

# Express Coefficients in 13-ary, Radix-4 CSD to Create Computationally Efficient Multiplierless FIR Filters

Jeffrey O. Coleman\*

**Abstract** — A two's complement DSP signal can be multiplied by a fixed coefficient using an adder tree operating on input shifts corresponding to nonzero coefficient bits. Alternatively, the signed bits of a canonical-signed-digit (CSD) coefficient representation specify an add/subtract network, with one third fewer terms required on average. That well-known approach is generalized here to a radix-4 CSD system that turns out to save 36% relative to conventional CSD in the FIR-filter application but at a per-filter overhead cost of six small-integer scaling operations that represent common subexpressions implicitly factored from the add/subtract network. The approach works for both direct-form and transposed-form filters and generalizes easily to other number systems.

## 1 Introduction

A signal can be multiplied by a fixed coefficient in custom DSP hardware using those hardwired shifts and adds of the signal that are implicitly specified by the binary coefficient bits. The number of arithmetic operations required, ignoring shifts (which come free), can be reduced by an average of 33% by instead expressing coefficients in canonical-signed-digit (CSD) form. CSD is a radix-two number system with ternary coefficient set  $\{-1, 0, 1\}$  and having the “canonical” property that  $-1$  and  $1$  are always followed by  $0$  in CSD strings. CSD string  $1\ 0\ 0.\ -1\ 0\ 1$ , for example, represents  $3.625$ , the dot product of ternary coefficient vector  $(1, 0, 0, -1, 0, 1)$  with powers  $(2^2, 2^1, 2^0, 2^{-1}, 2^{-2}, 2^{-3})$ . So a CSD coefficient specifies which input-signal shifts to add to the output, which to subtract from it, and which to ignore.

In the next section, the principles underlying the CSD number system in standard radix-2 form [1] are applied to create the radix-4 equivalent—CSD4 is a good name for it—and determine its basic properties. These principles force CSD4 to be a 13-ary system, with digits drawn from alphabet  $\{-15, -13, -11, -9, -7, -5, 0, 5, 7, 9, 11, 13, 15\}$ . Section 3 shows how factoring the absolute values of these digit values (nearly) out of both the direct-form and the transposed-form FIR filter structures leads to efficient realizations, particularly for long or large-wordlength filters.

## 2 Number Systems

We use a two-step approach, first deriving CSD- $r$  families of number systems and then specializing to radix  $r = 4$ . This actually simplifies matters because it keeps the “fourness” of our particular system from obscuring the ideas behind the various systems’ properties.

### 2.1 Properties Desired

Let us begin quite generally and add restrictions to narrow our focus as we proceed. Denoting the real numbers by  $\mathbb{R}$ , begin with an arbitrary radix  $r \in \mathbb{R}$ , an arbitrary alphabet  $\mathcal{A} \subset \mathbb{R}$ , and the idea that, using arbitrary  $a_k \in \mathcal{A}$ ,

$$\begin{array}{l|l} \text{sequence} & \text{represents} \\ \hline 0.a_1 a_2 \dots & x = \sum_{k=1}^{\infty} a_k r^{-k} \\ a_1.a_2 a_3 \dots & rx = a_1 + \sum_{k=1}^{\infty} a_{k+1} r^{-k}. \end{array} \quad (1)$$

To ensure that the sums converge, require radix  $|r| > 1$  and require all elements of alphabet  $\mathcal{A}$  to meet some common magnitude bound. Of course if the number of elements in  $\mathcal{A}$  is finite, they are bounded automatically. But surprisingly,  $\mathcal{A}$  need not be finite.

The relation of product to shift in (1) above determines what set of numbers can be represented in the system. To begin, define set  $\mathcal{F}_\alpha \subset \mathbb{R}$  by requiring that its translation  $\alpha + \mathcal{F}_\alpha$  contain all products  $rx$  in (1) when  $a_1 = \alpha \in \mathcal{A}$  and the sequence  $a_1, a_2, \dots$  is finite in length. This makes  $\mathcal{F}_\alpha$  the set of possible fractional values when  $\alpha$  appears left of the radix point. Let us require our alphabet  $\mathcal{A}$  to contain zero, so that  $\mathcal{F}_0$  will be simply the set of possible fractions in our system. Then in the spirit of (1), the numbers in our system having just one “digit” and a finite-length fractional part can be written in either of two ways, so

$$r\mathcal{F}_0 = \bigcup_{\alpha \in \mathcal{A}} \alpha + \mathcal{F}_\alpha. \quad (2)$$

\*Naval Research Laboratory, Radar Division, Code 5327, 4555 Overlook Ave SW, Washington DC 20375-5336, USA. Email: jeffc@alum.mit.edu. This work was supported by the AMRFC program (ONR 31) of the Office of Naval Research, Arlington VA, USA.

