

Optimizing Computations in a Transposed Direct Form Realization of Floating-Point LTI-FIR Systems *

N. Sankarayya and K. Roy

Purdue University

Dept. of Electrical & Computer Engineering

West Lafayette, IN 47907-1285

sankar, kaushik @ecn.purdue.edu

D. Bhattacharya

Texas Instruments Inc.

13510 N. Central Expressway

Dallas, TX 75243

bhattach@sunspot.hc.ti.com

Abstract

The inherent computational redundancy in discrete-time LTI-FIR system response computations in Digital Signal Processing have been exploited in a variety of ways to minimize the computational complexity. We present an improved algorithm-level computational optimization that uses sorted recursive differences between coefficients representing the system transfer function with a Floating-Point number representation to extract maximum benefits from this redundancy. It can be applied to any LTI-FIR system and there is no deterioration in accuracy compared to directly using the coefficients. The results for several practical FIR systems show that there is a significant reduction in the computational complexity, hence power consumed, using this technique.

1 Introduction

The recent proliferation of high-performance portable battery-powered electronic gadgets that make extensive use of Digital Signal Processing (DSP) like cellular phones, laptop computers, etc. has made power consumption in VLSI DSPs an increasingly important issue. This has made low-power-high-performance VLSI DSP an important research area. One of the most basic operations performed in DSP applications is the Finite Impulse Response (FIR) computation. As is well known, the output of a Linear Time-Invariant (LTI) system can be obtained by convolving in the time domain the input to the system and the transfer function of the system. For discrete-time LTI-FIR systems, this can be expressed mathematically as:

$$Y_j = \sum_{n=0}^{N-1} C_n \cdot X_{j-n} \quad (1)$$

*This research was supported in part by DARPA (F33615-95-C-1625), NSF CAREER award (9501869-MIP), IBM, AT&T and Rockwell

The notation A_r is used to denote the r -th term of the sequence A . X and Y are the discrete-time input and output signals, respectively, represented as sequences. The sequence C represents the transfer function of the system. For FIR filters, C also corresponds to the set of filter coefficients and the length of the sequence which is N , is called the number of taps or the length of the filter.

FIR system realizations can vary widely from one that uses dedicated hardware multipliers and adders to one that uses software executed by a general purpose processor and its ancillary units. A combination of hardware and software that allows sharing of multi-purpose hardware units (like adders and multipliers), without using a full-fledged processor can also be used. In all these implementations the basic FIR computation algorithm called the *multiply and accumulate* (MAC) sequence, remains the same: for any j , each *product term* $C_n \cdot X_{j-n}$ in Eqn. (1) is computed, and they are added up to obtain Y_j . Therefore the large number of taps required for practical narrow-transition bandwidth FIR filters [4] means a large number of multiplications per filter output and hence more power dissipated in multiplications.

2 Low-Power FIR Filter Realization

There are three sources of power dissipation in CMOS circuits: switching power P_{sw} , short-circuit power P_{sc} , and leakage power $P_{leakage}$ [3]. P_{sw} is often the most significant source, therefore efforts to reduce the power consumed in FIR system realizations, focus on reducing P_{sw} .

Since multiplications represent the most complex task in FIR computations, a lot of research has been carried out on reducing the complexity of or totally eliminating multiplications in computing the product terms in Eqn. (1). Some of these also lead to a reduction in the net switching power consumed and hence low-power FIR system realizations. Some of these earlier efforts towards reducing the complexity of multiplications are : approximation by signed and unsigned powers-of-two coefficients (2PFIR) [6, 21], coefficient representation using redundant number systems

including Canonical Signed Digits (CSD) encoding [17] and Residue Number Systems (RNS) [10], coefficient word-length reduction using sigma-delta modulation [15, 18] and linear predictive coding [22], filter realization as a prefilter-equalizer pair [7, 8] or as cascaded sub-filters [19] with an overall reduced multiplication complexity, the use of Distributed Arithmetic architectures for filter implementation [5], multiplier-less stored-product digital filters (SPDF) using ROMs [14], the frequency-response masking technique for design of filters [23], realization of filters using multi-stage interpolation and decimation units [9, 20, 16] and the extraction of common sub-expressions across coefficients for maximizing computational reuse [1, 2, 11, 12]. While significant reductions have been achieved using these techniques, quite a few rely on approximating the optimally designed coefficients to achieve their results.

