Algorithmic and Architectural Transformations for Low Power Realization of FIR Filters

Mahesh Mehendale

Texas Instruments (India) Ltd.
Golf View Homes, Wind Tunnel Road,
Bangalore 560017, India

S.D. Sherlekar, G. Venkatesh

Dept. of Computer Sc. and Engg.
Indian Institute of Technology, Bombay,
Powai, Mumbai, 400076, India

Abstract

We present algorithmic and architectural transforms for low power realization of Finite Impulse Response (FIR) filters implemented both in software on programmable DSPs and as hardwired macros. For the programmable DSP based implementation, these transform address power reduction in the program memory address and data busses and also the multiplier. We also propose architectural extensions to support some of these transformations. The transforms for hardwired FIR filters aim at reducing the supply voltage while maintaining the throughput. We also present transforms that reduce the computational complexity of the FIR filter computation and thus achieve power reduction.

1 Introduction

FIR filtering is achieved by convolving the input data samples with the desired unit impulse response of the filter. The output Y_n of an N-tap FIR filter is given by the weighted sum of latest N input data samples.

$Y_n = \sum_{i=0}^{N-1} A_i \times Xn - i \qquad \ldots I$

The weights (A_i) in the expression are the filter coefficients. The number of taps (N) and the coefficient values are derived so as to satisfy the desired filter response in terms of passband ripple and stopband attenuation. In this paper we present algorithmic and architectural transforms for low power, targetted to both the software and the hardware realization of FIR filters.

The sources of power dissipation in CMOS circuits can be classified as dynamic power, short circuit currents and leakage. For most CMOS designs dynamic power is the main source of power dissipation and is given by

$P_{dynamic} = C_{switch}.V^2.f \qquad \ldots II$

where V is the supply voltage, f is the operating frequency, C_{switch} is the switching capacitance given by the product of the physical capacitance being charged/discharged and the corresponding switching probability. The low power transformations described in this paper aim at reducing one or more of these factors while maintaining the throughput.

The paper is organized as follows. Section 2 presents FIR filter implementation on a programmable DSP, identifies the measures of power dissipation and presents various techniques for power reduction. In section 3 we present transformations for low power realization of hardwired FIR filters. Finally we present conclusion in section 4 and highlight few other transforms which can potentially result in low power FIR filters.

2 Low Power Realization of FIR Filters on Programmable DSPs

As can be seen from equation I, multiply-accumulate is the key computation at the core of FIR filtering. Datapaths of most programmable DSPs hence have a hardwired multiplier and can implement repeated single cycle multiply-accumulate operations. Programmable DSPs also have Harvard type architecture[3] characterized by separate program and data memory spaces that can be accessed simultaneously.

Figure 1 shows a suitable abstraction of the datapath of TMS320C5x DSP[4]. It consists of a 16 bit fixed point multiplier and a 32 bit ALU. It supports a MAC instruction that simultaneously accesses a coefficient from the program memory and a data from the data memory and performs multiply-accumulate operation. MAC instruction when repeated is executed in a single cycle. FIR filtering is thus implemented on this DSP as a sequence of multiply-accumulate operations (RPT - MAC).

2.1 Measures of Power Dissipation

Each step in the FIR filtering algorithm involves getting the appropriate coefficient and data values and

1063-9667/97 $10.00 © 1997 IEEE

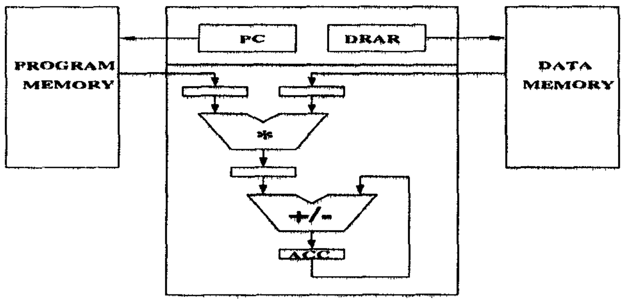

Figure 1: Suitable Abstraction of TMS320C5x Architecture for FIR Filtering

performing a multiply-accumulate computation. Thus address and data busses of both the memories and the multiplier-ALU datapath experience the highest activity during FIR filtering. These hardware components hence form the main sources of power dissipation.

