

Real Number Representation for Image Processing on FPGAs

A.S.L. Bainbridge-Smith

Dept. Electrical & Computer Engineering.

University of Canterbury, Private Bag 4800, Christchurch, New Zealand

Email: Andrew.Bainbridge-Smith@canterbury.ac.nz

Abstract

A general framework for representing real numbers on a binary digital system is described. The flexibility of the framework allows different encoding schemes to be used depending upon knowledge of the probability density function (pdf) of the amplitudes of the signals. A signal to quantisation noise analysis shows that where the pdf of signal amplitudes is uniform a fixed point representation is most appropriate. However, other pdfs will lead to alternatives, for which floating point need not be optimal.

Keywords: Fixed and floating point representation, SQNR, pdf

1 Introduction

Real numbers can be represented on binary digital systems in a large number of different ways. The choice made on how best to represent these numbers will depend upon the statistical properties of the numbers and the uses to which they will be put, i.e. what processing will take place. As there are only a finite number of bits available any choice made is will lead to some quantisation error, and the best representation will be the one that minimises this error. However, other factors influence the choice that will be made, which include the hardware platform and the programming work required to implement and use the representation.

In practice many of us simply use 64 or 32 bit IEEE Standard 754 floating point numbers (double and single precision respectively) on a PC hardware system. This choice being made because the hardware is available on our desktops and the suites of software, e.g. MATLAB, are available using this representation. Fixed point representation is also readily available, but not usually adopted because either little thought is placed on the choice, or the extra work required to ensure overflow and underflow problems in calculations do not occur are deemed too difficult to solve. Interestingly a fixed point based representation often may give better overall noise performance. Nevertheless the floating point decision is made because of the ease and prototyping nature of the work being undertaken.

When considering a production system the competent designer will consider number representation more carefully. If implementation is to be made on

a microprocessor or DSP system the choice will be highly constrained by the bit width of the hardware platform and its internal number representation, i.e. fixed or floating point. These days floating point invariably means IEEE Standard 754 [5, 6]. Some analysis beforehand, in terms of required bit width and implementation difficulty (fixed v floating) will help drive this choice.

However, increasingly a number of signal and image processing algorithms are being implemented on field programmable gate arrays (FPGAs). Their increase in popularity for computing purposes is due in part to the massive degree of parallelism they can implement and their flexibility to represent custom arithmetic and logic devices. They also afford greater flexibility in the bit length of numbers, indeed bit length need not be constant through out the entire design, and number representation need not be restricted to fixed or IEEE floating point form.

This paper presents a general framework for real number representation in binary systems. The work is based on the idea by Ewe et. al. [2] and is intended for applications being implemented on custom logic or FPGAs where great savings in silicon area or resources can be achieved by not using floating point. This work is in its self interesting, showing a review of number representation, Section 2, and demonstrating the importance of knowing the probability density function (pdf) of the numbers being processed via the examination of theoretical signal to quantisation noise ratio (SQNR) performance in Section 4. What this paper lacks, however, are concrete examples to demonstrably show the utility of this work. This problem comes

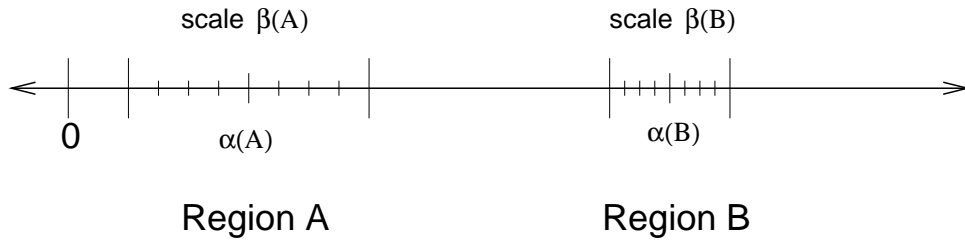


Figure 1: Encoded regions of the real number line.

about because often we do not know the probability density function (pdf) of the signals we are processing. This is precisely the motivation for this paper. Firstly, to show that where the pdf of the numbers being processed are unknown then fixed point number representation should be chosen for best SQNR. And secondly, to consider the implications when the pdf of amplitudes of images and signals are known.

