus, each digit has a value: $10^{\wedge 0}=1$ for the least significant digit, increasing to $10^{\wedge 1}=10$, $10^{\wedge 2}=100$, $10^{\wedge 3}=1000$, and so forth.

Likewise, the least significant digit in a hexadecimal number has a value of $16^{\wedge 0}=1$ for the least significant digit, increasing to $16^{\wedge 1}=16$ for the next digit, $16^{\wedge 2}=256$ for the next, $16^{\wedge 3}=4096$ for the next, and so forth. Thus, 1234 means that there are four boxes (digits); and there are 4 one's in the right-most box (least significant digit), 3 sixteen's in the next box, 2 256's in the next, and 1 4096's in the left-most box (most significant digit). The total is:

$1*4096 + 2*256 + 3*16 + 4*1 = 4660$

In summary, the conversion from any base to base 10 can be obtained from the formulae

$x_{10} = \sum_{i=-m}^{n-1} d_i b^i$ Where b is the base, d_i the digit at position i, m the number of digit after the decimal point, n the number of digits of the integer part and X_{10} is the obtained number in decimal. This form the basic of the polynomial method of converting numbers from any base to decimal

Example. Convert 234.14 expressed in an octal notation to decimal.

$2*8^2 + 3*8^1 + 4*8^0+1*8^{-1} + 4*8^{-2} \quad = 2*64 +3*8 +4*1 +1/8 +4/64 =156.1875$

Example. Convert the hexadecimal number 4B3 to decimal notation. What about the decimal equivalent of the hexadecimal number 4B3.3?

Solution:

```
Original Number:      4    B    3  .  3
                      |    |    |     |
How Many Tokens:      4   11    3     3
Digit/Token Value:  256   16    1     0.0625
Value:             1024 +176  + 3   + 0.1875  = 1203.1875
```

Example. Convert 234.14 expressed in an octal notation to decimal.

Solution:

```
Original Number:      2    3    4  .  1       4
                      |    |    |     |       |
How Many Tokens:      2    3    4     1       4
Digit/Token Value:   64    8    1     0.125   0.015625
Value:              128 + 24  + 4   + 0.125 + 0.0625    = 156.1875
```

- Relationship between Binary - Octal and Binary-hexadecimal

As demonstrated by the table bellow, there is a direct correspondence between the binary system and the octal system, with three binary digits corresponding to one octal digit. Likewise, four binary digits translate directly into one hexadecimal digit.

BIN	OCT	HEX	DEC
0000	00	0	0
0001	01	1	1
0010	02	2	2
0011	03	3	3
0100	04	4	4
0101	05	5	5
0110	06	6	6
0111	07	7	7
1000	10	8	8
1001	11	9	9
1010	12	A	10
1011	13	B	11
1100	14	C	12
1101	15	D	13
1110	16	E	14
1111	17	F	15

With such relationship, In order to convert a binary number to octal, we partition the base 2 number into groups of three starting from the radix point, and pad the outermost groups with 0's as needed to form triples. Then, we convert each triple to the octal equivalent.

For conversion from base 2 to base 16, we use groups of four.

Consider converting 10110_2 to base 8:

$10110_2 = 010_2\ \ 110_2 = 2_8\ \ 6_8 = 26_8$

Notice that the leftmost two bits are padded with a 0 on the left in order to create a full triplet.