Certain properties are desired in sets  $\alpha + \mathcal{F}_\alpha$ . First, any element of  $r\mathcal{F}_0$  above that falls in more than one of the sets  $\alpha + \mathcal{F}_\alpha$  on the right can be represented in our system in more than one way, that is, more than one choice of  $\alpha$  can be used for the first digit. So, to make representations of numbers as finite sequences in our system unique, we must require the sets on the right in (2) to be disjoint.

Second, let us set limits on the memory inherent in sequences. Ordinary binary is an example of a memoryless system, one in which the portions of the sequence to the right and left of any given reference point in the sequence can be chosen independently. Ordinary CSD has memory, however, because if the digit just to the left is nonzero, the digit just to the right must be zero. So, let's restrict our number system considerably while still permitting both these cases by requiring any memory to be finite, in a sort of Markovian spirit. To do this, structure the sequence to the right of any  $\alpha$  into an initial component followed by a tail component. Limit the length of the initial sequence component to  $N_\alpha$  and let  $\mathcal{I}_\alpha$  denote the  $\alpha$ -dependent set of values this initial component can represent. Then let the tail component of the sequence represent any value in some set  $\mathcal{T}$  that is independent of  $\alpha$ . This limits memory to  $N_\alpha$  digits and sets  $\mathcal{F}_\alpha = \mathcal{I}_\alpha + r^{-N_\alpha}\mathcal{T}$ . Now let's take a very restrictive but convenient step and further require  $\mathcal{I}_\alpha = 0$ , so that every digit of value  $\alpha$  in our system is followed immediately by  $N_\alpha$  (highly desirable) zero digits. Of course  $N_0 = 0$  is essential, else sequences will get "stuck at zero" forever, so  $\mathcal{F}_0 = \mathcal{T}$ . Now memory is limited not only in length but in character, and for any  $\alpha \in \mathcal{A}$ ,

$$\mathcal{F}_\alpha = r^{-N_\alpha}\mathcal{F}_0. \quad (3)$$

Third and finally, we need some kind of continuity in the set of representable numbers. In the ordinary case in which alphabet  $\mathcal{A}$  contains only a finite number of elements, our set  $\mathcal{F}_0$  of real numbers that can be represented with finite-length fractional sequences will be countably infinite and so certainly cannot be equal to an interval, the natural goal of our number-system construction. So let us also consider numbers that, while not representable exactly with finite-length sequences, are representable to arbitrary precision with such sequences. The number 1, for example, can be approximated as accurately as desired with a finite-length fractional binary sequence simply by taking enough digits of 0.111... Those reals not in  $\mathcal{F}_0$  but that can be approximated arbitrarily closely by elements of  $\mathcal{F}_0$  are just its limit points, which when appended to  $\mathcal{F}_0$  turn it into  $\overline{\mathcal{F}_0}$ , its (topological) closure. (This is one of several equivalent definitions [2] of closure in  $\mathbb{R}$ .) So in order to ensure that there are no gaps in the range of numbers representable in this expanded sense, let us require  $\overline{\mathcal{F}_0}$  to be a (nontrivial) closed interval.

This continuity requirement is central here, because scaling relationship (3) then forces every  $\overline{\mathcal{F}_\alpha}$  to be a closed interval as well, and making the union in (2) disjoint then means that the translations  $\alpha + \overline{\mathcal{F}_\alpha}$ , also closed intervals, must be placed end to end so that their endpoints just touch. Using  $\mu$  for the (nonzero) width of interval  $\overline{\mathcal{F}_0}$  (its Lebesgue measure), disjoint union (2) and scaling relationship (3) now imply that

$$r\mu = \sum_{\alpha \in \mathcal{A}} r^{-N_\alpha}\mu,$$

or simply

$$\sum_{\alpha \in \mathcal{A}} r^{-(N_\alpha+1)} = 1. \quad (4)$$

The terms of sum (4) give the relative proportions of  $\overline{\mathcal{F}_0}$  represented by sequences beginning with each  $\alpha \in \mathcal{A}$ . (Constraint (4) forces alphabet  $\mathcal{A}$  to be countable, but nothing yet requires  $\mathcal{A}$  to be finite, an interesting fact of no obvious usefulness.)