2 Generalised number representation

Numerous binary real number systems have been proposed including: fixed point, floating point, A-law and μ -law representations [6]. This work groups all of these real number representation systems into a common framework.

We start by stating our requirement to represent or encode each number value in a b bit format. In the simplest form we can simply divide the real number line between two finite limits, say $-A$ to A , into 2^b equal sized steps, i.e. a linear scale. This is the classic fixed-point representation and aside from being simple it also leads to exceedingly fast and small foot print (silicon area) digital arithmetic circuitry. Another consequence of this choice is that the circuitry will be of low power-drain and relatively cheap. Thus fixed-point processing is the preferred format for most signal processing applications.

The generalised framework for number representation involves splitting the b available bits into two subsets of p precision bits and s scale bits in length, such that the total number of encoding bits is,

$$b = p + s. \quad (1)$$

The first subset of p bits ($p < b$) is used to represent a linear fixed step size within a particular region of the real number line. The second subset of s bits is used to encode different regions of the number line. Using the number line shown in Figure 1 as a guide, we can therefore write a particular real value v as,

$$v = {}_pM \times \beta({}_sR) + \alpha({}_sR), \quad (2)$$

where ${}_pM$ encodes the step number within a region ${}_sR$ of the number line such that $\max\{{}_pM\} = 1.0$, $\beta({}_sR)$ is the scaling function of that region, and $\alpha({}_sR)$ is a shifting function of the region from the origin of the number line. It should be noted that no restriction is placed on the form of either the scaling or shifting functions, i.e. they could be highly nonlinear. The epilogue subscript is used to indicate the number of bits available to each component.

With this general representative framework we see that for a fixed point real number $\beta({}_sR)$ is a constant scale factor equal to the largest real value that must be represented and $\alpha({}_sR)$ is equal to any dc offset. As both $\beta({}_sR)$ and $\alpha({}_sR)$ are constant values it is unnecessary to reserve any bits for s , hence $p = b$.

3 Floating point format

The generalised framework of Eq.(2) is useful as it encompasses all practical number representation schemes. For example the ITU-T standard G.711 companding PCM signals (both A and μ -laws) uses a piecewise linear approximation to model the logarithmic compression applied to the signal [4]. In this case, $\beta({}_sR) \neq 0$, $\alpha({}_sR) \neq 0$ and $s \neq 0$. However, performing arithmetic directly using G.711 form numbers is complex and effectively requires a conversion to linear PCM first. Hence G.711 is only suitable when the signal does not require any further processing and we wish to reduce signal bandwidth.

What makes arithmetic particular difficult is where the ${}_sR$ regions do not overlap and that $\alpha({}_sR) \neq 0$. Consequently, where we wish to represent real numbers with very large limits, $A \rightarrow \infty$, and to very fine scales of precision, but not both simultaneously, then we usually employ a floating point representation in the form of,

$$v = {}_pM \times \beta_0^{{}_sR}, \quad (3)$$

i.e. $\beta({}_sR) = \beta_0^{{}_sR}$ and where ${}_pM$ and ${}_sR$ are fixed point real numbers, and β_0 is a constant base. ${}_pM$ is called the mantissa or significant of the number

