IMPLEMENTATION OF LOW POWER AND HIGH-SPEED HIGHER ORDER CHANNEL FILTERS FOR SOFTWARE RADIO RECEIVERS

A. P. Vinod[+], E. M-K. Lai[*] and S. Emmanuel[@]
School of Computer Engineering, Nanyang Technological University
Singapore 639798
Email: {asvinod[+], asmklai[*], asemmanuel[@]}@ntu.edu.sg

ABSTRACT

The most computationally intensive part of the wideband receiver of a software defined radio (SDR) is the channelizer since it operates at the highest sampling rate. Higher order FIR channel filters are needed in the channelizer to meet the stringent adjacent channel attenuation specifications of wireless communications standards. In this paper, we present a coefficient-partitioning algorithm for realizing low power and high-speed channel filters. Design examples of the channel filters employed in the Digital Advanced Mobile Phone System (D-AMPS) and Personal Digital Cellular (PDC) receivers show that the average reductions of memory and power consumption achieved using our method over existing method are 25% and 50% respectively.

I. INTRODUCTION

The use of SDR technology is predicted to replace many of the traditional methods of implementing transmitters and receivers while offering a wide range of advantages including adaptability, reconfigurability, and multifunctionality encompassing modes of operation, radio frequency bands, air interfaces, and waveforms [1]. The most computationally intensive part of the digital front-end of an SDR receiver is the channelizer since it operates at the highest sampling rate [2]. Channelization in SDR receivers involves the extraction of multiple narrowband channels from a wideband signal using several bandpass filters called channel filters [3]. The channel filter must have very narrow transition band and considerable stopband attenuation to meet the stringent adjacent channel attenuation requirements of wireless communications standards. Therefore, FIR filters with large number of taps (typically 200 to 1200 taps) are employed in the channelizer. Furthermore, the channel filters must have low power consumption and high-speed [4]. The key functional units in a digital filter are delay, adder, and multiplier – out of which multiplier dominates the hardware complexity. The complexity of the FIR multiplier is dominated by the number of adders (subtractors) employed in the coefficient multipliers. Among the approaches for reducing the number of adders in the multipliers of FIR filters, the common subexpression elimination (CSE) techniques in [5, 6] produced the best hardware reduction since it deals with multiplication of one variable (input signal) with several constants (coefficients). However, the CSE methods in [5, 6] have not addressed the issue of minimizing the complexity of each adder of the multiplier, which is significant in low power and high-speed implementations. Moreover, most of the FIR filter optimization methods proposed in literature addressed lower order filters (less than 300 taps) and no attempt was given to realize filters with very large number of taps (up to 1200 taps). In our recent work [7], we have analyzed the complexity of implementation of FIR filters in terms of the number of full adders (FAs) required for each multiplier. A method for optimizing the CSE method in [5] to implement higher order channel filters for wideband receivers with less complexity was also proposed in [7]. This technique is based on the extension of conventional two-nonzero bit (2-bit) common subexpressions (CS) in [5] to form three-nonzero bit and four-nonzero bit super-subexpressions (called 3-bit and 4-bit SS, respectively) by exploiting identical shifts between a 2-bit bit CS and a third nonzero bit, or between two 2-bit CS. Since employing SS reduces the number of adders, the number of FAs is also reduced correspondingly – this is the basic approach adopted in [7]. The main limitation of the method in [7] is its dependence on the statistical distribution of shifts between the 2-bit CS in the canonic signed digit (CSD) representations of FIR filter coefficients. Moreover, the routing complexity of the filters designed using the method in [7] is higher than that of the 2-bit CSE techniques in [5] as the former method has more number of subexpressions.

In this paper, we show the implementation of higher order channel filters for SDR receivers with minimum multiplier complexity, memory and power consumption. We combine three techniques: the coefficient-partitioning algorithm, the pseudo floating-point representation and the CSE, to reduce the complexity of channel filters. Our method does not employ the super-subexpressions used in [7] and hence it does not have the dependence on statistical distribution of shifts between the 2-bit CS.

