Low Complexity FIR Filters Using Factorization of Perturbed Coefficients[†]

Cassondra Neau, Khurram Muhammad*, and Kaushik Roy
ECE, Purdue University, West Lafayette, IN 47907
*Texas Instruments, Dallas, TX

Abstract

This paper presents a factorization based technique to reduce the computational complexity of implementing Finite Impulse Response (FIR) digital filters. It is possible to design FIR filters in which all of the filter coefficients are products of the first seven prime numbers. For such filters, factorization of the filter coefficients allows the reuse of intermediate results among computations involving common factors. Since the coefficients are products of only small prime numbers, it is also possible to generate each of the partial products with a single shift and add operation. Compared to a traditional implementation, this results in a 35-50% reduction in computational complexity, which is shown to translate into lower power consumption.

1. Introduction

In an age of portable multi-media devices, there is an ever-increasing demand for higher speeds, greater functionality, and longer battery life. As a result, low-power design techniques for high-performance digital signal processors are essential to further growth in the portable computing area.

Power reduction strategies can be applied to all levels of a system design: New device technology can be applied; Circuit design can be optimized for power consumption; Different architectures could be chosen; Or new algorithms developed [1]. This paper focuses on the last area. Since Finite Impulse Response (FIR) digital filters are an essential component of digital signal processors, optimizing the algorithm employed by these filters, has a significant effect on the power consumption of the system.

Low-complexity design reduces the number of operations required to achieve a desired output. Since each operation takes time and increases power consumption due to increased switching activity, reducing the computational complexity leads to improved data processing speed and lower-power consumption by reducing the number of operations required at the highest level of design abstraction [2]. Since multiplication is the most computationally expensive operation in the FIR filter, simplifying or even removing the multiplication is highly desirable for complexity reduction.

This paper briefly describes a perturbation method to design FIR filters with coefficients that have first seven prime numbers as their only prime factors. By reorganizing the computation of the filter output, it is possible to share the partial products generated by "multiplication" by these factors.

Furthermore, it is possible to generate these partial products without true multiplication. Using the numerical representation and the relationships between the small prime numbers, a single shift and add operation can produce these partial products. These facts combine to produce significant savings in computational complexity and power consumption.

2. Fundamentals

2.1 Frequency response

The output of a FIR filter is described by the following equation:

$$y(n) = \sum_{i=0}^{M-1} C_i x[n-i]$$

C_i represents the i^{th} filter coefficient and M is the length of the filter (the number of taps). This is a linear time-invariant system which is typically implemented with a multiply and accumulate sequence. The coefficients are multiplied by the input data. The sum of these multiplications gives the filter output.

[†] This research was supported by the SRC Master's Scholarship Program.

2.2 Factorization

According to the Fundamental Theorem of Arithmetic [4], each whole number can be represented as a unique product of prime numbers. Hence, the filter coefficients can be written in the following form:

$$C_i = \prod_{k=1}^{Q(n)} \{f_n(k)\}^{p_n(k)}$$

where f_n are prime factors (2, 3, 5, 7, 11, 13…) Q(n) is the cardinality of the set of prime numbers occurring in the coefficient, and the exponent p_n is the the number of times a given factor f_n occurs in the factorization of C_i.

This representation is a useful tool in identifying the common factors amongst the filter coefficients. Once identified, common factors allow a reduced complexity filter implementation as follows.

If $C_i = f_a \times f_b$ and $C_{i+1} = f_a \times f_c$, f_a is a common factor of C_i and C_{i+1}. Therefore the product $f_a \times x[n]$ can be used in calculating both the $C_i \times x[n]$ and $C_{i+1} \times x[n-1]$ terms in equation 1. This type of reuse can occur across all M terms of the filter.

