NUMBER REPRESENTATIONS FOR REDUCING SWITCHED CAPACITANCE IN SUBBAND CODING

John R. Sacha and Mary Jane Irwin

Computer Science and Engineering Department
The Pennsylvania State University
University Park, PA 16802
Phone: (814) 863-4162
E-mail: sacha@cse.psu.edu

ABSTRACT

In low power VLSI design, fixed point number representations are standard. For some signal processing applications, however, achieving sufficient dynamic range with fixed point may lead to computations utilizing more precision than necessary. In such cases, trading precision for dynamic range through the use of floating point and logarithmic number system representations can potentially provide power savings. This is demonstrated for a subband speech coding application using architectural-level capacitance modeling.

1. INTRODUCTION

The chief concerns in the design of VLSI signal processors have traditionally been speed and layout area. More recently, the issue of circuit power dissipation has become prominent. This has been largely motivated by the emergence of portable computing and digitally-based communications devices, where conserving energy is important because of battery limitations. In non-portable computing environments, power dissipation is also an issue, since temperature affects packaging costs and circuit reliability [1]. Conversely, as effective chip areas have increased due to improved manufacturing and reduced feature sizes, area is somewhat less of a constraint than before; indeed, area is often sacrificed to reduce power [2].

Fixed point arithmetic is the norm for low power applications. However, fixed point representations may have to provide more precision than is needed for a particular application, simply to handle the dynamic range of the data encountered. For example, in lossy speech and image coding, output precision is reduced through quantization. This paper examines floating point and logarithmic representations, which both support wide dynamic ranges and have similar numeric properties, as low power alternatives to fixed point. Architecture-level power analysis of a speech compression application using subband coding is employed to study tradeoffs between various number representations and precisions.

2. POWER ESTIMATION

In well-designed CMOS circuits, the main source of power dissipation is the charging and discharging of node capacitances. In exploring the tradeoffs between alternative implementations of a processing module, an appropriate measure for comparison is the power dissipated over the duration of the access. This gives the *energy per access*,

$$E_m \quad = \quad C_m V_{dd}^2. \tag{1}$$

where C_m is the *switched capacitance*, and V_{dd} is the supply voltage [3]. The switched capacitance describes the average capacitance charged per access. Energy usage can be minimized by a combination of reducing V_{dd} (affecting circuit delay) and C_m. Reducing switched capacitance will be addressed here.

Generally the first step in assessing the tradeoffs among various approaches is an architectural level estimation, which is less accurate than transistor or gate level estimates, but is orders of magnitude less costly. At the architectural level, general formulas for the capacitances of building blocks such as adders and registers, parameterized by fundamental quantities such as word length, are used. The average switched capacitance for a complex operation can then be obtained by summing capacitances over all the component modules, weighted by activity factors describing the fraction of the time that a particular component is accessed [4]. Capacitance models for arithmetic operation under various numeric representations will now be discussed.

2.1. Fixed Point Arithmetic

Many past and present DSP systems rely on fixed-point arithmetic. Signal processing inputs are typically fixed point, and fixed-point hardware is well-understood, and relatively simple. In the multiply-accumulate operations which dominate signal processing, the bulk of the power is dissipated in the multiplier. For an $n_1 \times n_2$-bit multiply, the switched capacitance is modeled as

$$C_{mult}^{(Fix)}(n_1, n_2) \quad = \quad c_{\times}\, n_1 n_2, \tag{2}$$

and the capacitance of an n-bit add is

$$C_{add}^{(Fix)}(n) \quad = \quad c_{a1}\, n + c_{a2}, \tag{3}$$

where $c_{\times}$, c_{a1}, and c_{a2} are model constants [2][4].

Much research is being devoted to reducing the power dissipation of various fixed-point units; one approach to energy saving related to number representation is to reduce word sizes by truncating least significant bits [6].