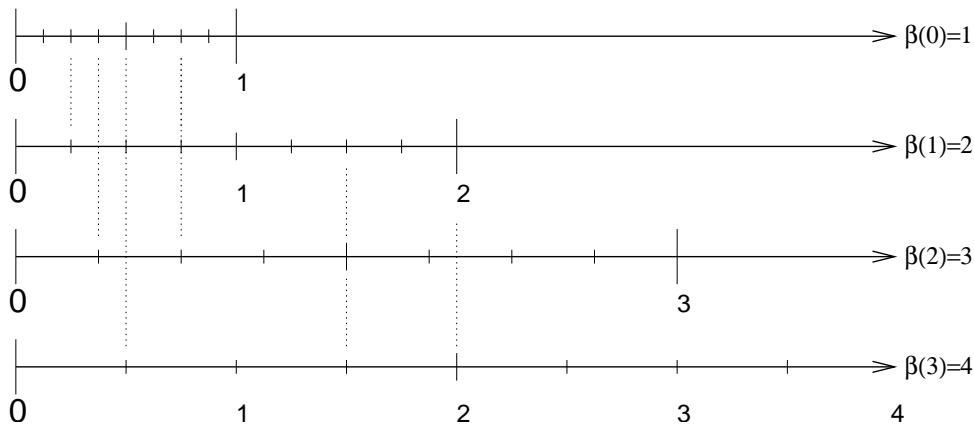


Figure 2: Simple linear encoded overlapping regions of the real number line.

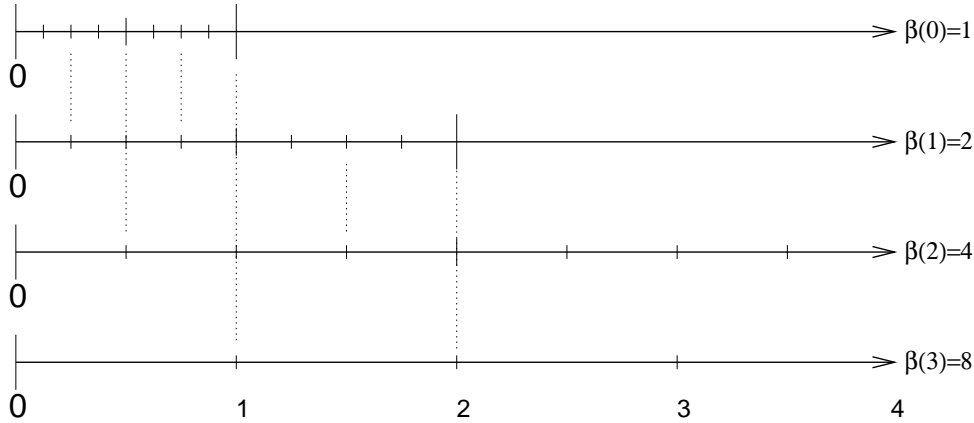


Figure 3: Integer exponent encoded overlapping regions of the real number line.

and ${}_sR$ the exponent. If ${}_sR$ is represented as integers then β_0 is typically chosen as an integer value so that at least some numbers within the different scale regions of ${}_sR$ overlap.

Floating point representation is a compromise in complexity of circuit design between fixed point and exponential representation when extensive arithmetic computations on the signals are required. Multiplications require fixed point multipliers for the mantissa and simple adders for the exponent. When a choice of $\beta_0 = 2$ is made then additions of numbers with different exponents can be made with barrel shifters and simple adders. Nevertheless full floating point units require considerable more resources to implement and power to operate than fixed point numbers [7, 3].

It is important to note that while $\beta({}_sR)$ could take any form the exponential form is the best choice. Consider the simple linear representation of $\beta({}_2R) = \{1, 2, 3, 4\}$, for illustrative purposes we also set $p = 4$ and show in Figure 2 the number lines for each scale. It is instructive to see that when adding numbers from different scales to the scale $\beta(R) = 1$ a simple barrel shift, addition and

truncation are all that are required, with a constant loss of precision with any addition. However, when adding numbers from different scales, without this restriction being applied, we see that due to a lack of a common denominator a simple shift of the finer scale representation to the coarser cannot take place. This leads to a much more complex addition circuit with a nonlinear degradation of signal quality for signals of different values for the two chosen scales.