The paper is organized as follows. Section II provides a brief review of FIR filter coefficient multiplier complexity. In Section III, we present our coefficient-partitioning method for realizing low power and high-speed channel filters. In section IV, we illustrate the implementation of channel filters for the D-AMPS and the PDC standards using our CP technique and provide comparisons with the CSE method. Section V provides our conclusions.

II. REVIEW OF FIR FILTER COMPLEXITY

The number of FAs needed to realize the adders used in the multipliers determines the cost of the coefficient multiplier. For completeness, we provide a review of the complexity of multiplier block implementation in terms of the number of FAs that we formulated in [7]. An adder that adds two n-bit numbers requires at the most $(n+1)$ FAs to compute the sum. We assume that ripple carry adders (RCAs) are used on account of its low power consumption. The area, power, and speed of an adder depend on the *adder width,* $(n+1)$. Therefore, the number of FAs required to implement the multipliers must be minimized. Filter coefficients in CSD form with wordlengths up to 24-bits are considered for analyzing the adder complexity. Since no adjacent bits in CSD are one's, a 24-bit CSD number can have a maximum of 12 nonzero bits and hence at the most twelve nonzero operands could occur in multiplication.

Case I: Odd number of operands: The number of FAs, N_0, required to compute the output corresponding to a coefficient with n operands can be determined using the expression [7]:

$$N_o = (r_2+1)+a_1(r_3+1)+(2r_4+3)+a_3(r_5+1)+(r_6+1)+ a_5(2r_7+3)+(3r_8+6)+a_7(r_9+1)+(r_{10}+1)+a_9(2r_{11}+3) \quad (1)$$

where r_n is the range (number of bits) of the nth operand and a_is are equal to zero except a_{n-2}, which is 1.

Case II: Even number of operands: The number of FAs, (N_e), required to compute the output corresponding to a coefficient with n operands is given by [7]:

$$N_e = (r_2+1)+(2r_4+3)+c_0(r_6+c_0')+(3r_8+6)+ c_1(r_{10}+c_1')+(3r_{12}+6) \quad (2)$$

where $c_0 \equiv \begin{cases} 2, \text{for } n=6 \\ 1, n\neq 6 \end{cases}$, $c_0' \equiv \begin{cases} 1.5, \text{for } n=6 \\ 1, n\neq 6 \end{cases}$,

$c_1 \equiv \begin{cases} 2, \text{for } n=10 \\ 1, n\neq 10 \end{cases}$ and $c_1' \equiv \begin{cases} 1.5, \text{for } n=10 \\ 1, n\neq 10 \end{cases}$.

Note that (1) and (2) are same as in [7], except that we use the notation r_n here instead of s_n and *range* is same as *span* in [7]. Also, here we assume that addition of two n-bit numbers requires at the most $(n+1)$ FAs whereas the assumption in [7] was n FAs (overflow case was ignored in [7]).

The computation of FAs using (1) and (2) can be illustrated through an example. Consider the implementation of the filter tap, $h_k = 0.0000101001010101$. In direct implementation, (i.e., the implementation using shifts and adds and without CSE or any other multiplier optimization techniques) the output of the filter tap is

$$y_k = 2^{-5}x_1 + 2^{-7}x_1 + 2^{-10}x_1 + 2^{-12}x_1 + 2^{-14}x_1 + 2^{-16}x_1 \quad (3)$$

In this case, n is 6 (even), r_2, r_4, and r_6 are 15, 20 and 24 respectively. Using (2) the total number of FA's required to compute (3) in direct method is $(r_2+1)+(2r_4+3)+2(r_6+1.5)$ i.e110 FAs.

The goal of the CSE technique [5] is to identify multiple occurrences of identical bit patterns that are present in the coefficient set. The pattern [1 0 1] is present thrice, which can be expressed as a common subexpression (CS):

$$x_2 = x_1 + x_1 >> 2 \quad (4)$$

where '>>' represents shift right operation. (The notation $x >> a$ represents $2^{-a}.x$). Using the CS given by (4), the output of the filter can be expressed as

$$y_k = x_2 >> 5 + x_2 >> 10 + x_2 >> 14 \quad (5)$$

The numbers of operands (n) in (4) and (5) are 2 and 3 respectively. Therefore, it requires $(r_2+1)=11$ FAs for computing (4) and $(r_2+1)+(r_3+1)=22+26=48$ FAs for (5) as shown alongside the adders A_1, A_2 and A_3 in Fig. 1. The numerals adjacent to the data path represent the number of bitwise right shifts and the numerals in brackets alongside the adders indicate the number of FAs used in the adder.