For a typical embedded processor, address and data busses are networks with a large capacitive loading [5]. Hence signal switching in these networks has a significant impact on power consumption. In addition to the capacitance of each signal, intersignal capacitance also contributes to bus power dissipation. The power dissipation due to intersignal capacitance varies depending on the adjacent signal transitions. Power analysis of TMS320C5x[6] shows that the current required for signals to switch between 5's (0101b) and A's (1010b) is about 25% more than the current required for the signals to switch between 0's (0000b) and F's (1111b).

The Hamming distance between consecutive signal values and the number of adjacent signals toggling in opposite direction thus form measures of bus power dissipation.

Various architectures of hardwired parallel multipliers have been presented in the literature[7,8]. These include array multipliers using carry-save adders, Booth multipliers and Wallace tree based multipliers. While the multiplier power depends on the input signal values and also the number of transitions on its inputs, the exact nature of dependency varies across the multiplier architectures. Analysis[9] for an array multiplier (figure 2(a)) shows that the power dissipation is directly related to the transition densities[10] and the probabilities of the multiplier inputs being 1. Further analysis also shows that the signal transitions in LSBs result in a relatively higher power dissipation than the signal transitions in MSBs.

Figure 2(b) shows a micro-architecture model for the Booth multiplier. Power analysis of the multiplier[11] shows that the power dissipation is directly dependent on the number of 1s in the booth-encoded

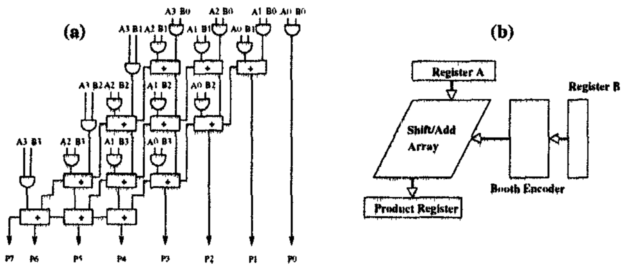

Figure 2: (a) 4x4 Array Multiplier, (b) Microarchitecture model for the Booth Multiplier

input.

2.2 Techniques for Power Reduction

We first present techniques that can be applied on the existing DSP architectures (such as TMS320C5x) to minimize one or more of the power components discussed above.

1. Coefficient Scaling

For an N-tap filter with N coefficients the output Y_n is given by equation I. Scaling the output preserves the filter characteristics in terms of passband ripple and stopband attenuation, but results in an overall magnitude gain equal to the scale factor. For a scale factor K, from equation I we get

$K \times Y_n = K \times \sum_{i=0}^{N}(A_i \times X_{n-i})$

$= \sum_{i=0}^{N}((K \times A_i) \times X_{n-i}) \qquad \ldots III$

Thus the coefficients of the scaled filter are given by $(K \times A_i)$. Given the allowable range of scaling (e.g. ±3db), an optimal scaling factor K can be found such that the total number of ones in the binary representations of $(K \times A_i)$ coefficients is less than the total number of ones in the binary representation of A_i coefficients. This can save the power in an array multiplier. In case of the Booth multiplier, the scaling can performed in such a way that the total number of ones in the Booth-encoding of $(K \times A_i)$ coefficients is less than the the number of ones in the Booth encoding of A_i coefficients.

Since the scaling changes the bit pattern of the coefficients, it also affects the Hamming distance between the consecutive coefficients. It can thus also be used as a technique to reduce Hamming distance related multiplier power and also the coefficient data bus power.

We have presented results in [9] for 10 low pass FIR filters, which show that using the coefficient scaling technique the total number of ones in the 2's complement binary representation of the coefficients can be reduced by 8% to 24%.

2. Coefficient Optimization

Given the filter requirements in terms of passband ripple, stopband attenuation and the order of the filter, multiple coefficient sets can be derived so as to satisfy these requirements. These coefficient sets are typically in a close vicinity in terms of coefficient values. However a small change in coefficient value (e.g. 31 00011111b to 32 00100000b) can have significant impact on the bit pattern and hence the power dissipation.