2.3 Numerical representation

An important consideration in any filter implementation is the choice of numerical representation [5]. The amount of computation required in calculating the filter output depends on the numerical representation. Thus this choice impacts the performance and power consumption of the filter as well as the quantization effects. We chose to use a 16-bit sign-magnitude representation. It is a straight-forward representation and shows the benefit of factorization in a direct manner. In applications that process zero mean signals, using sign-magnitude representation is preferable to two's complement representation [6] and Canonical Sign Digit often requires recoding. This work could be extended to other representations. For instance, for floating-point arithmetic, factorization would occur on the mantissa of each coefficient.

3. Perturbation

3.1 Motivation

The problem with factorization is the lack of "good" common factors between the filter coefficients. It is highly unlikely that a filter designed by conventional methods would have an abundance of large common factors across several coefficients. Without a significant number of common factors, there is little benefit to factorization. To remedy this, the coefficients can be perturbed. By adding or subtracting a small amount from a coefficient, completely different sets of factors can be obtained.

Perturbation is very useful in low-complexity design, but there is a tradeoff between compromising the filter response and reducing the computation required by such filters [3]. The filter coefficients cannot be arbitrarily perturbed to optimize factorization. This would result in a poor filter response. Therefore the filter coefficients must be perturbed in a controlled manner such that the frequency response is constrained.

The choice of factors determines the extent of factorization. As the cardinality of the set of factors increases, the probability that a factor will be used multiple times decreases, thereby reducing the potential computational sharing. But, if the set of possible factors is too limiting, the frequency response is significantly altered and the resulting filter is likely not to be within the bounds. As discussed in the next section, the set of the first seven prime numbers is a good compromise.

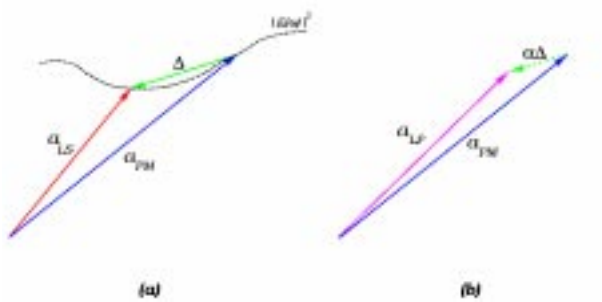

Figure 1. Linear perturbation vector ($0 < \alpha < 1$).

3.2 Method

It has been shown [2] that a Parks-McClellan filter (PM) and a Least Squares filter (LS) with the same pass-bands and stop-bands can be used to generate additional filters with the same critical frequencies.

Let the coefficients of these filters be represented by $\mathbf{a}_{PM}$ and $\mathbf{a}_{LS}$, respectively. Since the PM filter may require fewer taps than the LS for a given frequency response, we pad $\mathbf{a}_{PM}$ with zeros on both sides such that $\mathbf{a}_{PM}$ and $\mathbf{a}_{LS}$ are of identical size. Next let us define an error vector Δ as

$$\Delta = \mathbf{a}_{LS} - \mathbf{a}_{PM}$$

This interpretation is shown in figure 1. In figure 1(a) we have shown the vectors $\mathbf{a}_{PM}$ and $\mathbf{a}_{LS}$ in relationship to a hypothetical error surface. Because $\mathbf{a}_{LS}$ corresponds to the least squares solution, this vector is shown as the global optimum of the surface $|E(\omega)|^2$. The vector $\mathbf{a}_{PM}$ is in the vicinity of $\mathbf{a}_{LS}$ and can be reached through Δ [2].