3 Sorted Recursive Differences

We had earlier analyzed the usage of recursive differences between coefficients, called the Differential Coefficients Method (DCM) [13] for reducing the complexity of multiplications. One limitation of DCM was that it could be applied only to systems where the envelope generated by the coefficient sequence (and various orders of differences) is a smoothly varying continuous function, thus it was beneficial largely for low-pass FIR filters. We present an improved version of DCM, called the *Sorted Recursive Differences* (SRD) method, which uses recursive sorting of coefficients and various orders of differences to maximize the computational reduction. This recursive sorting is made possible by the Transposed Direct Form of FIR output computation. Thus there are no restrictions on the coefficient sequence to which it is applicable (or the sequences of various orders of differences). The effective word-length reduction using DCM was not the same for each coefficient. Instead of pessimistically using the worst case reduction as we did earlier we will now use a simple statistical estimate for the effective reduction in the number of ones in the coefficients. The earlier analysis was implicitly for a fixed-point implementation because it used the ratios of the dynamic ranges of various orders of differences to arrive at the effective word-length reduction. This paper presents the application of SRD to a Floating-Point (FP) number representation and it is shown that there are significant benefits in terms of computational reduction in doing this.

3.1 Differential Coefficients Method

We first present a brief recapitulation of DCM [13]. The *first-order differences* $\delta_{k-1/k}^1$ between consecutive terms C_k and C_{k-1} of the sequence C , was defined using the recurrence:

$$C_k = C_{k-1} + \delta_{k-1/k}^1 \quad \text{for } k=1 \text{ to } (N-1) \quad (2)$$

Therefore a product term $C_k X_j$ can be expressed recursively in terms of the previous product term $C_{k-1} X_j$ and the first-order partial product $\delta_{k-1/k}^1 X_j$. Thus we can compute all product terms involving data X_j , excepting the first $C_0 X_j$ which has to be computed directly, by computing only the first-order partial products by actual multiplication and adding them to previous product terms. This was called the *first-order differences algorithm* for computing the filter output and it requires $N - 1$ extra additions for computing all product terms.

We can also use greater orders of differences to recursively define coefficients. The *second-order differences* $\delta_{k-2/k}^2$, between consecutive first-order differences, was defined using the recurrence:

$$\delta_{k-1/k}^1 = \delta_{k-2/k-1}^1 + \delta_{k-2/k}^2 \quad \text{for } k=2 \text{ to } (N-1) \quad (3)$$

From this definition and Eqn. (2), except C_0 and C_1 , all other coefficients can be expressed as:

$$C_k = C_{k-1} + \delta_{k-2/k-1}^1 + \delta_{k-2/k}^2 \quad \text{for } k=2 \text{ to } (N-1) \quad (4)$$

Thus we can compute all first-order partial products involving data X_j , excepting the first one $\delta_{0/1}^1 X_j$ which has to be computed directly, by computing only second-order partial products $\delta_{k-2/k}^2 X_j$ by actual multiplication and adding them to previous first-order partial products. And all product terms, except the first, can be generated recursively if we have all the first-order partial products and the first product term. For example if we compute $C_0 X_j$ and $\delta_{0/1}^1 X_j$ by direct multiplication, $C_1 X_j$ can be obtained as $C_0 X_j + \delta_{0/1}^1 X_j$ and $C_2 X_j$ can be obtained by computing $\delta_{0/2}^2 X_j$ by multiplication and adding it to the sum of $C_1 X_j$ and $\delta_{0/1}^1 X_j$. All remaining product terms can be obtained likewise by computing only one second-order partial product by multiplication and this was called the *second-order differences algorithm*.

