Concepts of Conversion of Base 10 Number to Base 2 Floating Point Binary 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 base-10 real number to a base-2 floating point binary representation using loops and various conditional statements. The user inputs a base-10 number in any format, the total number of bits, and the number of bits for the mantissa in the *Input* section of the program. The program will then convert the base-10 number into a floating point binary representation by first converting the given number into a number of the format 1.xxx * 2^m where *m* represents the exponent of the number.

Section 1: Input

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

The generalized formula for floating point format in base-10 is given by

$y = \sigma \cdot m \cdot 10^e$

- Enter number to be converted to a floating point binary number

  dec_num := 5.8946

- Enter the total number of bits to be used (1st bit will be used for the sign of the number, 2nd bit will be used for the sign of the exponent.)

  tot_bits := 9

- Enter number of bits for mantissa (*m*)

  mant_bits := 4

Section 2: Procedure

The number of bits to be used for the exponent will be the total number of bits minus the number of bits used for the mantissa, minus the bits used for the sign of the decimal number, and minus the bit used for the sign of the exponent.

$$exp\_bits := tot\_bits - mant\_bits - 1 - 1$$

To find the sign of the decimal number (sigma), and the sign of the exponent, we test to see if the number is negative and if it is less than 1.

$$\sigma := \left| \begin{array}{ll} \text{"0"} & \text{if } dec\_num > 0 \\ \text{"1"} & \text{otherwise} \end{array} \right.$$

$$exp\_sign := \left| \begin{array}{ll} \text{"1"} & \text{if } \left|dec\_num\right| < 1 \wedge \left|dec\_num\right| > 0 \\ \text{"0"} & \text{otherwise} \end{array} \right.$$

Calculating the maximum possible value given the number of bits as specified by the user.

Calculating the maximum value for the exponent.

$$exp\_sum := \left| \begin{array}{l} exp\_sum \leftarrow 0 \\ \text{for } i \in 0 .. exp\_bits - 1 \\ \quad exp\_sum \leftarrow exp\_sum + 2^{i} \\ exp\_sum \end{array} \right.$$

Calculating the maximum value for the mantissa

$$mant\_sum := \left| \begin{array}{l} mant\_sum \leftarrow 1 \\ \text{for } i \in 1 .. mant\_bits \\ \quad mant\_sum \leftarrow mant\_sum + 2^{-i} \\ mant\_sum \end{array} \right.$$

If the absolute value of *dec_num* is not within the range of maximum and minimum values that can be represented in floating point binary format, then the number of bits specified is not sufficient. If this is the case either specify fewer mantissa bits, or more total bits for the worksheet.

$$\text{dec\_num} := \left|\text{dec\_num}\right| = 5.895$$

$$\text{maxval} := \text{mant\_sum}\cdot 2^{\text{exp\_sum}} = 248$$

$$\text{minval} := \text{mant\_sum}\cdot 2^{-\text{exp\_sum}} = 0.015$$

We will convert the given variable *dec_num* into some number 1.xxx * 2^{m} , where *m* is the value for the exponent. First, we will find this value *m* for the exponent using the properties of logarithms. Since the exponent value is now known, it is now possible to find the fractional portion of the decimal number (1.xxx). To isolate the fractional portion, we simply subtract one from the *mant_val* variable.

$$\text{exp\_val} := \text{floor}\left(\frac{\log(\text{dec\_num})}{\log(2)}\right)$$

$$\text{mant\_val} := \frac{\text{dec\_num}}{2^{\text{exp\_val}}}$$

$$\text{mant\_frac} := \text{mant\_val} - 1$$

Using loops to approximate the mantissa up to the specified number of mantissa bits.

```
Mantissa := | for i ∈ 0 .. mant_bits − 1
            |   | new_mant_frac ← mant_frac·2
            |   | Mant_i ← floor(new_mant_frac)
            |   | mant_frac ← new_mant_frac − Mant_i
            |   | Mantissa_i ← num2str(Mant_i)
            | Mantissa
```

```
Mantstr := | Mantstr ← ""
           | for i ∈ 0 .. mant_bits − 1
           |   Mantstr ← concat(Mantstr, Mantissa_i, "|")
           | Mantstr
```

Approximating the exponent value based on the calculated value of *m*. Then approximating the floating point values for the exponent value *m*.

```
Bin_exp := | exp_val ← |exp_val|
           | for i ∈ 0 .. exp_bits − 1
           |   | new_exp_val ← floor(exp_val / 2)
           |   | Bin_exp_i ← ceil(exp_val / 2) − new_exp_val
           |   | exp_val ← new_exp_val
           | Bin_exp
```

Using this method, the floating point values approximated for the exponent are in reverse order. This section of the program reverses this and creates a character array with the existing values as well.

$t := \mathrm{length}(Bin\_exp)$

$$
Expstr := \left|\begin{array}{l} Expstr \leftarrow \text{""} \\ \text{for } i \in 0..t-1 \\ \quad \left|\begin{array}{l} Binary\_exp_i \leftarrow \mathrm{num2str}\left(Bin\_exp_{t-i-1}\right) \\ Expstr \leftarrow \mathrm{concat}\left(Expstr, Binary\_exp_i, \text{"|"}\right) \end{array}\right. \\ Expstr \end{array}\right.
$$

Concatenating all previously calculated binary components of the floating point binary number.

$$
Mantissa\_comb := \left|\begin{array}{l} Mantissa\_comb \leftarrow \text{""} \\ \text{for } i \in 0..mant\_bits-1 \\ \quad Mantissa\_comb \leftarrow \mathrm{concat}\left(Mantissa\_comb, Mantissa_i\right) \\ Mantissa\_comb \end{array}\right.
$$

$$
Bin\_exp\_comb := \left|\begin{array}{l} Bin\_exp\_comb \leftarrow \text{""} \\ \text{for } i \in 0..t-1 \\ \quad \left|\begin{array}{l} Binary\_exp_i \leftarrow \mathrm{num2str}\left(Bin\_exp_{t-i-1}\right) \\ Bin\_exp\_comb \leftarrow \mathrm{concat}\left(Bin\_exp\_comb, Binary\_exp_i\right) \end{array}\right. \\ Bin\_exp\_comb \end{array}\right.
$$

$Final\_Form := \mathrm{concat}(\text{"|"}, \sigma, \text{"|"}, \text{" |"}, exp\_sign, \text{"| |"}, Mantstr, \text{" |"}, Expstr)$

$Final\_Form = \text{"|0| |0| |0|1|1|1| |0|1|0|"}$

Conclusion

This worksheet illustrates the use of Mathcad to convert a base-10 number to a floating point binary representation. Recall that floating point representation is used more often than fixed point representation due to two primary advantages: floating point representation supports a much larger range of values while maintaining a relative error of similar magnitude for all numbers.

References

Floating Point Representation.
See:
http://numericalmethods.eng.usf.edu/mcd/gen/01aae/mcd_gen_aae_txt_floatingpoint.pdf