2.2. Floating Point

Floating point number representations enjoy widespread use in scientific computing and signal processing because they simplify programming and provide wider dynamic range than fixed point for the same number of bits. Floating-point has not found favor in low power applications due to its perceived complexity and overhead [7]. A floating-point multiply consists of a fixed-point multiplication of the mantissas, and a fixed-point add of exponents (followed by a subtraction if excess notion is used). For a floating point representation with a k-bit excess-notation exponent and l-bit mantissa, the capacitance can be modeled as

$$C_{mult}^{(Flt)}(k,l) \quad = \quad C_{mult}^{(Fix)}(l) + 2C_{add}^{(Fix)}(k). \tag{4}$$

A float addition is composed of: a prenormalization step (subtracting exponents and shifting one mantissa); an addition of mantissas (possibly extended precision); and a postnormalization (shifting the sum, and adjusting the exponent); thus, the capacitance of a double precision accumulate can be modeled as

$$C_{add}^{(Flt)}(k,l) \quad = \quad C_{add}^{(Fix)}(2l) + 2C_{add}^{(Fix)}(k) + 2C_{shift}(2l) \tag{5}$$

To first order, the switched capacitance of an n-bit shifter is given by

$$C_{shift}(n) \quad = \quad c_{s1}n + c_{s2}\log(n+1), \tag{6}$$

assuming that the maximum allowed shift is n.

Depending upon the application, floating point may have some low power advantages. The underlying mechanism is that reducing the precision by a factor of a leads to a reduction in multiplicative power dissipation by a factor of a^2. The practical question is whether or not this savings offsets the added complexity of other floating-point operations. For the matrix-vector multiplies occuring in graphics projection transformations, floating point has been shown to dissipate less power than fixed point [7].

2.3. Logarithmic Number System

The Logarithmic Number System (LNS) can be interpreted as an "extreme" form of floating point. In LNS, a number X is represented by its sign, s_X, and the logarithm to the base β of its absolute value:

$$L_X \quad = \quad \log_\beta |X|. \tag{7}$$

Generally the base used is 2, and the n-bit fixed-point value is treated as consisting of a k-bit integer part and an l-bit fraction with $n = k + l$. The numeric properties of LNS and floating point are similar. The logarithmic system encodes 2^l values in the interval $[2^d, 2^{d+1})$, with distances which are harmonically related by the factor $2^{1/2^l}$, while a floating-point scheme with an l-bit mantissa plus hidden bit would divide the same interval into 2^l equal-sized pieces. LNS provides similar dynamic range to floating point, and statistical analyses show that it can be slightly more accurate than single precision floating point [8]. However, unlike LNS, floating point allows the possibility of double precision accumulation of sums.

The chief attraction of LNS arithmetic is that the product of two numbers A and B is computed by the fixed-point addition of their logarithmic representations:

$$s_{AB} \quad = \quad s_A \oplus s_B \tag{8}$$

$$L_{AB} \quad = \quad L_A + L_B. \tag{9}$$

Consequently, the switched capacitance is

$$C_{mult}^{(LNS)}(k,l) \quad = \quad C_{add}^{(Fix)}(k+l). \tag{10}$$

(The log can also be coded in excess notation as is commonly done with floating-point exponents but the overhead is higher since the full word add is required to restore the offset – clearly a drawback in the context of low power processing.) LNS multiplication clearly has major advantages over conventional fixed- and floating-point representations in regard to both speed and switched capacitance; a 5:1 power-delay advantage over floating point is reported for one particular implementaion [9].

Logarithmic addition and subtraction are more complicated; they are based on the identities

$$A \pm B \quad = \quad A(1 \pm B/A). \tag{11}$$

In logarithmic form,

$$L_{A\pm B} \quad = \quad L_A + U_\pm(L_B - L_A) \tag{12}$$

where

$$U_\pm(L_X) \quad = \quad \log_2(1 \pm 2^{L_X}) \tag{13}$$

Without loss of generality (at the cost of a compare), it can be assumed that $L_A \geq L_B$, so that U_+ and U_- are defined only for negative arguments.

The functions U_+ and U_- are typically implemented using table lookup, and represent a major impediment to widespread adoption of LNS. Table sizes grow exponentially with precision, and much LNS research activity is devoted to table compression in order to allow increased word lengths [10][11][12]. One characteristic that can be exploited is that the two functions decrease in magnitude as L_X decreases. Typically, the domain is divided into small intervals, each with its own ROM whose wordlength is no wider than necessary to encode the function for that interval. This fits in well with low power considerations; not only is ROM width reduced, but so is the number of active lines [4][13]. A reasonable model for memory access (ROM, register file, etc.) capacitance is given by [2][4]

$$C_{mem}(N,b) = c_{m1} + c_{m2}N + c_{m3}b + c_{m4}Nb, \tag{14}$$

where N is the number of entries in the memory bank and b is the width in bits.