A simple example for $\beta({}_2R) = \{2^0, 2^1, 2^2, 2^3\}$ is shown in Figure 3. It is clear from this example that conversion from one scale to another is straightforward with a linear degradation in quality due to the common step size factor.

IEEE Standard 754 for floating point numbers are defined for $b = 32$ and $b = 64$, utilise a base power of 2 and linearly distribute the exponent over the range of values available for s bits; an approach taken by almost all floating point representations. However there is no need for the distribution of exponents to be linear. In the following section we quantify the signal quality by examining the signal to quantising noise ratio (SQNR).

4 SQNR performance of floating point representation

Consider an arbitrary signal $x(t)$, then its signal power is,

$$S\{x(t)\} = S_x = kA^2, \quad (4)$$

where k is a scaling constant and is equal to $\frac{1}{2}$ when the signal is a sinusoid.

If the signal is quantised using b bits over the full $-A$ to A range of the signal then there will be a constant step size of,

$$\Delta = \frac{A}{2^{b-1}}. \quad (5)$$

Furthermore, if the signal amplitudes have a uniform pdf then the power of the quantisation error will be [1],

$$N\{x(t)\} = N_x = \frac{\Delta^2}{12}, \quad (6)$$

and the overall SQNR in decibals can then be written as,

$$\begin{aligned} \text{SQNR} &= 10 \log_{10} \frac{kA^2}{\Delta^2/12} \\ &\approx C + 6.02(b-1) \text{ dB}. \end{aligned} \quad (7)$$

This is the traditionally quoted SQNR performance for fixed point representation and is fundamentally based on the assumption of a uniform pdf of signal amplitude.

Now let us examine the theoretical performance of a two range encoded floating point number. This representation can also be called dual fixed-point [2]. Take the same signal $x(t)$ which is now to be quantised over two ranges R_1 and R_2 ($s=1$) with p bits in each range, such that $b=p+1$. The ranges are overlapping and signal amplitudes within the range of R_1 will be quantised within that range, otherwise they will be quantised in range R_2 . The signal power is still the same as given in Eq.(4).

The quantisation step sizes for each range R_1 and R_2 respectively are,

$$\Delta_1 = \frac{R_1}{2^{p-1}}, \quad (8)$$

and

$$\Delta_2 = \frac{R_2}{2^{p-1}} = \frac{A}{2^{p-1}} = n \frac{R_1}{2^{p-1}}. \quad (9)$$

Note the requirement that the two step sizes must have an integer ratio n . Thus the SQNR for our two region quantisation scheme is,

$$\begin{aligned} \text{SQNR} &= 10 \log_{10} \frac{kA^2}{q_1 \Delta_1^2/12 + q_2 \Delta_2^2/12} \\ &= C + 10 \log_{10} \frac{A^2}{q_1 \Delta_1^2 + q_2 \Delta_2^2} \text{ dB}, \end{aligned} \quad (10)$$

where q_1 and q_2 are the probabilities that the input signals are within the regions R_1 or R_2 respectively. Note that while region R_2 overlaps with region R_1 (in terms of its scale) when computing p_2 it is considered disjoint, i.e.,

$$q_1 = P(R_1) = \int_0^{R_1} p(x) dx, \quad (11)$$

and

$$q_2 = P(R_2) - P(R_1) = \int_{R_1}^{R_2} p(x) dx, \quad (12)$$

where $P(x)$ is the probability distribution function (PDF) and $p(x)$ is the probability density function (pdf).

