An Efficient Coefficient-Partitioning Algorithm for Realizing Low Complexity Digital Filters

A. P. Vinod, *Member, IEEE* and Edmund M-K.Lai, *Senior Member, IEEE*

***Abstract*— The algorithms that minimize the complexity of multiplication in digital filters focus on reducing the number of adders needed to implement the coefficient multipliers. Previous works have not analyzed the complexity of each adder, which is significant in low-complexity implementation. A multiplication algorithm for low complexity implementation of digital filters with a minimum number of full adders and improved speed is proposed here. We exploit the fact that when multiplication is implemented using shifts and adds, the adder width can be minimized by limiting the shifts of the operands to shorter lengths. The coefficient-partitioning algorithm proposed here minimizes the shifts of the operands of the adders by partitioning each coefficient into two sub-components. We show that by combining three methods: the coefficient-partitioning algorithm, an efficient coefficient-coding scheme known as pseudo floating-point representation and the well-known common subexpression elimination (CSE), the number of full adders required in each adder of the multiplier can be reduced considerably. Design examples show that our method offers an average full adder reduction of 30% for finite impulse response (FIR) filters and 20% for infinite impulse response (IIR) filters over CSE methods.**

***Index Terms*— Adder complexity, Common subexpression elimination, Pseudo floating-point representation, Coefficient-partitioning, FIR filters, IIR filters.**

I. Introduction

DIGITAL filtering finds extensive application in mobile communication systems to perform various functions such as channelization, channel equalization, matched filtering and pulse shaping. Low-complexity and high-speed digital filtering for mobile computing and communication applications generally require dedicated hardwired implementations of the filters. Although programmable filters based on digital signal processor cores offer the advantage of flexibility and high sampling rates, they are not suitable for mobile applications that demand high throughput and low-power consumption. Since flexibility of a multiplier is not necessary in such applications, application specific digital filters are frequently adopted to meet the constraints of performance and power consumption. However, these filters employ a large number of multipliers that lead to excessive area and power consumption. Therefore, the problem of designing digital filters with small area and low-power consumption has received a great deal of attention in the last decade.

Manuscript received March 1, 2004.

The authors are with the School of Computer Engineering, Nanyang Technological University, Singapore 639798 (phone: 65-67906258; fax: 65-67926559; e-mail: asvinod@ntu.edu.sg; asmklai@ntu.edu.sg).

A. Related Work

The number of additions (subtractions) used to implement the coefficient multiplications determines the complexity of digital filters. Many approaches have been proposed in literature for reducing the number of adders (subtractors) in the multipliers of digital filters. These approaches include coefficient coding using efficient arithmetic schemes [1, 2], coefficient optimization techniques [3]-[5], distributed arithmetic techniques [6, 7], ROM based designs [8, 9] and common subexpression elimination (CSE) techniques [10]-[18]. Among these approaches, the CSE techniques in [10]-[18] produced the best hardware reduction since it deals with the multiplication of one variable (input signal) with several constants (coefficients). The CSE techniques focus on eliminating redundant computations in multiplier blocks by employing the most common subexpressions consisting of two-nonzero bits. In general, techniques [1]-[18] discuss the complexity of multipliers in terms of the reduction of the number of adders and the critical path. The methods in [1]-[18] have not addressed the issue of minimizing the complexity of each adder of the multiplier, which is significant in low-complexity and high-speed implementations. The differential coefficient method DCM [19] uses differential coefficients to multiply with inputs and compensates the effect of differential coefficients by adding the stored partial product of previous computation. Since differential coefficients have shorter word length, the resulting design and output can also use shorter word length, and thus can reduce power consumption. However, this method results in a lot of overheads, which is proportional to the product of the order of difference and filter tap number. The main idea in [20] is reordering computations and identifying common computations that maximize computation sharing between different multipliers. However the method in [20] offers only a slight improvement in reduction of adders (11%) over the CSE method [11]. Moreover, this method results in an increase in delay, corresponding to the delay of one adder-step on average.

In our recent work [21], we have analyzed the complexity of implementation in terms of full adders (FAs) required for each multiplier of the filter. Two techniques for optimizing the CSE methods to implement low-complexity FIR filters have been proposed in [21]. These techniques are based on the extension of conventional two-nonzero bit (2-bit) common subexpressions (CS) in [10] - [18] 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. These super-subexpressions (SS) eliminate redundant computations of two-nonzero bit CS and hence reduce the number of adders. Since employing SS reduces the number of adders, the number of FAs is also reduced correspondingly – this is the basic approach adopted in [21]. However, it must be noted that the formation of 3-bit and 4-bit SS is based on the occurrence of 2-bit CS with identical shifts between them. Therefore, the main limitation of the method in [21] is its dependence on the statistical distribution of shifts between the 2-bit CS in the CSD representations of LPFIR filter coefficients. It has been shown in [21] that the number of SS grows linearly with the wordlength and hence this technique is more advantageous only when the coefficient wordlength is relatively larger. Moreover, the routing complexity of the method in [21] is higher than that of the 2-bit CSE techniques in [10]-[18] as the former method has more number of subexpressions.