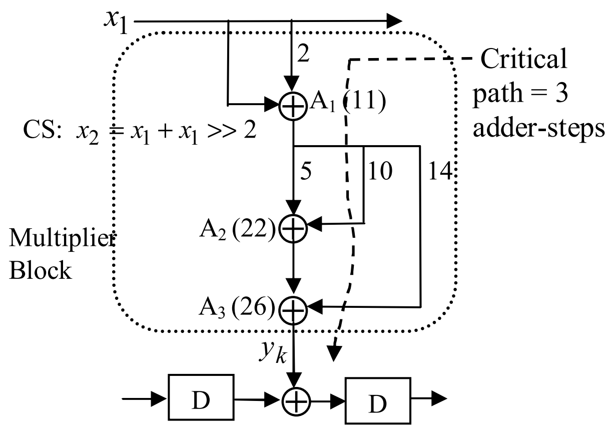

Figure 1. FIR filter tap implementation using CSE.

The number of FAs required for computing y_k in CSE implementation is the sum of FAs required for the adders A_1, A_2 and A_3, which is 59. Thus, the CSE implementation offers 46% reduction of FAs over the direct method in this case. The critical path length (number of adder-steps) of the multiplier realized using CSE method is 3, which is same as that of direct method. Therefore both methods have identical delay.

III. Proposed Method

The key idea in our approach is to reduce the *ranges* of the operands so that the adder width can be reduced which in turn minimizes the number of FAs. In this section, we show that by encoding the filter coefficients using the pseudo floating-point (PFP) arithmetic scheme, the ranges of the operands can be reduced considerably. Further, we employ a coefficient-partitioning algorithm, which offers substantial reduction of FAs in implementing the PFP-coded coefficient multiplier when combined with the CSE method.

A. Coefficient-Partitioning

The coefficient, $h_k = 0.0000101001010101$, in the example in Section II can be represented in PFP as $2^{-5}(2^0 + 2^{-2} + 2^{-5} + 2^{-7} + 2^{-9} + 2^{-11})$. In this expression, the term 2^{-5} is the *shift* part (implying 'right shift by 5'), and the bracketed term is the *span* part. Note that only 3 bits are needed for storing the shift value (CSD representation of 5 is 101) and 11 bits for the span value (bracketed term). Hence h_k can be represented in PFP using 14 bits, whereas its CSD representation requires 16 bits. The basic idea in this approach is to reduce the range of the span part of PFP by partitioning it into two subcomponents, called *sub-coefficients.* We shall now show that the FA requirement can be drastically reduced by coding the sub-coefficients using PFP. We first express each CSD coefficient using CS and the resulting expression is then coded using PFP representation. Let M represents the span of the PFP representation. The span part is partitioned into two sub-components (sub-coefficients) of length $M/2$ (or two sub-components of lengths $\lfloor M/2 \rfloor$ and $\lceil M/2 \rceil$ if M is odd). The latter sub-component is then scaled by its *order* (most significant bit) to reduce its span. The 'partitioned and scaled' versions of the PFP coefficients thus obtained can be added using fewer numbers of FAs since their ranges are reduced. Consider the same example of the filter tap shown in Fig. 1. Using PFP, the filter output obtained in CSE method (5) can be expressed as $2^{-5}(x_2 + 2^{-5}x_2 + 2^{-9}x_2)$. In this case, the span (M) is 9 and the shift is 5. Partitioning the span part into two sub-coefficients, $h_1(n)$ and $h_2(n)$, we have

$$h_1(n) = x_2 \text{ and } h_2(n) = 2^{-5}x_2 + 2^{-9}x_2 \qquad (6)$$

where $h(n)$ is the sum of $h_1(n)$ (MSB half) and $h_2(n)$ (LSB half). The LSB sub-coefficient is further scaled by its order, 2^{-5}, and expressed as $h_2(n) = 2^{-5}(x_2 + 2^{-4}x_2)$. Fig. 2 shows the implementation of the filter tap using our coefficient-partitioning method (CPM).