Before attempting to simplify Eq.(10) we should first note that signal amplitudes must lie completely within either regions R_1 and R_2 and thus $P(R_2) = 1$. Also that $R_1 = A/n$. Therefore Eq.(10) becomes,

$$\begin{aligned} \text{SQNR} &= C + 10 \log_{10} \frac{A^2}{\Delta_1^2} \frac{1}{q_1 + n^2 q_2} \\ &\approx C + 6.02(p-1) \\ &\quad - 10 \log_{10} (1 + (\frac{1}{n^2} - 1)P(R_1)) \text{ dB}. \end{aligned} \quad (13)$$

We can find the optimal partition and thus step size factor n by,

$$\max\{\text{SQNR}\}_n = \frac{d\text{SQNR}}{dn} = 0, \quad (14)$$

and

$$\begin{aligned} \frac{d\text{SQNR}}{dn} &= -\frac{10}{\ln 10} \frac{1}{1 + P(R_1)((1/n)^2 - 1)} \times \\ &\quad (P'(R_1)((1/n)^2 - 1) - \frac{2}{n^3} P(R_1)). \end{aligned} \quad (15)$$

Consider the uniform distribution for $x(t)$ – the only reasonable distribution we can consider without any additional information,

$$\begin{aligned} \therefore P(R_1) &= \int_0^{R_1} \frac{1}{A} dx \\ &= 1/n \\ \therefore \frac{dP(R_1)}{dR_1} &= -1/n^2. \end{aligned} \quad (16)$$

For which the optimal choice for n is,

$$n = \lceil \sqrt{3} \rceil = 2, \quad (17)$$

with an SQNR of $6.02(p-1) + 2.04$. This SQNR figure is less than what would have been achieved if a single scale had been used and the extra scale bit been used in the precision coding. This is not surprising given the assumption of a uniform pdf. Quite different results could have been achieved if a different pdf had been assumed.

5 Conclusions

We can easily extend the analysis to n different regions. If n is chosen to be equal to the number of bits in the exponent of IEEE 754 floating point and the scale regions linearly increase in powers of 2 then we can demonstrate the SQNR performance of IEEE 754 floating point representation. Moreover if the pdf of the signal amplitudes is assumed to be uniform, which is the only reasonable choice without aprior knowledge, then we will find that the SQNR will be less than if a fixed point representation with an equal number of bits had been adopted. However, where the pdf is not uniform, perhaps being normal, skewed or multimodal then we could well find the floating point representation to give a superior SQNR over fixed point. Such non-uniform distributions may also arise after processing.

Furthermore with careful design it may be possible to show that a coding scheme using fewer and arbitrary scaled regions could provide an equal or better performing SQNR than either fixed or IEEE 754 floating point. Such a design would be preferable because it would use less bits. Also, such a design could be readily implemented on an FPGA using simple look-up-tables and most importantly would use less resources and power.

In conclusion, a general number representation framework has been demonstrated. Also shown is that the exponential scaling form used in floating point representation is the most optimal in hardware efficiency. An approach has been shown for quantifying the quality in terms of SQNR of each form provided the underlying statistical model of the pdf of the signal amplitudes, or their amplitudes after processing is known. To further advance this requires consideration of the amplitude pdf for particular classes of images.

6 Acknowledgements

This work stems from my time at Imperial College London and follows directly on from the research work of PhD student C.T. Ewe, Professor P.Y.K. Cheung and Dr G.A. Constantinides.

References

- [1] H. Baher. *Analog and Digital Signal Processing*. Wiley, 1990.
- [2] C. T. Ewe, P. Y. K. Cheung, and G. A. Constantinides. Dual fixed-point: An efficient alternative to floating-point computation. In *Field Programmable Logic and Applications*, volume Lecture Notes in Computer Science. Springer-Verlag, 2004.
- [3] Gene Frantz and Ray Simar. *TI Application Note: Comparing Fixed- and Floating-Potin DSPs*. Texas Instruments, 2004.
- [4] Simon Haykin. *Analog and Digital Communications*. Wiley, 1989.
- [5] IEEE. *IEEE 754 Standard for Binary Floating-Point Arithmetic*. IEEE, 1984.
- [6] Israel Koren. *Computer Arithmetic Algorithms*. A K Peters, 2002.
- [7] Jian Liang, Russell Tessier, and Oskar Mencer. Floating point generation and evaluation for fpgas. In *11th Annual IEEE Symposium on Field Programmable Custom Computing Machines*. IEEE, 2003.