In [12] we have presented an algorithm that solves the following problem. Given an N-tap FIR filter with coefficients $A_i, i = 0, N-1$ that satisfy the filter response in terms of passband ripple, stopband attenuation and linear phase, find a new set of coefficients $LA_i, i = 0, N-1$ such that the total Hamming distance between successive coefficients is minimized while still satisfying the desired filter characteristics in terms passband ripple and stopband attenuation. Also retain the linear phase characteristics if such an additional constraint is specified.

The results for 6 low pass filters show that the combination of scaling and coefficient optimization with no linear phase constraint can achieve upto 36% reduction in the total Hamming distance and upto 88% reduction the total number of adjacent signal transitions in opposite directions.

We now present techniques that result in power reduction but require additional architectural support.

3. **Selective Negation**

The FIR coefficients are stored in the coefficient memory in 2's complement form. For a given number N and the number of bits B used to represent it, the number of 1s in the 2's complement representation of +N and -N can differ significantly. For each coefficient A_i, either A_i or $-A_i$ can be stored in the coefficient memory, depending on the value that has lesser number of 1s in its 2's complement binary representation. If $-A_i$ is stored in the memory, the corresponding product $(A_i \times X_{n-i})$ needs to be subtracted from the accumulator so as to get the correct Y_n result. This technique of selective coefficient negation can result in significant reduction in multiplier power.

Since selective coefficient negation impacts the bit representation of the coefficients, it also affects the Hamming distance between the consecutive coefficients. It can thus also be used as a technique to reduce Hamming distance related multiplier power and also coefficient data bus power.

The results presented in [9] for 10 low pass filters show that for low pass filters selective negation selectes 37% to 51% of coefficients for negation, and results in 26% to 56% reduction in the total number of ones. It also results in 13% to 50% reduction in the total Hamming distance between consecutive coefficients.

4. **Coefficient Ordering**

Since the summation operation is both commutative and associative, the filter output is independent of the order of computing the coefficient products. Thus for a 4 tap filter, the output can be computed as
$Y_n = A_0X_n + A_1X_{n-1} + A_2X_{n-2} + A_3X_{n-3}$ or as
$Y_n = A_1X_{n-1} + A_3X_{n-3} + A_0X_n + A_2X_{n-2}$
The order of coefficient-data product computation directly affects the sequence of coefficients appearing on the coefficient memory data bus and hence the power dissipation on the data bus.

The problem of finding an optimum order of computation so as to minimize the total Hamming distance beween consecutive coefficients, can be formulated as a Travelling Salesman problem, with the coefficients representing the cities and the Hamming distance beween the coefficients representing the distance between the cities. Since Travelling Salesman problem is NP complete, many heuristics have been proposed to get near optimal solution in polynomial time. For the various low pass filter coefficients that we experimented with, the heuristic based on the nearest-neighbour selection worked the best, resulting in the reduction of 54% to 83% in the total Hamming distance.

5. **Gray Coded Addressing**

The power dissipation in the address bus of the coefficient memory is dependant on the Hamming distance between consecutive memory addresses. Given the order of accessing the filter coefficients, their locations can be decided so as to minimize this Hamming distance. Gray coded addressing[9,5] results in least address bus power dissipation for sequential access. In this scheme the Hamming distance between any two consecutive addresses is one and consequently the number of adjacent signals toggling in opposite direction is zero. Such an addressing scheme can result in 50% power reduction compared to binary addressing of sequential data.

6. **Coefficient-Data Swapping**

As mentioned earlier the power dissipation in case of the Booth multiplier is dependent on the number of 1s in the Booth-encoded input. Since the multiplication operation is commutative, the input to be Booth-encoded can be either the data or the coefficient during each MAC operation of the filter. Since it is difficult to decide on the coefficient-data swapping at run-time (the power dissipated in the decision making logic can be more than the saving), the swapping decisions need to made statically based on the coefficients alone. The

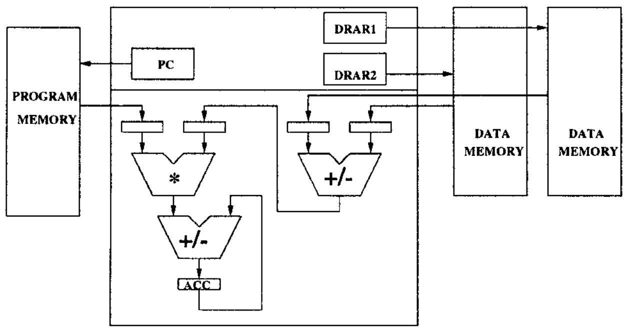

Figure 3: Suitable Abstraction of TMS320C54x Architecture for Linear Phase FIR Filtering

approach suggested in [11] decides to Booth-encode data (i.e. swap the multiplier inputs) for those coefficients whose Booth-encoded weights are above a certain limit. The experimental results shown in [11] indicate that such a technique does result in significant power saving in many cases.