The final filter is generated from a perturbation of the Linearly Perturbed filter. In figure 1(b) we show the linearly perturbed vector

$$\mathbf{a}_{LP} = \mathbf{a}_{PM} + \alpha\Delta$$

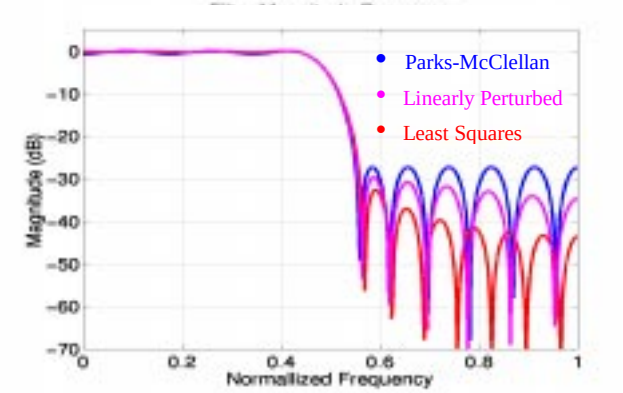

Figure 2. Sample low-pass filter frequency response (α=0.3).

where $0 < \alpha < 1$. The value $\alpha = 0$ corresponds to $\mathbf{a}_{PM}$ and $\alpha = 1$ corresponds to $\mathbf{a}_{LS}$.

Figure 2 shows the frequency response of a low pass filter obtained in this manner. The frequency characteristics of the linearly perturbed vector $\mathbf{a}_{LP}$ lie between the characteristics of $\mathbf{a}_{PM}$ and $\mathbf{a}_{LS}$. By varying α in the range (0,1) the filter characteristics vary between the characteristics of $\mathbf{a}_{PM}$ and $\mathbf{a}_{LS}$. Thus the frequency characteristics of the linearly perturbed vector lie within the extremes provided by the PM and LS responses [2].

This linear perturbation preserves the symmetry of the original filter coefficients around the center tap. This coefficient symmetry is exploited by sharing the multiplications between symmetric taps (as shown in section 4) and guarantees that the final filter will have an exact linear phase response, which is important in some applications [7].

Using $\mathbf{a}_{LP}$ as a starting point, each coefficient is perturbed until it is a product of only the first few prime numbers. For each element in $\mathbf{a}_{LP}$, there are several possible values of $a_i + \delta$ which correspond to such a factorization. Working in order of increasing sensitivity, each coefficient is perturbed to the value that results in the greatest suppression in the stop band while leaving the pass-band within the bounds. After perturbation is completed, every coefficient in the new filter is the product of a small set of prime numbers.

The choice of a Parks-McClellan and a Least Squares filter as a starting point is somewhat arbitrary. The LS filter requires significantly more taps than the PM when sharp transition bands are required. This results in more terms in the filter response equation and therefore more computation. This computation is easily avoided by starting with two Parks-Mclellan filters. Both filters should have the same pass-band and stop-band edges, but one has about 10% more taps, used to give greater stop-band attenuation. (The number of additional taps determines the size of the potential solution space. If more taps are used, the filter coefficients can be perturbed further without

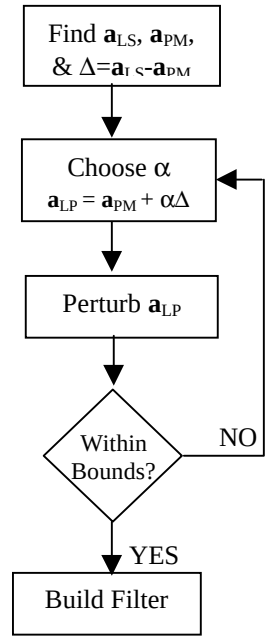

Figure 3. Block diagram of the design cycle

violating the frequency response constraints and therefore can be generated from fewer prime factors. But longer filters have more multiplications and additions. The choice of 10% more taps is related to the choice of 7 prime factors). The same method is applied to find an intermediate filter which is perturbed to achieve a good factorization. Once again, original filter symmetry is preserved.

In addition, if certain pass-band and stop-band criteria are met, it is possible to design a PM FIR filter in which half of the coefficients are zeros. This results in 50% less computation than a filter of same length without this property. Applying this factorization method preserves this property and achieves additional computational savings. An example of such a filter is filter 3 in Table 1.