The algorithm was generalized to use upto m -th order differences, with the m -th order difference defined as:

$$\delta_{k-m/k}^m = \delta_{k-m+1/k}^{m-1} - \delta_{k-m/k-1}^{m-1} \quad \text{for } k=m \text{ to } (N-1); m=2 \text{ to } (N-1) \quad (5)$$

The recurrence relation between coefficients using m -th order differences is as follows:

$$C_k = C_{k-1} + \sum_{i=1}^{m-1} \delta_{k-i-1/k-1}^i + \delta_{k-m/k}^m \quad \text{for } k=m \text{ to } (N-1); m=2 \text{ to } (N-1) \quad (6)$$

We can again compute each product term incrementally as shown before for the second-order differences algorithm and each product term from $C_m X_j$ upwards will require computation of only one m -th order partial product by multiplication. It can be seen that a total of : $1 + 2 + \dots + (m-1) + m(N-m) = \frac{m}{2}(2N-m-1)$, extra additions are needed.

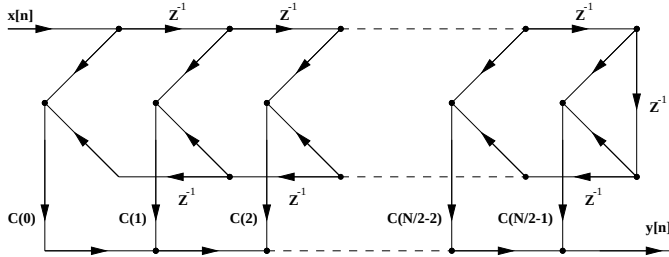


Figure 1: Signal flow graph for Direct Form realization of an even length FIR system.

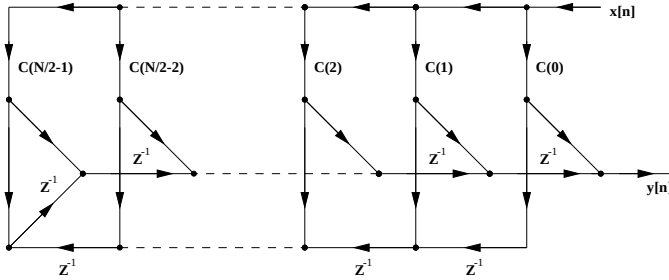


Figure 2: Signal flow graph for Transposed Direct Form realization of the same FIR system.

3.2 Transposed Direct Form Computation

In a Transposed Direct Form (TDF) FIR computation, all N product terms involving a particular data are computed before any product term with the next sequential data is computed. The product terms are accumulated in N different registers since they belong to N sequential outputs. The effective throughput remains the same as direct form computation. Signal flow graphs for direct form and TDF computation are shown in Figs. 1 and 2 respectively.

One advantage of using TDF lies in the fact that it does not matter in which order we compute the product terms involving a particular data so long as we accumulate them in the right registers. Therefore we can sort the coefficients in non-decreasing (or non-increasing) order before taking first order differences. It can be shown that this ordering minimizes the sum of the absolute value of first order differences. Sorting can also be applied to differences. Thus we can generate the second order differences from the sorted set of first order differences. This could be recursively done upto any order of differences. The various permutations of the sets of different orders of differences needed for a correct restoration could be hardwired in the control unit. This would enable it to accumulate partial products in appropriate locations so that the correct output is produced. Thus if DCM is applied in a TDF realization with recursive sorting we can eliminate the restrictions that were earlier imposed on the coefficients and various orders of differences for DCM to be viable.

