Reconfigurable Computing

Precision

“It is the mark of an educated mind to rest satisfied with the degree of precision which the nature of the subject admits and not to seek exactness where only an approximation is possible.”

– Aristotle

Permission to use figures have been gained where possible. Please contact me if you believe anything within infringes on copyright.

Philip Leong (philip.leong@sydney.edu.au)
School of Electrical and Information Engineering
http://www.ee.usyd.edu.au/~phwl
Overview

› Number systems
  - Unsigned
  - Two’s complement
  - Two’s complement fractions
  - Floating Point
  - Logarithmic number system

› Case study
  - CORDIC algorithm
Number Systems
Unsigned integers are used to represent the nonnegative integers. An $N$-bit unsigned integer has a range $[0, 2^N - 1]$ and can be described in binary form, with $u_i$ being the $i$’th binary digit:

$$U = (u_{N-1}u_{N-2}\ldots0), \ u_i \in \{0, 1\}.$$ 

This represents the number

$$U = \sum_{i=0}^{N-1} u_i 2^i.$$
\[ X = (x_{N-1} x_{N-2} \ldots 0), \quad x_i \in \{0, 1\}. \]

\( X \) has a range of \([ -2^{N-1}, 2^{N-1} - 1 ]\) and represents

\[ X = -x_{N-1} 2^{N-1} + \sum_{i=0}^{N-2} x_i 2^i \]
The most significant $N - F$ bits of the number represent the integer part and the remaining $F$ bits are the fractional part of the number

$$Y = (a_{N-1} \ldots a_F a_{F-1} \ldots a_0).$$

This corresponds to a scaling of the two’s complement integer representation by the factor $S = 2^{-F}$ and the two’s complement fraction number $Y$ represents

$$Y = 2^{-F} \times (-x_{N-1}2^{N-1} + \sum_{i=0}^{N-2} x_i 2^i)$$

Note that the two’s complement fraction $(N, 0)_I$ corresponds to the two’s complement integer case and $(N, N)_I$ has a range of $[-1, 1)$. 
If we wish to perform arithmetic on two (N,F) format 2’s complement fractions

Addition and subtraction
- Normal addition

Multiplication
- An (N,F) multiplication gives a (2N, 2F) result so you need to do an arithmetic right shift by F bits from the 2N multiplier output
- E.g. for (4,3) 0.75*0.75 = 0.110*0.110=00.100100 >> 3=0.100=0.5

Question: how about division?
Number systems – Floating Point

\[ Z = (A \underbrace{b_{J-1} \ldots b_0}_{B} C_{F-1} \ldots b_0) \]

A represents the sign \( S \) where

\[
S = \begin{cases} 
+1 & \text{if } a_0 = 0 \\
-1 & \text{if } a_0 = 1 
\end{cases}
\]

The unsigned integers \( B \) and \( C \) are encoded representations of the exponent and mantissa respectively. The exponent \( E \), is stored in a biased representation with \( E = B - (2^{J-1} - 1) \). For normalized numbers, \( B \neq 0 \) and the significand is represented by \( M = 1 + C \times 2^{-F} \). This is a two’s complement fraction \((F + 1, F)\) with the most significant bit being implicitly set to 1. If \( B = 0 \), it is called a denormalized number, and there is no implicit 1 in the \((F, F)\) fraction.
Number systems – Floating Point

\[
Z = \begin{cases} 
S \times 2^E \times M & \text{if } (0 < B < 2^J - 1) \\
S \times 2^E \times (M - 1) & \text{if } (B = 0) \\
S \times \infty & \text{if } (B = 2^J - 1 \text{ and } C = 0) \\
NaN & \text{if } B = 2^J - 1 \text{ and } C \neq 0.
\end{cases}
\]
The logarithmic number system (LNS) is a special case of floating point in which the mantissa is always 1 (i.e. only the sign and exponent fields are used). It has the advantages of simplified implementation at the expense of reduced precision. For an $N$ bit LNS number, $(N, F)_L$, the most significant bit is a zero flag, $Z$. $Z$ is zero if the number is zero (since there is no log of zero), otherwise set. The next most significant bit is used for a sign bit and the rest of the number is the base 2 logarithm of the magnitude of the number to be represented in $(N-2, F)_T$ two’s complement fraction format. If $E$ is the value of this two’s complement fraction and $S$ is defined as for floating point, then

$$L = \begin{cases} 0 & \text{if } Z = 0 \\ L = S \times 2^E & \text{if } Z = 1 \end{cases}$$