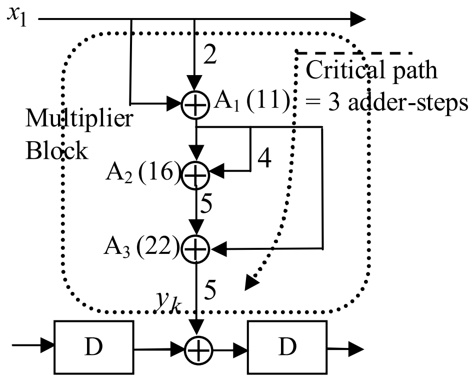

Figure 2. Filter implementation using our CPM.

If x_1 is an 8-bit quantized signal, the ranges of the operands corresponding to the span part of $h_2(n)$ are 11 and 15 and hence the adder A_2 requires at the most 16 FAs. Similarly, the ranges of the operands of A_3 are 11 and 21. Hence A_3 require 22 FAs. Thus, when compared with the direct implementation, the adders A_2 and A_3, have shorter widths since the ranges of their operands are shorter. The shift 2^{-5} of $h_2(n)$ and that of the final expression, $2^{-5}(x_2 + 2^{-5}x_2 + 2^{-9}x_2)$, are performed after the addition stages as shown alongside the data paths at the outputs of adders A_2 and A_3 respectively. Our coefficient-partitioning method requires only 49 FAs to implement the filter tap, which is a reduction of 17% compared with conventional CSE implementation [5]. Even though the critical path lengths of the multiplier obtained using the CSE and our coefficient-partitioning methods are identical (3 adder-steps), the latter method offers better speed since the widths of the adders are less when compared to CSE. The speedup achieved using our coefficient-partitioning method over the CSE is $m.T_{FA}$, where m is the difference in the number of FAs between the CSE and coefficient-partitioning methods and T_{FA} is the delay of one FA in a RCA. Thus our method produces high-speed FIR multipliers when compared to previous methods.

In our method, we combine coefficient-partitioning with the CSE method to further minimize the multiplier complexity. The steps involved in our method are as follows.

Step 1) Design the filter of length N according to the desired specification.
Step 2) Obtain the CSD representation of the infinite-precision coefficients for a desired wordlength. Set $k = 0$.
Step 3) Identify the CS [1 0 1] and [1 0 –1] and their negated versions in $h(k)$. Express the filter output corresponding to the coefficient $h(k)$ using CS.
Step 4) Express the filter output corresponding to $h(k)$ in PFP. Set $M = span.$
Step 5) Partition the span part into two sub-coefficients of length $M/2$ (or lengths $\lfloor M/2 \rfloor$ and $\lceil M/2 \rceil$ if M is odd). Scale the latter sub-coefficients by its *order.*
Step 6) Increment k. If $k \neq N$, go to Step 3. Otherwise, terminate the program.

B. Net Memory

Assume S_{mem} represents the size in bits of memory needed to read the stored input data and to write the current input data. Then S_{mem} for N_i input is [8]:

$$S_{mem} = N_i(W_c + W_x - W_d) \qquad (7)$$

where W_c and W_x are the wordlengths of the coefficients and the input data respectively, and W_d is the difference between the coefficient word length represented in CSD and the wordlength of the span part of the PFP-coded coefficients after coefficient-partitioning. Our CPM reduces the net memory since the number of bits required for the span part is less than that of the CSD representation. Note that the shift part can be stored using fewer number of bits (For 24-bit coefficients, only 5 bits are required to store the maximum shift value 2^{-24}, since $24_{(10)}$ is $11000_{(2)}$. Therefore, the total memory requirement of the span and the shift parts in our CPM is less than that in the CSD representation. We use (7) to compute the net memory needed by the coefficient multipliers.