3.3 Number Representation

Whereas sorting guarantees that differences of all orders will be non-negative, the coefficients can be positive or negative. When two consecutive coefficients in the sorted sequence are of opposite signs the magnitude of the algebraic difference between them is larger than the magnitude of either one. To decrease the range of the coefficients (hence smaller differences) we will use absolute values to compute differences and then manipulate the sign. As an example consider two consecutive coefficients C_r , C_{r+1} , with values $+0.8$ and -0.9 . The first-order differences algorithm would compute $C_{r+1}X_j$ as: $C_rX_j + \delta_{r/r+1}^1 X_j$. Let $\delta_{r/r+1}^1$ be generated as the difference between the absolute values of C_r and C_{r+1} , which in this case is $+0.1$. Then we can compute the partial product $+0.1X_j$, add it to C_rX_j and complement the sign-bit of the sum to obtain the correct value of $C_{r+1}X_j$. The sign-bit manipulation could be handled by the control unit and if the differences are known in advance the necessary sequence of sign-bit manipulations could be hardwired in the control unit. As another example suppose the two terms had values -0.7 and $+0.6$. In this case too we could compute $+0.1X_j$, add it to C_rX_j and complement the sign-bit of the sum to obtain the correct value of $C_{r+1}X_j$. Thus we see that a sign-magnitude number representation is very suitable for SRD and differences will be computed between absolute values of terms and it is assumed that necessary sign-bit changes are done by the control unit.

Towards further reducing the magnitudes of differences, we looked at the benefits of using a simplified Floating-Point (FP) number representation for the coefficients and data. Note that the mantissas of non-zero coefficients will always lie between 0.5 and 1 if they are normalized, thus allowing the same percentage precision for all coefficients. If differences are computed between mantissas, since only the mantissas are actually multiplied together in a FP multiplication, and if the $[N/2]$ (symmetric filter) mantissas are uniformly distributed in the range 0.5 to 1, the average first-order difference can be expected to be: $(1 - 0.5)/[N/2] \approx 1/N$. For example if we have 50 taps, this amounts to a 5 to 6 bits drop in magnitude of the mantissas of first-order differences. Therefore compared to the mantissas of C the first 5 to 6 bits of these will be all zeroes. A larger N therefore leads to smaller magnitudes of first-order differences. This is also roughly true for higher orders of differences. However since the distribution is not really uniform and to prevent the accumulation of the spread (small and large magnitude differences representing the tail ends of a more realistic Gaussian distribution) we could normalize the mantissas each time we take differences without changing the accuracy. However the controller will have to track a set of exponents associated with each order of difference to shift the partial products when recursively restoring, by the amounts they were shifted when normalizing. In the following analysis,

for the sake of simplicity, we will ignore the computations involving the exponents and the sign-bits.

4 Multiplication Complexity Analysis

Let the coefficients or the differences (as appropriate) be the multiplier and the data be the multiplicand. Let the word-widths used for mantissas of coefficients and data be W_c and W_x bits, respectively. The word-width required for a product term is therefore $(W_c + W_x)$ bits.

To analyze the computational reduction using SRD we will express a multiplication as equivalent shifts and additions. A multiplication will be performed by repeatedly adding the multiplicand to the left end of a partial sum, with the MSB of the multiplicand aligned with the second MSB of the partial sum (to allow for a carry from the addition). The multiplicand will be added only if the multiplier bit is one. The partial sum will be arithmetically right shifted by one bit after each addition. At the beginning of the multiplication the partial sum will be initialized to zero or the multiplicand, depending on whether the LSB of the multiplier is zero or one. The word-width of the partial sum will be equal to the word-width of a product term.

Consider the application of this model to a W_x bit by W_c bit multiplication. Since additions are skipped where the multiplier bits are zero, the number of additions required for a W_c bits wide multiplier can range from $(W_c - 1)$ to zero and it can be considered a discrete random variable (RV) between these bounds. Assuming a uniform distribution it has an expected value of: $\mathcal{E}\{n_a\} = \frac{1}{2}(W_c - 1)$ (any other symmetrical density function, for example a truncated Gaussian distribution, will give the same result). The expected value of this RV will be used to obtain a statistical estimate of the number of additions. Each addition of the W_x bits wide multiplicand can be considered equivalent to $(W_x + 1)$ one bit full additions. Therefore we get the expected value of the total number of one bit full additions for the multiplication as:

$$n_A = (W_x + 1)\mathcal{E}\{n_a\} = \frac{1}{2}(W_x + 1)(W_c - 1) \quad (7)$$

Similarly we can arrive at an expression for the total expected number of shifts. Since a partial sum needs a width of $(W_c + W_x)$ bits, right shifting it by one bit involves shifting a field $(W_c + W_x)$ bits wide and $(W_c - 1)$ such one bit shifts are needed for an W_x by W_c multiplication. This can be considered equivalent to a total of $(W_c + W_x)(W_c - 1)$ one bit shifts. Alternatively, if we assume that a multiple bit shifter (like array shifters, barrel shifters or logarithmic shifters) is used and that a run of zeroes in the multiplier is handled in a single shift operation, the number of shifts would be approximately the same as the number of adds, since each add must be preceded by a shift (of variable size). We get the expected value of the number of variable

sized shifts for the multiplication as:

$$n_S \approx (W_c + W_x)\mathcal{E}\{n_a\} = \frac{1}{2}(W_c + W_x)(W_c - 1) \quad (8)$$

For a symmetric (or antisymmetric) N tap FIR filter we have only $\lceil N/2 \rceil$ unique coefficients and the unique multipliers for SRD is the sequence: $C_0, \delta_{0/1}^1, \dots, \delta_{0/m-1}^{m-1}, \delta_{0/m}^m, \delta_{1/m+1}^m, \dots, \delta_{\lceil N/2 \rceil - m - 1 / \lceil N/2 \rceil - 1}^m$. The notation $|\delta_{k-m/k}^m|$ will be used to denote the reduced word-width required for the mantissa of $\delta_{k-m/k}^m$. An expected value of the number of ones in each SRD multiplier is derived below. A reduction in the magnitudes of the mantissas by taking differences can be considered as effectively skewing the distribution of the total number of ones in its binary representation, since a string of MSBs will be all zero. Let n'_a be the discrete one dimensional RV for the number of adds required for each SRD multiplier. Assuming a symmetrical distribution (this was verified by simulations) over its domain and using the relationship derived earlier between the word sizes of the multiplicand and multiplier and the total expected number of adds required for our shift and add model, we obtain: $n'_A = (W_x + 1)\mathcal{E}\{n'_a\}$ for each SRD multiplier. For example for $\delta_{k-m/k}^m$ it is: $\frac{1}{2}(W_x + 1)(|\delta_{k-m/k}^m| - 1)$. Summing up, we get the net expected number of adds for all the SRD multipliers as:

$$\begin{aligned} \sum_{\forall \text{mults}} n'_A &= \sum (W_x + 1)\mathcal{E}\{n'_a\} \\ &= \frac{1}{2}(W_x + 1) \left(\sum_{k=m}^{\lceil N/2 \rceil - 1} |\delta_{k-m/k}^m| + \sum_{k=1}^{m-1} |\delta_{0/k}^k| + |C_0| - \lceil N/2 \rceil \right) = N'_A \quad (9) \end{aligned}$$

We can likewise obtain the net expected number of adds using the coefficients directly as:

$$\begin{aligned} \sum_{\forall \text{mults}} n_A &= \frac{1}{2}(W_x + 1) \left(\sum_{k=0}^{\lceil N/2 \rceil - 1} |C_k| - \lceil N/2 \rceil \right) \\ &= \frac{1}{2}(W_x + 1)(W_c - 1)\lceil N/2 \rceil = N_A \quad (10) \end{aligned}$$

The expected value of the number of adds can be directly used to approximate the expected value of the number of shifts as mentioned before. Therefore from Eqns. (8) and (9) we get the net expected value of the shifts required for the SRD multipliers as:

$$\begin{aligned} \sum_{\forall \text{mults}} n'_S &= \sum (W_c + W_x)\mathcal{E}\{n'_a\} \\ &= \frac{(W_c + W_x)}{(W_x + 1)} N'_A = N'_S \quad (11) \end{aligned}$$

Likewise for the direct form multipliers we get :

$$\sum_{\forall \text{mults}} n_S = \frac{(W_c + W_x)}{(W_x + 1)} N_A = N_S \quad (12)$$