Substituting (3) into (2), taking the closure, and dividing by radix  $r$  produces the recursive relationship central to our construction. The intervals in this union touch only at their endpoints:

$$\overline{\mathcal{F}_0} = \bigcup_{\alpha \in \mathcal{A}} \frac{\alpha}{r} + r^{-(N_\alpha+1)}\overline{\mathcal{F}_0}. \quad (5)$$

## 2.2 Construction Procedure

Given the radix  $r$ , we can now design a number system using a three-step procedure.

**Choose the range of fractions.** Choose some nontrivial closed interval  $\overline{\mathcal{F}_0}$  containing the values to be represented arbitrarily closely by fractional sequences. The all-zero sequence is a finite (in the required sense) fractional sequence, so  $0 \in \mathcal{F}_0$  is required, and therefore so is  $0 \in \overline{\mathcal{F}_0}$ .

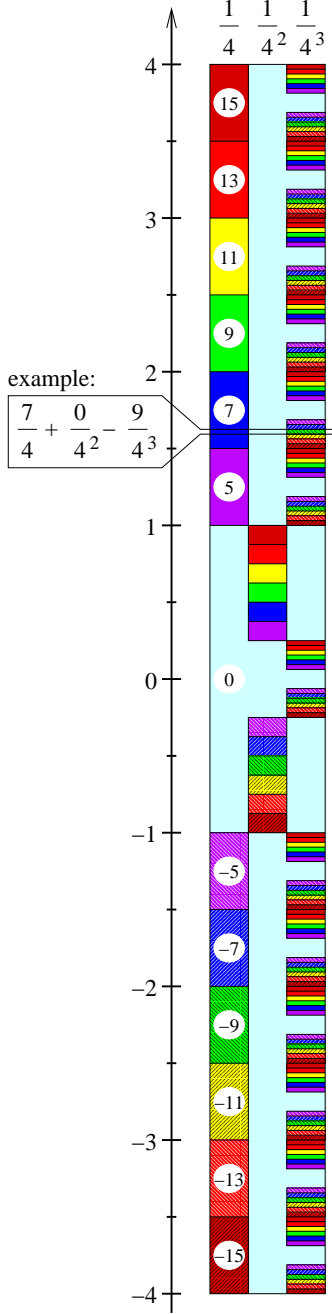
**Proportion according to initial digit.** Choose proportions  $r^{-(N_\alpha+1)}$  so that their sum is unity, with  $r^{-(N_0+1)} = 1/r$  required.

**Choose the alphabet.** Each  $r^{-(N_\alpha+1)}$  value just chosen determines one of the sets  $\alpha/r + r^{-(N_\alpha+1)}\overline{\mathcal{F}_0}$  in (5). It remains only to choose each  $\alpha$  to place these sets end to end.

Scaling every  $\alpha$  and  $\mathcal{F}_\alpha$  afterward by some common factor is sometimes necessary (and sometimes inadequate) to produce the integer-only alphabet that (only) custom requires.

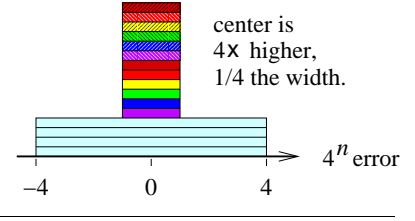
The simplest example is ordinary binary, a radix-2 system with a binary alphabet. Once the range of fractions  $\overline{\mathcal{F}_0}$  is set to interval  $[0, 1]$ , the proportions must be chosen as  $\frac{1}{2}$  and  $\frac{1}{2}$ , and alphabet  $\mathcal{A} = \{0, 1\}$  follows.

Figure 1: A three-digit ruler for CSD4 fractions.



Conventional CSD is also simple. Specify radix 2 and a ternary alphabet, and center the range of fractions  $\overline{\mathcal{F}}_0$  about zero for unbiased truncation errors (more on this below, in the radix-4 context). The initial scale is irrelevant, so choose  $\overline{\mathcal{F}}_0 = [-1, 1]$  for convenience, and choose proportions  $\frac{1}{2}$ ,  $\frac{1}{4}$ , and  $\frac{1}{4}$ , assigning the two nonzero  $\alpha$ 's the same pro-

Figure 2: The asymptotic (in the number of digits kept) error pdf for CSD4 conversion of a uniform random variable.



portion of the total range, a reasonable symmetry. Recursion (5) now requires choosing these two nonzero  $\alpha$  values to place the two  $\alpha/r + \frac{1}{4}\overline{\mathcal{F}}_0$  intervals on either side of and end-to-end with interval  $\frac{1}{2}\overline{\mathcal{F}}_0$ . Given our earlier choice of  $\overline{\mathcal{F}}_0 = [-1, 1]$ , this requires  $\mathcal{A} = \{-\frac{3}{2}, 0, \frac{3}{2}\}$ . Scaling by  $\frac{2}{3}$  will yield the smallest possible integer magnitudes for our final alphabet  $\mathcal{A} = \{-1, 0, 1\}$  and will set  $\overline{\mathcal{F}}_0 = [-\frac{2}{3}, \frac{2}{3}]$ . (Reference [1] has more to say on this example.)