4. Synthesis

The multiply and accumulate sequence of a FIR filter is typically implemented in either the direct form or the transposed direct form. The difference between the two structures is that in the transposed direct form all of the multiplications involving a particular input data value occur before any multiplication involving the next data value. This is helpful for sharing the partial products. And since we are using symmetric (linear phase) FIR filters, $C_i = C_{n-i}$. Thus a folded structure will result in half as many multiplications as the non-folded structure. Figure 4 shows the folded, transposed direct form for a 6 tap FIR filter.

Our method will use a modification of a folded transposed direct form. All of the multiplication blocks are

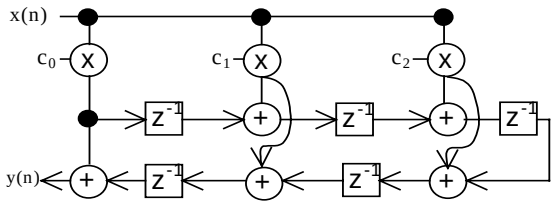

Figure 4. Transposed direct form of a 6 tap FIR filter.

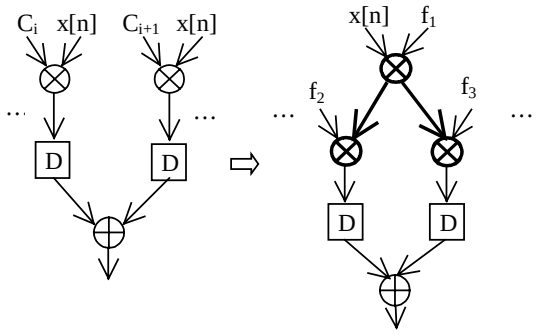

Figure 5. First two steps in data flow graph transformation. ($C_i = f_1f_2$ and $C_{i+1} = f_1f_3$). The factorization tree is represented by the bold portion.

removed and replaced with a network of shifts and additions. This network will collectively be referred to as the factorization tree.

Figure 5 shows the initial transformation of the data flow graph for a sample FIR filter with common factor f_1. The first step in the transformation is replacing the coefficient multiplication with multiplication by the coefficient factors. This step is followed by common subexpression elimination wherever common factors can be shared. Instead of M/2 multiplications involving multipliers with absolute values up to 2^{15}, we are now left with greater than M/2 multiplications involving a small set of possible multipliers.

The next optimization step, operator strength reduction, exploits the properties of binary arithmetic and the small prime factor multipliers. There are three possible node types in the factorization tree as shown in figure 6.

The first possibility is a multiplication by 2. These nodes can be replaced by a simple shift operation. The next case is multiplication by 3, 5, or 17. Each of these factors has a total of 2 1's in binary form (011, 101, and 1001, respectively). Multiplication by one of these factors is easily replaced by one shift and one addition operation. Case 3 involves the other possible factors (7, 11, and 13). Since these factors have a total of 3 1's in binary form (111, 1011, and 1101, respectively), it would be possible to replace these nodes with two shift and two addition operations. But, there is a less costly alternative. We can use the results of a type 2 multiplication to generate the

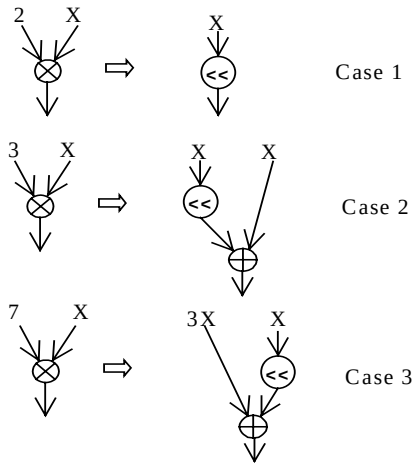

Figure 6. Operator Strength reduction

type 3 multiplication with a single additional shift and addition. For example, since 7x = 3x + 4x, we can use the already produced result of a multiplication by 3 and add a shifted value of the partial product (4x = two shifts).