Table 1: Maximum Savings in adds using SRD

FIR Sys.	N	Simulated max S_{ADD} , opt m			
		8 bits	12 bits	16 bits	24 bits
Low Pass	51	37.9, 2	24.9, 3	23.8, 4	11.0, 1
	101	56.0, 2	44.3, 3	39.8, 4	30.7, 5
	201	63.5, 2	56.5, 4	52.5, 3	45.1, 5
Band Stop	51	46.0, 2	31.1, 3	23.5, 4	11.5, 6
	101	50.4, 2	44.7, 3	36.9, 4	30.2, 4
	201	62.2, 2	55.5, 2	50.1, 3	43.9, 5
Band Pass	51	43.5, 2	32.7, 3	23.9, 4	12.4, 3
	101	51.4, 2	43.8, 3	36.9, 3	30.3, 4
	201	62.9, 2	55.8, 3	50.2, 4	43.4, 5
High Pass	51	34.9, 2	21.5, 3	23.0, 4	12.4, 1
	101	53.9, 2	43.6, 3	39.6, 3	31.9, 5
	201	62.8, 2	56.2, 4	52.6, 3	45.5, 6

4.1 Overhead Additions for SRD

In Section 3.1 we had derived an expression for the extra additions using m -th order differences. These additions will be pessimistically taken to each be the width of a product term. Substituting $\lceil N/2 \rceil$ in place of N (symmetric case), the net number of equivalent one-bit overhead additions using SRD follows as:

$$N_A'' = \frac{m}{2}(2\lceil N/2 \rceil - m - 1)(W_c + W_x) \quad (13)$$

The percentage savings in adds and shifts follows as:

$$S_{ADD} = [1 - (N_A' + N_A'')/N_A] * 100 \quad (14)$$

$$S_{SHIFT} = [1 - N_S'/N_S] * 100 \quad (15)$$

5 Simulation Results

The results of simulating four different kinds of linear-phase LTI-FIR systems of 51, 101 and 201 tap lengths is presented below. The simulator was developed in MATLAB and it was used to track the exact number of adds and shifts required for direct form and SRD computations. The metrics are for computation of all $\lceil N/2 \rceil$ product terms with a randomly generated data. The coefficients of these systems and the data were generated as floating point numbers with four different mantissa word-widths of 8, 12, 16 and 24 bits. Table 1 shows the maximum percentage savings in the net number of adds using SRD over DF computation as defined by Eqn. (14) and the order m at which the maximum savings was obtained. Though higher order differences reduce the multiplication complexity the overhead additions required by SRD increase with increasing orders of differences. Thus there is a point at which the

Table 2: Savings in shifts using SRD

FIR Sys.	N	Simulated S_{SHIFT} at opt m			
		8 bits	12 bits	16 bits	24 bits
Low Pass	51	87.10	82.27	82.81	19.35
	101	95.19	94.27	93.33	83.62
	201	98.64	99.29	94.35	93.51
Band Stop	51	89.13	84.83	81.22	74.14
	101	94.19	93.40	92.71	74.16
	201	98.51	89.62	92.86	93.13
Band Pass	51	88.17	85.42	81.63	46.71
	101	94.80	93.03	82.51	74.28
	201	98.52	97.99	97.48	92.78
High Pass	51	86.05	82.09	84.32	20.96
	101	95.56	94.85	84.60	85.15
	201	98.61	99.46	94.82	96.44