### 2.3 A New System: CSD4

Radix-4 CSD is now straightforward to construct. Let the arbitrary initial zero-centered choice of  $\overline{\mathcal{F}}_0$  be  $[-\frac{1}{2}, \frac{1}{2}]$  for its convenient unit width. Since  $N_0 = 0$  is always required and  $N_\alpha = 1$  for  $\alpha \neq 0$  is here needed also, so that nonzeros are always followed by zeros, proportions  $r^{-(N_\alpha+1)}$  are  $\frac{1}{4}$  for  $\alpha = 0$  and  $\frac{1}{16}$  for nonzero  $\alpha$ . There must be exactly twelve of the latter for the proportions to sum to unity. Place the nonzero- $\alpha$  intervals  $\frac{\alpha}{4} + \frac{1}{16}\overline{\mathcal{F}}_0$  of recursion (5) end to end by setting  $\alpha = \pm \frac{2k+3}{8}$  for  $k = 1, \dots, 6$ . Now scale the whole system by 8 so that the  $\alpha$  are the smallest integers possible. The system of Fig. 1 results, with  $\overline{\mathcal{F}}_0 = [-4, 4]$  and  $\mathcal{A} = \{-15, -13, -11, -9, -7, -5, 0, 5, 7, 9, 11, 13, 15\}$ . This is CSD4.

What is the average proportion of zeros in CSD4 digit strings? Suppose a random variable uniformly distributed over  $\overline{\mathcal{F}}_0$  is expressed in CSD4, and let  $Z_n$  denote the event “[the  $n$ -th CSD4 digit is zero].” Then, using conditionals,

$$P(Z_n) = P(Z_n | Z_{n-1})P(Z_{n-1}) + P(Z_n | Z_{n-1}^c)(1 - P(Z_{n-1}))$$

with (see Fig. 1)

$$P(Z_n | Z_{n-1}) = \frac{1}{4}$$

$$P(Z_n | Z_{n-1}^c) = 1,$$

so  $P(Z_n) = 1 - \frac{3}{4}P(Z_{n-1})$ . Since  $P(Z_1) = \frac{1}{4}$ , this recursion is solved by

$$P(Z_n) = \sum_{k=0}^{n-1} \left(-\frac{3}{4}\right)^k$$

$$= \frac{4}{7} + \frac{3}{7}\left(-\frac{3}{4}\right)^n$$

$$\rightarrow \frac{4}{7}. \quad (6)$$

In the coefficient-multiply application, each nonzero CSD4 digit, asymptotically  $\frac{3}{7}$  of the total, specifies an add or subtract of the input after it is shifted by an even number of bits to multiply it by some  $4^{-k}$ , and no operations are required on the input shifted by odd numbers of bits. A CSD4 coefficient multiplication therefore asymptotically requires summing  $\frac{3}{14}$  of the possible input shifts. Ordinary CSD is well known to require  $\frac{1}{3}$ , so the

$$\frac{3}{14} = \frac{1}{3} \left(1 - \frac{5}{14}\right)$$

of CSD4 saves  $\frac{5}{14} \approx 36\%$  compared to CSD, “on the average.” (The price paid for this will be discussed shortly.)

If this random CSD4 fraction is truncated after one digit, (5) gives the truncation-error distribution, conditioned on  $\alpha$ , as  $r^{-(N_\alpha+1)}$  times a uniform distribution over  $\overline{\mathcal{F}}_0$ . Truncating after  $n$  digits instead just scales this by  $r^{-(n-1)}$ , so the truncation error after  $n$  digits is  $r^{-(N_\alpha+n)}$  times a random variable distributed uniformly on  $\overline{\mathcal{F}}_0 = [-4, 4]$ , where  $\alpha$  is now the value of

the  $n$ th digit. This suggests using conditional densities:

$$f_n(x) = f_n(x | Z_{n-1}) P(Z_{n-1}) + f_n(x | Z_{n-1}^c) (1 - P(Z_{n-1})).$$

The *a priori* probabilities are given by (6). The simple asymptotic result is sketched in Fig. 2. (The algebraically awkward general case is left to the reader.) The asymptotic standard deviation, using conditionals, is

$$4^n \sigma = \sqrt{\frac{8^2}{12} \times \frac{4}{7} + \frac{2^2}{12} \times \frac{3}{7}} = \sqrt{\frac{67}{21}}.$$

CSD-fraction values are upper bounded in magnitude by 4, so the CSD4 dynamic range, computed as  $20 \log(|\text{peak}|/\sigma)$ , is  $\approx 4.6$  dB plus 12 dB per CSD4 digit kept (as against  $\approx 6$  dB plus 6 dB per CSD digit kept for ordinary CSD).