Handling fractional parts of more than about thirteen bits requires the use of linear interpolation. This adds further overhead to LNS addition/subtraction, although it has been shown that the required multiplication can also be done logarithmically [10]. For the application to be considered next, it will be seen that large fractions are not required, and so LNS addition capacitance can be modeled without interpolation as

$$C_{add}^{(LNS)}(k,l) = 3C_{add}^{(Fix)}(k+l) + \sum_j p_j C_{mem}^{[j]}(N_j, b_j), \tag{15}$$

where p_j represents the probability that the jth ROM block is accessed.

3. SUBBAND CODING EXAMPLE

The tradeoffs between the above number representations were explored in the context of a speech coding application. Fourteen-bit speech data sampled at 14.7 kHz was input to the 24-tap quadrature mirror filter (QMF) pair from the G.728 recommendation [14]. Computationally, this is two-output-channel FIR filtering dominated by multiply-accumulate operations. Identical structures were implemented in fixed point, floating point, and LNS representations, at a variety of precisions. Two issues were addressed: the switched capacitance of each implementation; and the reconstruction distortion. The tradeoff between these two quantities is important, because some degree of degradation may be tolerated if enough power is saved.

3.1. Capacitance

The coefficients for the capacitance models are based on [2], and are shown in Table 1. Several comments are in order. First, this model is employed only to obtain relative capacitances for comparison purposes; different technology processes will give different values. Applying various low power design optimizations will have an effect; however, just as power can be reduced for fixed- and floating- point multiplies [5], ROM power can be reduced as well [13]. Second, the coefficients are based on random inputs, and may not accurately reflect switched capacitance for correlated inputs, since bit positions with little activity can effectively shorten the input words lengths [4]. This is most critical in the case of the multiplies, where capacitance varies as the square. In a QMF-pair, the ith high-band coefficient is related to that of the low band via $h_H[i] = (-1)^i h_L[i]$. Thus, the most efficient data/coefficient access pattern is to multiply and sum the even- and odd-numbered coefficients separately, which tends to reduce the effect of sample-to-sample correlations for each bit position, generating a more "white" process. Figure 1 shows the $0 \rightarrow 1$ transition probabilities of successive fixed-point multiplier inputs for each bit position. Random activity corresponds to a value of 0.25. The higher-order data bit positions have been whitened to a degree. The second multiplier input exhibits a worse than random switching activity in the high-order bits, due to the sign pattern of the filter coefficients.

The effective ROM switched capacitance depends upon both the ROM organization and the access frequency. It was assumed that there is a separate ROM bank for each unit interval in the domains of U_+ and U_-. The problem under consideration involves limited precision, so it was assumed that no linear interpolation was required. Next, because $|dU_+(x)/dx| \leq 1$ for all x, and $|dU_-(x)/dx| \leq 1$ for $x \leq -1$, stored values can change only by the value of the *lsb* between adjacent entries. Thus, it is sufficient to tabulate only the even-numbered entries, and store one additional bit per entry to indicate if the next (untabulated) odd-numbered entry is different; this converts a ROM of N b-bit entries into a ROM of size $N/2 \times (b+1)$ bits. Finally, it was also assumed that ROM blocks larger than 128 entries would be split. Access patterns were computed during the course of the simulations.

Switched capacitances (in pF) for single multiply-accumulate operations, showing their add and multiply components, are given in Table 2, along with ROM sizes (in bits) for LNS. In the *Precision* column, the floating point values indicate the mantissa length plus a hidden bit; for LNS the indicated value is for the fractional part only. A 5-bit exponent was used for floating point, while the LNS logarithms had 5-bit integer parts. To perform the reduced-precision fixed point processing, least significant bits were removed (with rounding). Double precision accumulates were used for fixed and floating point.

3.2. Reconstruction Distortion

To measure distortion, the following steps were performed. The filter bank outputs were passed through 64-bit floating-point reconstruction filters. This also included speech data processed by a 64-bit floating-point version of the QMF bank to serve as a reference. The distortion of the reconstructed signals relative to the reference was then measured. This serves to gauge the distortion inherent in each representation/precision combination, prior to compression. The Signal-to-Noise Ratio (SNR) was selected as the distortion measure, since it is revealing of reconstruction errors at low signal levels [15]. Next, the QMF outputs were quantized, the low frequency band to 5 bits, and the upper frequency band to 3 bits, and then put through the reconstruction and compared.