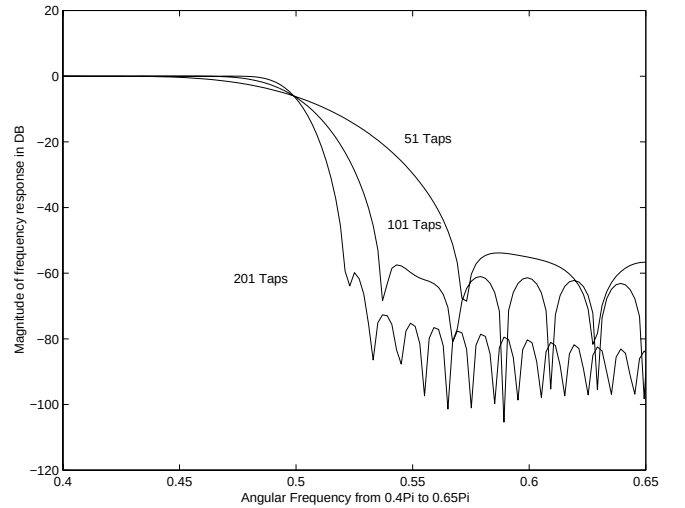


Figure 3: Detail of frequency response for the example low-pass filter in Tables 1 and 2 designed using a Hamming window.

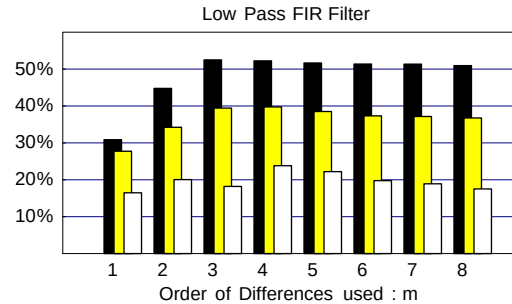


Figure 4: Savings in adds for the same low-pass filter, as a function of the order of differences using 16 bit mantissas. The black bars are for $N = 201$, the grey bars are for $N = 101$, the white bars are for $N = 51$.

overall number of adds is minimized. This is explicitly shown in Fig. 4 which plots the net savings in adds as a function of the order of differences used for our first example which is a low-pass FIR filter. It can be seen that there is a maxima at a different order for each of the three tap lengths. It can also be seen that the maximum savings increases with tap length as was explained in Section 3.3. Table 2 shows the percentage savings in the net number of shifts as defined by Eqn. (15) at the optimum order m at which the net adds is minimized. It is also possible to arrive at an optimum order based on the minimum weighted sum of net adds and shifts.

6 Conclusion

We have presented an improved version of the Differential Coefficients Method called the Sorted Recursive Differences Method. It has been shown that using the SRD method in a transposed direct form FIR structure with a Floating-Point number representation can result in a greatly reduced multiplication complexity as compared to direct form computation. The MATLAB simulation results presented for four practical FIR systems, confirm the theoretical predictions of reduced multiplication complexity. This computational reduction results in a net reduced switching and hence lower power consumption. The statistical analysis used to quantify the computational reduction can be applied to other multiplier models. For example, for an array multiplier a decrease in the number of ones in the multiplier words can be related to a decrease in switching activity within the multiplier and consequently to a lower power consumption.