C. Net Energy Dissipated

The average net computational energy dissipated (power consumed) per output sample, denoted by E_{NET}, is used to calculate the energy savings of our method. The number of adds (n_A) and shifts (n_S) needed for multiplications is given by (8) and (9) [8]:

$$n_A = \frac{1}{2}(W_x + 1).(W_c - W_d - 1) \qquad (8)$$

$$n_S = (W_x + W_c - W_d).(W_c - W_d - 1) \qquad (9)$$

Let E_{add} is the average energy dissipated in a single bit full addition, E_{shift} is the average energy dissipated per bit in a single bit arithmetic shift of a field and E_{mem} is the average energy dissipated in a storage access per bit. Then the average net multiplication energy dissipated, E_{MULT}, and the average net data and coefficients storage accesses energy, E_{MEM}, are obtained using equations (10) and (11) respectively [8].

$$E_{MULT} = N(n_A E_{add} + n_S E_{shift}) \qquad (10)$$

$$E_{MEM} = N(W_c + W_x - W_d)E_{mem} \qquad (11)$$

The average net energy for accumulation of product terms, E_{ACC} is [8]:

$$E_{ACC} = (N-1)(W_c + W_x + \lceil \log_2 N \rceil)E_{add} \qquad (12)$$

The average net overhead storage accesses energy, E'_{MEM} and the average net energy for overhead additions, E'_{ADD} are given by [8]:

$$E'_{MEM} = 2N(W_c + W_x)E_{mem} \qquad (13)$$

$$E'_{ADD} = N(W_c + W_x)E_{add} \qquad (14)$$

The average net computational energy per output, E_{NET} is given by [9]:

$$E_{NET} = \sum_i E_i + \sum_j E'_j, \qquad (15)$$
$$i \in \{MULT, MEM, ACC\}, j \in \{MEM, ADD\}$$

The numerical values for the various cost parameters (energy dissipated as switched capacitance for E_{add} and E_{shift} and delay value τ_{sh}) used in the calculation of E_{NET} were obtained from a characterized low power CMOS library [9]. We use the two different addressable-type memory models (square root and logarithmic) for storage for our analysis as in [9]. The parameter E_{mem} will be a function of the size of the memory. A bigger address space results in greater energy dissipated and delays in decoders and in charging and discharging capacitances during each access. In square root model, the cost will be proportional to the square root of the size of the net memory required, $K_{e1}\sqrt{S_{MEM}}$, whereas in logarithmic model, it will be proportional to the logarithm of the size, $K_{e2}\lceil \log_2(S_{MEM}) \rceil$ with K_{e1} and K_{e2}, the constants of proportionality.

IV. Design Examples

Example 1: The FIR filters employed in the channelizer of the D-AMPS in [7] are considered here. The sampling rate of the wideband signal chosen is 34.02 MHz as in [7]. The pass-band and stop-band edges are 30 kHz and 30.5 kHz respectively. The peak pass-band ripple specification is 0.1 dB. Channel filters with 260, 610, 940 and 1180 taps are chosen to meet the peak stop-band ripple (PSR) specifications of –48 dB, -65 dB, -85 dB and –96 dB at different frequencies as in PDC standard. The percentage reduction of net memory (S_{mem}) achieved using our CPM over the CSE method [5] for 16-bit and 24-bit coefficient wordlengths in designing FIR filters for DAPMS channelizer is shown in Fig. 3. The average reduction of S_{mem} offered by our CPM over the CSE method for the 24-bit and 16-bit coefficients are 24% and 32% respectively. Fig. 4 shows the percentage energy savings achieved in designing 260-tap and 1180-tap FIR filters (DAMPS) with 16-bit coefficient wordlength for the square root model. In this case, the average net energy reductions achieved using CPM for the 260-tap and 1180-tap FIR filters are 40% and 70% respectively. The energy reduction achieved using our CPM in designing these filters for the logarithmic model is shown in Fig. 5. The average energy reductions achieved using CPM over the CSE method for the 260-tap and 1180-tap filters are 35% and 70% respectively.