7. **Datapath to Support Linear Phase Filters**
The coefficient symmetry of linear phase filters can be used to reduce to half the number of multiplications per output computation. For N even, equation I can be written as:

$$Y_n = \sum_{i=0}^{N/2-1} A_i \times (X(n-i) + X(n-N+i+1)) \quad \ldots IV$$

While the core computation in equation IV is also Multiply-accumulate, the coefficient is multiplied with the sum of two input samples. The architectures such as shown in figure 1, do not support single cycle execution of this computation. While it is possible to compute data sum and use it to perform MAC, the resultant code would require more number of cycles and more number of data memory accesses than the direct implementation of equation I ignoring the coefficient symmetry.

Figure 3 shows a suitable abstraction of the datapath of TMS320C54X processor that supports single-cycle execution (FIRS instruction) of the multiply-accumulate computation of equation IV.

This architecture has one more data read bus, which enables fetching the coefficient and the two data values in a single cycle. It's datapath has an adder and a MAC unit, so that the sum of the input data samples and the multiply-accumulate operation can be performed simultaneously in a single cycle. Since the computational complexity of equation IV is lesser than that of equation I, the corresponding implementation of equation IV is significantly more power efficient.

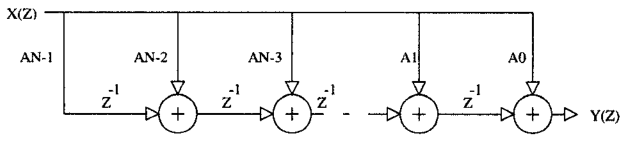

Figure 4: Signal Flow Graph of the Transposed FIR Filter

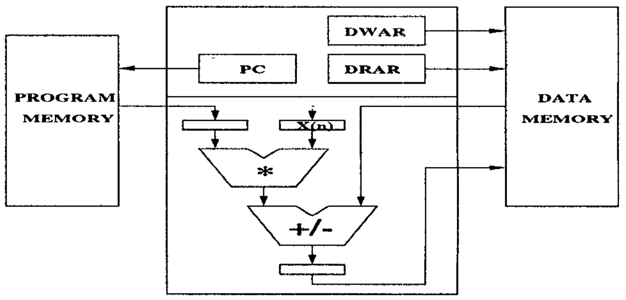

Figure 5: Architecture to Support Efficient Implementation of Transposed FIR Filter

8. **Implementing Transposed FIR Structure**
The signal flow graph represented by equation I can be transformed using the transposition theorem[2] into a signal flow graph shown in figure 4. While the computational comlexity of the transposed structure is same as the direct form structure (equation I), it involves multiplying all the coefficients by the same input data (X(n)). Thus throughout the filter computation, one of the multiplier inputs is fixed, resulting in significant power saving in the multiplier compared to the direct form implementation[1]. While the transposed form does not require movement of old data samples, it needs to store the results of the intermediate computations for future use. Since the intermediate results have higher precision (precision of the coefficient + precision of the data), wider busses are required to support single-cycle multiply-add operations.

Figure 5 proposes an architecture to support N cycle implementation of N tap transposed filter. The increased capacitance due to the wider read/write busses and an extra write performed every cycle result in increased power dissipation compared to the direct form implementation. This increase typically outweighs the saving in the multiplier power, thus making such an architecture overall less power efficient.

3 Techniques for Power Reduction in Hardwired FIR Filters

If the throughput requirement of an application is higher than the available MIPs, certain components of the application (such as FIR filters) need to be implemented as hardwired macros. Also in case of applications where the flexibility of the programmable processor is not required, hardwired implementation is the preferred choice as such an implementation typically results in higher throughput and lower power. Hardwired implementation also enables applying certain architectural transforms which reduce the computational complexity but also reduce the regularity of the computation (which makes it unsuitable for implementation on a programmable processor). In this section we present architectural transforms for reducing power dissipation of hardwired FIR filters.