References

- [1] A. Chatterjee, R. K. Roy and M. A. d'Abreu, "Greedy hardware optimization for linear digital systems using number splitting and repeated factorization", *6th Intl Conf on VLSI Design*, Jan 1993, India, pp. 154-159.
- [2] A. G. Dempster and M. D. Macleod, "Use of minimum-adder multiplier blocks in FIR digital filters", *IEEE Trans on Ckts & Sys II-Analog & Digital Signal Proc*, v 42, n 9, Sep. 1995, pp. 569-577.
- [3] A. P. Chandrakashan and R. Brodersen, "Low power digital CMOS design", *Kluwer Academic Publishers*, 1996.
- [4] D. M. Kodek and K. Steiglitz, "Filter-length word-length trade-offs in FIR digital filter design", *IEEE Trans on Acoustics, Speech & Signal Proc*, v ASSP-28, n 6, Dec 1980, pp. 739-744.
- [5] H. J. Shin et. al., "Custom design of CMOS low-power high-performance digital signal-processing macro for hard-disk-drive applications", *IBM Journal of Research & Development*, v 39, n 1-2, Jan-Mar 1995, pp 83-90.
- [6] H. Samuelli, "An improved search algorithm for the design of multiplierless FIR filters with powers-of-two coefficients", *IEEE Trans on Ckts & Sys*, v 36, n 7, July 1989, pp. 1044-1047.
- [7] J. W. Adams and A. N. Willson Jr., "A new approach to FIR digital filters with fewer multipliers and reduced sensitivity", *IEEE Trans on Ckts & Sys*, v CAS-30, n 5, May 1983, pp. 277-283.
- [8] J. W. Adams and A. N. Willson Jr., "Some efficient digital pre-filter structures", *IEEE Trans on Ckts & Sys*, v CAS-31, n 3, Mar 1984, pp. 260-265.
- [9] L. R. Rabiner and R. E. Crochiere, "A novel implementation for narrow-band FIR digital filters", *IEEE Trans on Acoustics, Speech & Signal Proc*, v ASSP-23, n 5, Oct 1975, pp. 457-464.
- [10] M. A. Soderstrand and K. Al-Marayati, "VLSI implementation of very-high-order FIR filters", *IEEE Intl Symp on Ckts & Sys*, v 2, 1995, pp. 1436-1439.
- [11] M. Mehendale, S.D. Sherlekar and G. Venkatesh, "Coefficient optimization for low power realization of FIR filters", *IEEE Workshop on VLSI Signal Processing*, Japan, 1995.
- [12] M. Yagyu, A. Nishihara and N. Fujii, "Fast FIR digital filter structures using minimal number of adders and its application to filter design", *IEICE Trans Fundamentals*, v E79-A, n 8, Aug 1996, pp. 1120-1128.
- [13] N. Sankarayya, K. Roy and D. Bhattacharya, "Algorithms for low-power and high speed FIR filter realization using differential coefficients", *IEEE Trans on Ckts & Sys II-Analog & Digital Signal Proc*, v 44, n 6, June 1997, pp. 488-497.
- [14] O. Monkewich and W. Steenaart, "Stored product digital filtering with non-linear quantization", *IEEE Intl. Symp on Ckts & Sys*, 1976, pp. 157-160.
- [15] P. W. Wong and R. M. Gray, "FIR filters with sigma-delta modulation encoding", *IEEE Trans on Acoustics, Speech & Signal Proc*, v 38, n 6, June 1990, pp. 979-989.
- [16] P. P. Vaidyanathan, "Efficient and multiplierless design of FIR filters with very sharp cutoff via maximally flat building blocks", *IEEE Trans on Ckts & Sys*, v CAS-32, n 3, Mar 1985, pp. 236-244.
- [17] R. Hartley, "Optimization of CSD multipliers for filter design", *IEEE Intl. Symp on Ckts & Sys*, v 4, 1991, pp. 1992-1995.
- [18] S. R. Powell and P. M. Chau, "Efficient narrow-band FIR and IFIR filters based on powers-of-two sigma-delta coefficient truncation", *IEEE Trans on Ckts & Sys II-Analog & Digital Signal Proc*, v 41, n 8, Aug 1994, pp. 497-505.
- [19] T. Saramaki, "Design of FIR filters as a tapped cascaded interconnection of identical subfilters", *IEEE Trans on Ckts & Sys*, v CAS-34, n 9, Sept 1987, pp. 1011-1029.
- [20] T. Saramaki, Y. Neuvo and S. K. Mitra, "Design of computationally efficient interpolated FIR filters", *IEEE Trans on Ckts & Sys*, v 35, n 1, Jan 1988, pp. 70-87.
- [21] Y. C. Lim and S. R. Parker, "FIR filter design over a discrete powers-of-two coefficient space", *IEEE Trans on Acoustics, Speech & Signal Proc*, v ASSP-31, n 3, June 1983, pp. 583-590.
- [22] Y. C. Lim, "Predictive coding for FIR filter word-length reduction", *IEEE Trans on Ckts & Sys*, v CAS-32, n 4, Apr 1985, pp. 365-371.
- [23] Y. C. Lim, "Frequency-response masking approach for the synthesis of sharp linear phase digital filters", *IEEE Trans on Ckts & Sys*, v CAS-33, n 4, Apr 1986, pp. 357-364.