Finally, recursion (5) implies an algorithm to convert to CSD4. (The approach was used for CSD in [1].)

### 3 Efficient FIR Filters Using CSD4 Coefficients

The output of a direct-form FIR filter is computed as linear combination  $y(n) = \sum_k h(k) x(n-k)$ . Here we can write the coefficients in CSD4 as  $h(k) = \sum_i a_i(k) r^{-i}$  with  $a_i(k) \in \mathcal{A}$  for

$$\begin{aligned} y(n) &= \sum_{k,i} r^{-i} a_i(k) x(n-k) \\ &= \sum_{k,i} r^{-i} |a_i(k)| \text{sgn}(a_i(k)) x(n-k). \end{aligned} \quad (7)$$

Since  $|a_i(k)|$  can take only seven possible values, one of which is zero, we can formally recognize function  $\mathcal{A} \xrightarrow{|\cdot|} \mathbb{R}$  as a “simple function” and write

$$|a_i(k)| = \sum_{m=1}^6 \gamma_m 1_{A_m}(a_i(k)),$$

where set-membership function  $1_{A_m}(\cdot)$  takes value unity or zero according as its argument is or is not in set  $A_m$ . Set  $A_m \subset \mathcal{A}$  contains all the alphabet elements with absolute value  $\gamma_m$ . Substituting into (7) yields

$$y(n) = \sum_{m=1}^6 \gamma_m R S_m \begin{pmatrix} x(n) \\ x(n-1) \\ \vdots \end{pmatrix}$$

where matrix  $S_m$  has elements in  $\{0, -1, 1\}$  given by

$$[S_m]_{ik} = 1_{A_m}(a_i(k)) \text{sgn}(a_i(k)),$$

and where matrix  $R$  is a diagonal matrix of digit-shifting factors  $r^{-i}$ . The result is now expressed as a linear combination of six intermediate outputs computed from the same data but with different coefficient

data matrices  $S_m$ . Element  $i, k$  of coefficient matrix  $S_m$  tells whether to add, to subtract, or to not include a  $2i$ -bit shift of  $x(n-k)$  in the summation that forms intermediate output  $m$ . These matrices have column dimension equal to the filter length (suitably modified for linear phase, etc.), have row dimension of about half the desired output wordlength in bits, and, according to the probabilistic analysis above, are quite sparse, with nonzero entries in about  $\frac{3}{7} \times \frac{1}{6} = \frac{1}{14}$  of their elements.

## 4 Observations and Conclusions

### Recipe: Efficient FIR-Filter Structure

**Ingredients:** One set of delay-line registers containing signal data, six sparse shift/add/subtract networks to compute intermediate outputs, and a linear combiner with weights 5, 7, 9, 11, 13, and 15.

**Features:** The linear combiner is fixed in size and complexity, no matter the filter length. And the idea also works in transposed form: The input is scaled by the six factors to create “intermediate inputs,” and then a sparse shift/add/subtract network operates on those six inputs to create data to sum into the delay line. For either direct or transposed form, the entire approach can be viewed as a form of common-subexpression elimination based on factoring out common number-system scale factors.

Here CSD4 was developed from general principles that invite the design of other number systems. Each such system implies a particular set of common subexpressions that are easily factored out of an FIR filter. The CSD4 path through these dense, generalized woods was (partially) cleared by this paper, largely to demonstrate the approach. Whether this approach better bypasses or combines with traditional algorithms for “optimal” common-subexpression elimination (e.g. [3]), time will tell.

## References

- [1] J. O. Coleman and A. Yurdakul, “Fractions in the Canonical-Signed-Digit Number System,” in *Proc. 2001 Conf. on Information Sciences and Systems*, (Johns Hopkins University), Mar. 2001.
- [2] W. Rudin, *Principles of Mathematical Analysis*. New York: McGraw-Hill, third ed., 1976.
- [3] A. Yurdakul and G. Dündar, “Multiplierless realization of linear DSP transforms by using common two-term expressions,” *J. VLSI Signal Processing* 22, pp. 163–172, 1999.