1. **Parallel Processing and Pipelining**

These techniques have been presented in the literature[13] as generic transforms for power reduction and are equally applicable to the implementation of FIR Filters[14]. In case of parallel processing, the datapath is replicated so as to perform multiple computations simultaneously. For example using two multiplier-adder units, the filter output can be computed in half the number of cycles compared to the implementation using a single multiplier-adder unit. If the same throughput is to be maintained, the two unit implementation can run at half the frequency of single unit implementation. Since the allowable delay for the multiply-add computation is doubled, the logic in the two unit implementation can run at a lower supply voltage. Since both the units are active each cycle the capacitance switched per cycle doubles. The increase in the capacitance is offset by the reduction in the frequency, and the reduction in the supply voltage result in power reduction proportional to the square of the reduction in the supply voltage.

Pipelining is another architectural transform that increases throughput (by increasing the operating frequency) while increasing the latency. For applications such as FIR filters which process the input data continuously at a constant rate, this tradeoff is very much acceptable. If the goal is to maintain the same throughput, the pipelined filter can operate at the same frequency but at a lower supply voltage resulting in power reduction proportional to the square of the reduction in the supply voltage.

2. **Block FIR Filters**

While the parallel processing and pipelining transformation do not impact the computational complexity, block FIR filters achieve power reduction by reducing the computational complexity. This is achieved by performing transformations on the direct form state space structure[15]. The reduction in the number of multiplications is accomplished at the expense of increased number of additions. Since the area and the

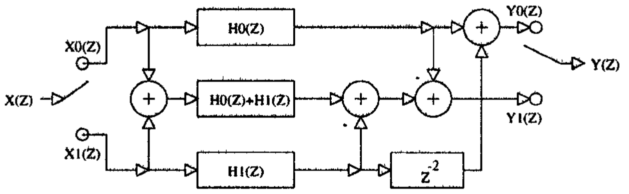

Figure 6: Multirate Architecture with Decimation Factor of 2

delay of the adder is less than the multiplier, the resultant structure results in lesser switching capacitance and can also operate at lower supply voltage while maintaining the same throughput, thus resulting in significant power reduction.

3. **Multirate Architectures**

Multirate architectures are another transforms for reducing the computational complexity of FIR filters. Multirate FIR structures involve implementing the filter in terms of its decimated sub-filters[16]. These structures can be derived using Winograd's algorithms for reducing computational complexity of polynomial multiplications[17]. Figure 6 shows a multirate architecture that uses the decimation factor of two. This architecture takes in two inputs and computes two filter outputs. For an N tap filter, the direct form FIR structure requires N multiplications and N-1 additions per output. The multirate architecture shown in figure 6 requires 3N/4 multiplications and (3N+2)/4 additions per output. This reduced computational complexity enables reduced switching capacitance and lower supply voltage, thus resulting in significant power reduction[17].

Since the multirate architecture partially retains the regularity (in terms of direct form decimated sub-filters), they can also be considered for implementation on a programmable processor. The filter implementation presented in [17] shows that on TMS320C5x the multirate architecture results in power reduction for FIR filters with more than 14 taps. This power reduction however comes at the expense of increased code size and increased data memory requirement.

4. **FIR Filter Realization using Differential Coefficients**

Differential Coefficients Method[18] is another approach to reducing computational complexity and hence the power dissipation of FIR filters. This is achieved by using various orders of differences between the coefficients in conjuction with stored precomputed results rather than using the coefficients directly. The results in [18] indicate that for certain type of FIR filters this technique enables significant power reduction.

5. **Using efficient Pre-Filter Structures**
Instead of realizing the filter in the direct form as shown in equation I, it can be implemented as a cascade of a "pre-filter" with an "equalizer". The pre-filter structures are computationally efficient as they use coefficients with values 1 or -1. Use of Cyclotomic Polynomial Filters as pre-filter structures has been proposed in [20]. With the correct choice of a pre-filter structure, the equalizer filter can be implemented with fewer number of taps than required for the direct form realization[21]. The cascade of "pre-filter"-"equalizer" thus requires fewer number of multiplications and hence is power efficient.