Performances are shown in Table 3. Both average and peak SNR are shown. Several observations can be made. Prior to quantization, the 14-bit fixed point introduced the least distortion. Reduced precision floating point and LNS could not achieve the SNR of the 14-bit fixed point. A 10+(1) bit floating-point mantissa and LNS 10-bit fraction were required to yield average SNR values within one dB of the SNR of the 14-bit integer version. Also observe that truncating the fixed-point arithmetic precision in an attempt to reduce capacitance caused a severe degradation in SNR.

After quantization, the story is different. The quantization noise masks the distortion caused by utilizing less precise arithmetic. A 7+(1)-bit floating-point mantissa and 9-bit LNS fraction yielded comparable performance to the 14-bit fixed-point version at about half the capacitance. If an extra 0.4 dB of distortion is acceptable, 5+(1)-bit float and 7-bit fraction LNS can be used, for power savings in excess of a factor of three. Finally, note that in spite of the close numeric correspondence between a k-bit fractional logarithm and a k-bit mantissa, the floating point version seemed to perform better for a given precision; recall that double precision accumulates were used with floating point.

4. CONCLUSIONS

Fixed point arithmetic is the norm for low power applications, but it may provide too much precision relative to the required dynamic range for some signal processing tasks. In a speech coding application, floating point and logarithmic number systems were compared to fixed point with regard to both accuracy and switched capacitance. For little or no increase in distortion, capacitance reduction factors of from two to three were achieved; in general, the distortion caused by reduced precision was masked by the effects of the compression quantization. The floating point representations generally required fewer bits than LNS to give the same distortion.

Future work will address the issue of V_{dd} reduction and delay.

5. REFERENCES

[1] M. Pedram, "Power Minimization in IC Design: Principles and Applications," *ACM Trans. Design Auto. Electr. Sys.*, Vol. 1, No. 1, Jan. 1996.

Table 1: Model Capacitance Coefficients (fF)

Coefficient	*Value*	*Coefficient*	*Value*
$c_{\times}$	253	c_{m1}	87
c_{a1}	214	c_{m2}	51
c_{a2}	-158	c_{m3}	35
c_{s1}	28.7	c_{m4}	8
c_{s2}	43.8	-	-

Table 2: Switched Capacitance (pF) Per Access

System	*Prec.*	*Mpy*	*Add*	*MAcc*	*ROM Size*
Fixed	10(*)	30.4	4.7	35.1	-
Fixed	12(†)	39.5	5.4	44.9	-
Fixed	14(†)	46.0	5.9	51.9	-
Float	4+(1)	10.1	1.6	11.6	-
Float	5+(1)	13.3	2.0	15.3	-
Float	6+(1)	17.1	2.4	19.5	-
Float	7+(1)	21.3	2.9	24.2	-
Float	8+(1)	26.1	3.4	29.5	-
Float	9+(1)	31.4	4.0	35.4	-
Float	10+(1)	37.1	4.6	41.7	-
Float	11+(1)	43.4	5.2	48.7	-
LNS	4	1.8	5.8	7.5	328
LNS	5	2.0	7.0	9.0	812
LNS	6	2.2	8.8	11.0	1914
LNS	7	2.4	11.8	14.2	4398
LNS	8	2.6	15.3	17.9	8320
LNS	9	2.8	17.2	20.0	17183
LNS	10	3.1	19.2	22.3	37438
LNS	11	3.3	21.3	24.6	83865
LNS	12	3.5	23.6	27.1	190064

* 12-bit filter coefficients † 13-bit filter coefficients

Table 3: Reconstruction SNR (dB)

