Concepts of Conversion of Base 10 Number to Base 2 Fixed Point Register Binary Number Representation

Sean Rodby, Luke Snyder, Autar Kaw
University of South Florida
United States of America
kaw@eng.usf.edu
Website: http://numericalmethods.eng.usf.edu
Version: Mathcad 14; Last Revised: August 28, 2008

Introduction

The following worksheet illustrates how to convert a positive decimal number to a fixed point register binary number using loops and various conditional statements. The user inputs a decimal number in the *Input* section of the program, and then an equivalent binary number is given as an output.

Section 1: Input

This is the only section where the user interacts with the program.

- Enter positive real number to be converted to binary number

  decnum := 54.3

- Enter number of bits to be used in *integer part*

  m := 8

- Enter number of bits to be used in *fractional part*

  n := 6

Section 2: Procedure

Calculating the minimum and maximum possible base-10 value based on the number of bits specified by the user.

$$minval := 2^{-n}$$

$$maxval := \left| \begin{array}{l} sum1 \leftarrow 0 \\ sum2 \leftarrow 0 \\ \text{for } i \in 0..m-1 \\ \quad sum1 \leftarrow sum1 + 2^{i} \\ \text{for } i \in 1..n \\ \quad sum2 \leftarrow sum2 + 2^{-i} \\ sum1 + sum2 \end{array} \right.$$

The magnitude of the minimum number that can be represented in floating point representation given the number of bits is

$$minval = 0.016$$

The magnitude of the maximum number that can be represented in floating point representation given the number of bits is

$$maxval = 255.984$$

If the maximum value that can be produced given the user specified number of bits is smaller than the number to be represented, then more bits are needed. Similarly, if the minimum value that can be produced given the user specified number of bits is larger than the number to be represented, a change in bits is needed.

Using the floor command to isolate each section of the decimal number to begin binary number conversion

before_decimal := floor(decnum)

after_decimal := decnum − floor(decnum)

Using a loop to perform the binary conversion repetitively until the necessary number of bits are used. Using the *floor* command to calculate the quotient of the decimal number. Using a combination of the *ceil* and *floor* commands to determine the remainder, and inputting this value into an array. Then calculating the binary conversion of the value before the decimal. Here we use the variable *Bold* to denote the value of the integer part of the decimal number.

$$
\begin{array}{l}
\text{Binary} := \left| \begin{array}{l}
\text{Bold} \leftarrow \text{before\_decimal} \\
\text{for } i \in 0 .. m - 1 \\
\quad \left| \begin{array}{l}
\text{Quotient} \leftarrow \text{floor}\left(\frac{\text{Bold}}{2}\right) \\
\text{Binary}_i \leftarrow \text{ceil}\left(\frac{\text{Bold}}{2}\right) - \text{floor}\left(\frac{\text{Bold}}{2}\right) \\
\text{if Quotient} = 0 \wedge \text{before\_decimal} \neq 0 \\
\quad \left| \begin{array}{l} \text{Binary}_i = 1 \\ \text{break} \end{array} \right. \\
\text{if before\_decimal} = 0 \\
\quad \left| \begin{array}{l} \text{Binary}_i = 0 \\ \text{break} \end{array} \right. \\
\text{Bold} \leftarrow \text{Quotient}
\end{array} \right. \\
\text{Binary}
\end{array} \right.
\end{array}
$$

Using this method, the placement of the binary numbers in the array *Binary* are backwards and so must be flipped around to form the actual binary number.

$$t := \text{length}(\text{Binary})$$

$$
\begin{array}{l}
\text{Binary\_Before\_Dec} := \left| \begin{array}{l}
\text{for } i \in 0 .. t - 1 \\
\quad \text{Binary\_Before\_Dec}_i \leftarrow \text{Binary}_{(t-i-1)} \\
\text{Binary\_Before\_Dec}
\end{array} \right.
\end{array}
$$

Converting the *fractional part* of the decimal into a binary number. Here we initialize the sum using a new value, *BoldA,* which represents the value after the decimal point.

```
Binary_After_Dec :=
    BoldA ← after_decimal
    for i ∈ 0 .. n − 1
        Num ← BoldA·2
        Binary_After_Dec_i ← 0  if Num < 1
        if Num ≥ 1
            Num ← (Num − floor(Num))
            Binary_After_Dec_i ← 1
        BoldA ← Num
    Binary_After_Dec
```

Combining the binary numbers and determining the number of terms in the resulting base 2 number. Then displaying the results.

BBDLen := length(Binary_Before_Dec)

BADLen := length(Binary_After_Dec)

```
str1 :=
    str1 ← ""
    for i ∈ 0 .. BBDLen − 1
        str1 ← concat(str1, num2str(Binary_Before_Dec_i))
    str1
```

```
str2 :=
    str2 ← ""
    for i ∈ 0 .. BADLen − 1
        str2 ← concat(str2, num2str(Binary_After_Dec_i))
    str2
```

Final_Bin_Num := concat(str1, "." , str2) = "110110.010011"

Using a loop to sum values of the *integer* portion of the base-2 number. The loop variable *sumint* is used for summation and is initialized at 0.

$$\text{int\_bin} := \text{floor}(\text{str2num}(\text{Final\_Bin\_Num}))$$

$$\text{str\_int} := \text{num2str}(\text{int\_bin})$$

$$m := \text{strlen}(\text{str\_int})$$

$$n := \text{strlen}(\text{Final\_Bin\_Num})$$

$$\begin{aligned}&\text{sumint} := \left|\begin{array}{l}\text{sumint} \leftarrow 0\\ \text{for } i \in 0..m-1\\ \quad \left|\begin{array}{l}\text{bin\_int}_i \leftarrow \text{str2num}(\text{substr}(\text{str\_int}, i, 1))\\ \text{sumint} \leftarrow \text{sumint} + \text{bin\_int}_i \cdot 2^{m-i-1}\end{array}\right.\\ \text{sumint}\end{array}\right.\end{aligned}$$

Using a loop to sum values of the *fractional* portion of the base-2 number. The loop variable *sumfrac* is used for summation and is initialized at 0.

$$\begin{aligned}&\text{sumfrac} := \left|\begin{array}{l}\text{sumfrac} \leftarrow 0\\ j \leftarrow 1\\ \text{for } i \in m+1..n-1\\ \quad \left|\begin{array}{l}\text{bin\_frac}_j \leftarrow \text{str2num}(\text{substr}(\text{Final\_Bin\_Num}, i, 1))\\ \text{sumfrac} \leftarrow \text{sumfrac} + \text{bin\_frac}_j \cdot 2^{-j}\\ j \leftarrow j + 1\end{array}\right.\\ \text{sumfrac}\end{array}\right.\end{aligned}$$

Adding the *fractional* portion of the base-2 number with the *integer* portion which yields the base-10 number.

$$\text{total\_dec} := \text{sumint} + \text{sumfrac} = 54.297$$

Then finding the relative percentage difference between the representation and the num to be represented.

$$\text{tot\_perc\_diff} := \frac{\text{decnum} - \text{total\_dec}}{\text{decnum}} \cdot 100 = 5.755 \times 10^{-3}$$

Conclusion

This worksheet illustrates the use of Mathcad to convert a base-10 number to a base-2 binary number. It is important to understand the binary system as it has numerous applications. Critical to this understanding is being able to convert binary numbers to base-10 numbers, and vice-versa.

References

Binary Representation of Numbers
See:
http://numericalmethods.eng.usf.edu/mcd/gen/01aae/mcd_gen_aae_txt_binaryrepresentation.pdf