4 Conclusion

This paper presents algorithmic and architectural transforms for low power realization of FIR filters implemented both on the programmable processors and also as hardwired macros. While we have covered most of the important transformations, there are still quite a few approaches that can potentially reduce power dissipation. These include distributed arithmetic based implementations, implementation using Residue Number System based arithmetic[19] and also other multiplier-less implementation approaches. All these transforms need to be captured in a framework which will allow quantitative evaluation of various alternatives and suggest the combination of transformations to achieve maximum power reduction under the specified throughput and area constraints.

References

[1] R. Mehra, D. B. Lidsky, A Abnous, P. E. Landman and J. M. Rabaey, "Algorithms and Architectural Level Methodologies for Low Power", Chapter 11 in *Low Power Design Methodologies*, Jan Rabaey and Massoud Pedram, Eds., Kluwer Academic Publishers, September 1995

[2] A.V. Oppenheim and R.W. Schaffer, *Discrete Time Signal Processing* , Prentice Hall, 1989

[3] Edward A. Lee, "Programmable DSP Architectures: Part I", IEEE ASSP Magazine, October 1988, pp 4-19

[4] TMS320C5x User's Guide, Texas Instruments, 1993

[5] Ching-Long Su, Chi-Ying Tsui and Alvin M. Despain, "Saving Power in the Control Path of Embedded Processors", IEEE Design and Test of Computers, Winter 1994, pp 24-30

[6] Jon Bradley, "Calculation of TMS320C5x Power Dissipation Application Report", Texas Instruments, 1993

[7] Gin-Kou Ma and Fred J. Taylor, "Multiplier Policies For Digital Signal Processing", IEEE ASSP Magazine, January 1990, pp 6-19

[8] Earl Swartzlander Jr.,"VLSI Signal Processing Systems", Kluwer Academic Publishers, 1985

[9] Mahesh Mehendale, G. Venkatesh and S.D. Sherlekar, "Techniques for Low Power Realization of FIR Filters", ASP-DAC'95

[10] Farid Najm, "Transition Density: A New Measure of Activity in Digital Circuits", IEEE Transactions on CAD, Feb 1993, pp 310-323

[11] V. Tiwari and S. Malik, "Power Analysis and Minimization Techniques for Embedded DSP Software", IEEE Transactions on VLSI Systems, March 1997

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

[13] A. P. Chandrakasan and R. W. Brodersen, "Minimizing Power Consumption in Digital CMOS Circuits", Proceedings of the IEEE, April 1995, pp 498-523

[14] K. K. Parhi, "Algorithms and Architectures for High-Speed and Low-Power Digital Signal Processing", Proceedings of 4th International Conference on Advances in Communications and Control, Rhodes, Greece, June 1993

[15] D. N. Pearson and K. K. Parhi, "Low-Power FIR Digital Filter Architectures", IEEE International Symposium on Circuits and Systems, ISCAS-1995

[16] Z-J Mou and P. Duhamel, "Short-Length FIR Filters and Their Use in Fast Nonrecursive Filtering", IEEE Transactions on Signal Processing, June 1991, pp 1322-1332

[17] M. Mehendale, S. D. Sherlekar, G. Venkatesh, "Low Power Realization of FIR Filters using Multirate Architectures", International Conference on VLSI Design, 1996, pp 370-375

[18] N. Sankarayya and K. Roy, "Algorithms for Low Power FIR Filter Realization Using Differential Coefficients", International Conference on VLSI Design, 1997, pp 174-178

[19] M. A. Soderstrand and K. Al-Marayati, "VLSI Implementation of Very-High- Order FIR Filters", IEEE International Symposium on Circuits and Systems, 1995, pp 1436-1439

[20] W. J. Oh and Y. H. Lee, "Cascade/Parallel Form FIR Filters with Powers-of- Two Coefficients", IEEE International Symposium on Circuits and Systems, ISCAS-1994, Vol. II, pp 545-548

[21] J. W. Adams and A. N. Willson, "Some Efficient Digital Prefilter Structures", IEEE Transactions on Circuits and Systems, May 1984, pp 260-265