Example 2: In this example, the channel filters employed in receivers for the PDC standard are implemented. The sampling rate of the wideband signal is 25.6 MHz, which covers 1024 channels of 25 kHz spacing. The peak pass-band ripple specification is 0.1 dB. Channel filters with 240, 590, 880 and 1000 taps are chosen to meet the PSR specifications of –45 dB, -62 dB, -80 dB and –90 dB at different frequencies as in PDC standard. Fig. 6 shows the reduction of net memory over CSE method. The average reductions of memory offered by CPM for the 24-bit and 16-bit coefficients are 26% and 33% respectively. The energy savings achieved using CPM in designing 220-tap and 1000-tap filters with 16-bit coefficient wordlength for the square root model are shown in Fig. 7. In this case, the average net energy reductions achieved using CPM for the 220-tap and 1000-tap FIR filters are 40% and 72% respectively. These reductions are 35% and 69% respectively for the logarithmic model.

V. Conclusions

We have proposed a coefficient-partitioning technique for implementing low power channel filters. Even though we use common subexpression technique for comparison, it must be noted that our coefficient-partitioning algorithm can also be applied to minimum-adder multipliers designed using other hardware optimization methods. Our approach in this paper offers a more general solution for multiplier complexity reduction.

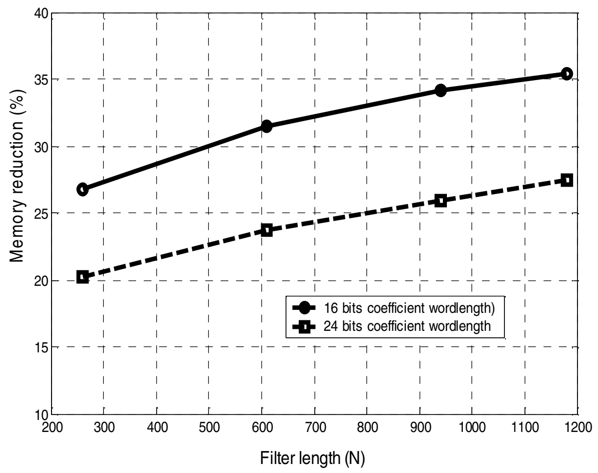

Figure 3. Reduction of net memory over CSE [5] in designing DAPMS channel filters using CPM.

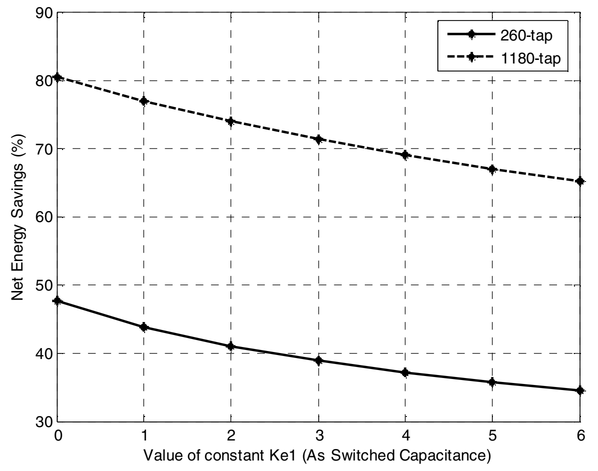

Figure 4. Energy savings using CPM over CSE [5] in designing DAMPS channel filters for square root model.

REFERENCES

[1] W. Tuttlebee and H. W. Wally, *Software Defined Radio: Enabling Technologies.* New York: Wiley, 2002.

[2] J. Mitola, *Software Radio Architecture.* New York: Wiley, 2000.

[3] T. Hentschel, "Channelization for software defined base-stations," *Annales des Telecommunications,* ISSN 0003-4347, vol. 57, pp. 386-420, no. 5-6, May/June 2002.

[4] D. B. Chester, "Digital IF filter technology for 3G systems: An introduction," *IEEE Commun. Mag.,* vol. 37, no. 2, pp. 102-107, February 1999.

[5] R. I. Hartley, "Subexpression sharing in filters using canonic signed digit multipliers," *IEEE Trans. Circuits Syst. II,* vol. 43, pp. 677-688, Oct. 1996.