		Before Quant.		*After Quant.*	
System	*Prec.*	*Avg*	*Peak*	*Avg*	*Peak*
Fixed	10	-23.3	-13.2	-17.1	-9.0
Fixed	12	-33.0	-22.8	-19.3	-11.6
Fixed	14	-39.2	-28.4	-19.8	-12.0
Float	4+(1)	-29.0	-20.2	-19.4	-12.3
Float	5+(1)	-32.3	-23.0	-19.5	-12.2
Float	6+(1)	-36.4	-26.5	-19.7	-12.0
Float	7+(1)	-37.2	-26.3	-19.9	-12.7
Float	8+(1)	-38.0	-27.2	-19.9	-12.2
Float	9+(1)	-38.7	-27.5	-20.1	-12.9
Float	10+(1)	-38.8	-27.4	-20.0	-12.4
Float	11+(1)	-38.7	-27.4	-19.8	-12.1
LNS	4	-19.2	-15.3	-16.0	-10.3
LNS	5	-25.5	-19.0	-18.5	-11.8
LNS	6	-28.4	-21.0	-19.0	-11.8
LNS	7	-31.3	-22.8	-19.5	-12.1
LNS	8	-34.6	-25.6	-19.6	-12.3
LNS	9	-37.3	-26.9	-19.8	-12.2
LNS	10	-38.3	-27.2	-19.7	-12.3
LNS	11	-38.5	-27.2	-19.8	-12.5
LNS	12	-38.7	-27.5	-19.9	-12.4

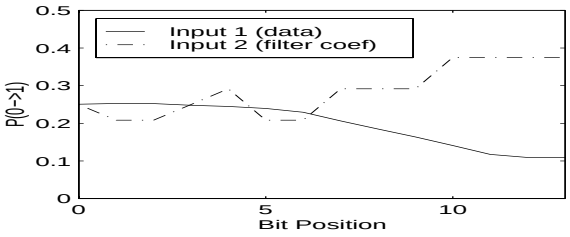

Figure 1: Bit Switching Activity of 14-bit Multiplier Inputs (First Input: Data Word; Second Input: Filter Coefficient)

[2] A. K. Chandrakasan, R. W. Broderson, *Low Power Digital CMOS Design*, Kluwer, 1995, pp. 282-283.

[3] H. A. Mehta, "System Level Power Analysis," Ph.D. Dissertation, Computer Science and Engineering Department, The Pennsylvania State University, University Park, PA, 1996.

[4] P. E. Landman, J. M. Rabaey, "Architectural Power Analysis: The Dual Bit Type Method," *IEEE Trans. VLSI Sys.*, Vol. 3, No. 2, June 1995, pp. 173-187.

[5] T. K. Calaway, E. E. Swartzlander, Jr., "Power-Delay Characteristics of CMOS Multipliers," *Proc. 1997 Int. Symp. Comp. Arith.*, pp. 26-32.

[6] Z-L He, K-K Chan, C-Y Tsui, M. L. Liou, "Low Power Motion Estimation Design Using Adaptive Pixel Truncation," *Proc. 1997 Int. Symp. Low Power Elec. Design*, pp. 167-172.

[7] K. P. Acken, "Low Power Architectural Optimizations for 3D Graphics Subsystems," Ph.D. Dissertation, Computer Science and Engineering Department, The Pennsylvania State University, University Park, PA, 1997.

[8] D. V. S. Chandra, "Accumulation of Coefficient Roundoff Error in Fast Fourier Transforms Implemented with Logarithmic Number System," *IEEE Trans. ASSP*, Vol. ASSP-35, No. 11, November 1987, pp. 1633-1636.

[9] F. J. Taylor, R. Gill, J. Joseph, J. Radke, "A 20 Bit Logarithmic Number System Processor," *IEEE Trans. Comp.*, Vol. 37, No. 2, February 1988, pp. 190-200.

[10] D. M. Lewis, "An Architecture for Addition and Subtraction of Long Word Length Numbers in the Logarithmic Number System," *IEEE Trans. Comp.*, Vol. 39, No. 11, November 1990, pp. 1325-1336.

[11] J. N. Coleman, "Simplification of Table Structure in Logarithmic Arithmetic," *Electron. Let.*, Vol. 31, No. 22, October 26, 1995, pp. 1905-1906.

[12] S-C Huang, L-G Chen, "A 32-bit Logarithmic Number System Processor," *J. VLSI Sig. Proc.*, Vol. 14, June 1996, pp. 311-319.

[13] E. de Angel, E. E. Swartzlander, Jr., "Survey of Low Power Techniques for ROMs," *Proc. 1997 Int. Symp. Low Power Elec. Design*, Aug. 1997, pp. 7-11.

[14] K. Sayood, *Introduction to Data Compression*, Morgan Kaufmann, 1996, pp. 297-305.

[15] A. Gersho, R. M. Gray, *Vector Quantization and Signal Compression*, Kluwer, 1992, pp. 326-327.