Another reduction technique is combining multiple nodes in the factorization tree. Some prime factors can be combined into larger common factors that are computationally less expensive. For instance 33=3*11 and requires only one addition. Combination of nodes is beneficial whenever such a combination does not reduce the amount of sharing in the factorization tree.

At the conclusion of operator strength reduction, all of the multiplication in the original filter implementation has been replace by a lower-cost shift and add network. The computational savings and the effect on power consumption are shown in the next section.

5. Results and conclusions

In calculating the computational savings, we are considering all operations in terms of the equivalent number of additions. The assumption in these calculations is that the cost of multiplication is equal to the number of additions in a shift and add multiplier. This method was implemented on different filter types (low-pass, band-pass, and high-pass) and various sizes (20-200 taps) resulting in computational savings of 35-50%. Filters generated with this method can have as few as 1 add/tap.

But how does this reduction in computation translate in terms of power consumption? This question is particularly important because multiplication is seldom accomplished as a series of shifts and additions. To verify the hypothesized reduction in power consumption, a conventional 8 tap low-pass FIR filter was compared to a 10 tap multiplier-less filter with the same frequency response. This filter showed a 48% reduction in computation using factorization of perturbed coefficients.

Table 1. Computational Savings Using Multiplier-less FIR

	Type	Pass-band	Stop-band	Taps1	Taps2	Savings
1	LPF	0.0-0.3	0.6-1.0	25	29	47%
2	LPF	0.0-0.4	0.5-1.0	51	57	42%
3	BPF	0.4-0.6	0.0-0.25, 0.75-1	51	57	35%
4	LPF	0.0-0.5	0.55-1.0	100	110	41%
5	LPF	0.0-0.5	0.52-1.0	170	200	38%

Table 2. Synopsys/Powermill Results for 8/10 Tap Filter (with a 48% reduction in computational complexity using factorization)

	Power	Delay
w/ Multipliers	4.74 mW	30.20 ns
w/ Factor Trees	2.94 mW	28.11 ns

Both filters were synthesized in Synopsys and the power consumption at a nominal frequency was measured by Powermill. The average power consumption was determined using random input vectors. These results are shown in table 2. The simulated power savings was 38% and there was a 7% decrease in delay. The reduction in power consumption is significant, but not as large as the decrease in computation. This is expected due to the storage overhead of this method.

In summary, it has been shown that an arbitrary FIR filter can implemented as a multiplier-less filter using perturbation and factorization. This implementation will require 35-50% less computation and this computational savings will translate into lower power consumption.

6. References

[1] Sankarayya, N., Roy, K., and Bhattacharya, D. "Algorithms for Low-Power and High-Speed FIR Filter Realization Using Differential Coefficients". *IEEE Trans. on Circuits and Systems*, Vol. 44, No. 6, pp. 488-497, Jun. 1997.

[2] Muhammad, K. "Algorithmic and Architectural Techniques for Low Power Digital Signal Processing", *Ph.D. thesis*, Purdue University, 1999.

[3] Mehendale, M., Sherlekar, S.D., and Venkatesh, G. "Coefficient Optimization for Low-Power Realization of FIR Filters", *IEEE Workshop on VLSI Signal Processing*, Japan, 1995.

[4] Graham, R., Knuth, D. and Potashnik, D. *Concrete Mathematics*, Addison-Wesley, 1994.

[5] Hartley, R. "Optimization of CSD Multipliers for Filter Design," *IEEE Intl. Symposium on Circuits & Systems*, Vol. 4, 1991. pp. 1992-1995.

[6] Azadet, K. and A.J. Nicole,"Low-Power Equilizer Architectures for High-Speed Modems," *IEEE Communications Magazine*, pp 118-126, Oct. 1998.

[7] Hawley, R., B. Wong, T. Lin, J. Laskowski, and H. Samueli. "Design Techniques for Silicon Compiler Implementations of High-Speed FIR Digital Filters," *IEEE J. of Solid State Circuits*, Vol 31, No. 5, May 1996, pp. 656-667.