[6] M. M. Peiro, E. I. Boemo, and L. Wanhammar, "Design of high-speed multiplierless filters using a nonrecursive signed common subexpression algorithm," *IEEE Trans. Circuits Syst. II,* vol. 49, no. 3, pp. 196-203, March 2002.

[7] A. P. Vinod and E. M-K. Lai, "On the implementation of efficient channel filters for wideband receivers by optimizing common subexpression elimination methods," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 24, no. 2, pp. 295-304, February 2005.

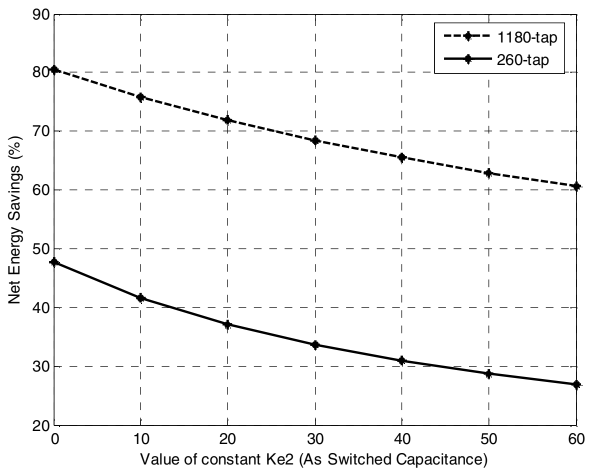

Figure 5. Energy savings using CPM over CSE [5] in designing DAMPS channel filters for logarithmic model.

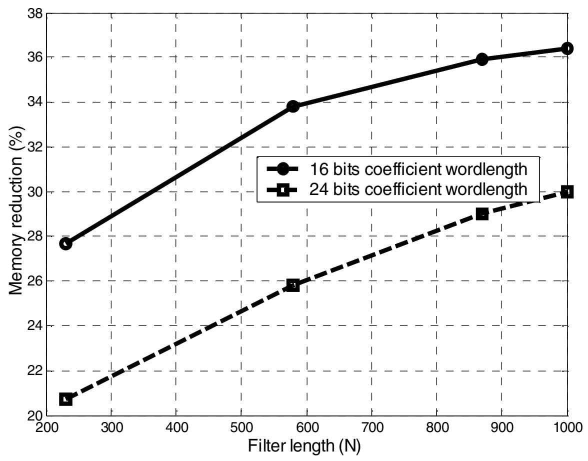

Figure 6. Reduction of net memory over CSE [5] in designing PDC channel filters using CPM.

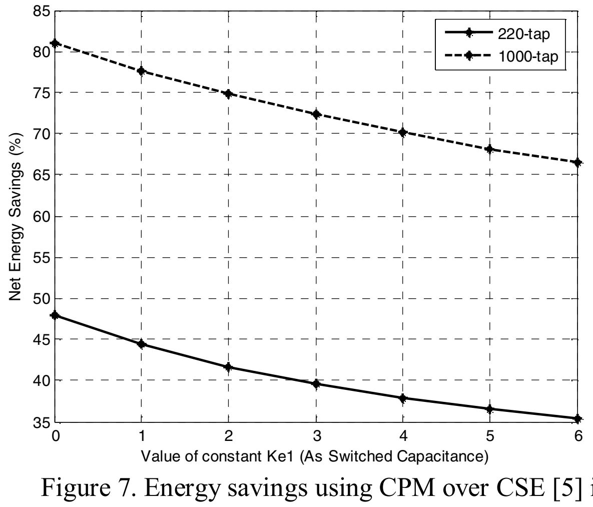

Figure 7. Energy savings using CPM over CSE [5] in designing PDC channel filters for square root model.

[8] N. Sankarayya, K. Roy and D. Bhattacharya, "Algorithms for low power and high speed FIR filter realization using differential coefficients," *IEEE Trans. Circuits Syst. II*, vol. 44, pp. 487-497, June 1997.

[9] T. Burd, "Low-power CMOS library design methodology," M. S. degree thesis, Univ. California, Berkeley, 1994.