In this paper, we propose an efficient coefficient-partitioning (CP) algorithm to implement the multipliers of digital filters with a minimum number of FAs. We combine three techniques: the CP algorithm, the pseudo floating-point representation and the CSE, to reduce the number of FAs. The FA reduction techniques proposed in this paper do not employ super-subexpressions and hence they do not have the dependence on statistical distribution of shifts between the 2-bit CS. Moreover, our method offers hardware reduction even in the case of filter coefficients with smaller wordlengths. The problem that we address here is how to minimize the number of FAs required in each adder of a minimum-adder filter structure.

The rest of the paper is organized as follows. In section 2, we briefly review the complexity analysis of coefficient multipliers [21] and then present our coefficient-partitioning algorithm. The techniques for optimizing the horizontal and vertical CSE methods by employing the coefficient-partitioning algorithm are presented in section 3. In section 4, we analyze the complexity of implementation of our method. Several design examples of FIR and IIR filters are presented in section 5. Section 6 provides our conclusions.

II. The Coefficient-Partitioning Algorithm

For completeness, a review of the complexity of multiplier block (MB) implementation in terms of FAs required for each multiplier of the filter that we formulated in [21] is presented here. Further, we present our coefficient-partitioning algorithm, and show that the FA requirement can be reduced considerably using our method.

A. Adder Complexity

An adder that adds two *n*-bit numbers requires at the most $(n+1)$ FAs to compute the sum. We consider ripple carry adder (RCA) through out the paper on account of its low power consumption. Even if carry look-ahead adder (CLA) is considered on account of its improved speed, the full adder requirement of CLA is identical to that of RCA (the difference is that CLA will have an extra carry look-ahead logic to reduce the delay at the cost of more 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 [21]:

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

where r_n is the range (number of bits) of the *n*th 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 [21]:

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

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 [21], except that we use the notation r_n here instead of s_n and *range* is same as *span* in [21]. Also, here we assume that addition of two *n*-bit numbers requires at the most $(n+1)$ FAs whereas the assumption in [21] was *n* FAs (overflow case was ignored in [21].

The computation of FAs using (1) and (2) can be illustrated through an example. Consider the CSE implementation of the filter tap, $h_k = 0.0000101001010101$. The pattern [1 0 1] is present thrice, which can be expressed as a common subexpression (CS):

$$x_2 = x_1 + x_1 >> 2 \quad (3)$$

Using the CS given by (3), the output of the filter can be expressed as

$$y_k = x_2 >> 5 + x_2 >> 10 + x_2 >> 14 \quad (4)$$

The numbers of operands (n) in (3) and (4) are 2 and 3 respectively. Therefore, it requires $(r_2+1) = 11$ FAs for computing (3) and $(r_2+1) + (r_3+1) = 22 + 26 = 48$ FAs for (4) as shown alongside the adders A_1, A_2 and A_3 in Fig. 1. The numerals in brackets alongside the adders indicate the number of FAs used in the adder. 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. Using (1) and (2), it can be computed that the *direct implementation* (i.e., shift and add method without using CSE) requires 110 FAs. Thus, the CSE implementation offers 46% reduction of FAs in this case.

B. Coefficient-Partitioning Approach

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 arithmetic scheme, the ranges of the operands can be reduced considerably. Further, we present a novel coefficient-partitioning algorithm, which offers substantial reduction of FAs in implementing the pseudo floating point-coded coefficient multiplier when combined with the CSE method.

The general representation of CSD for the i[th] filter coefficient that has a wordlength B is [2]:

$$h_i = \sum_{j=0}^{B-1} 2^{a_{ij}} \tag{5}$$

The Pseudo Floating-Point (PFP) representation [22] of (5) is:

$$h_i = 2^{a_{i0}} . \sum_{j=0}^{B-1} 2^{a_{ij}-a_{i0}} = 2^{a_{i0}} . \left[\sum_{j=0}^{B-1} 2^{c_{ij}} \right] \tag{6}$$

where $c_{ij} = a_{ij} - a_{i0}$. The term a_{i0} is known as the *shift* and the upper limit value, $(a_{i(B-1)} - a_{i0})$, is known as the *span*. Instead of expressing the coefficients using *B*-bit CSD, it can be expressed as a (*shift, span)* pair using fewer bits. The coefficient, h_k, in the above example 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. The shifts are less complex since they can be hardwired. Therefore, 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. (Note that further reduction of wordlength is possible by hard wiring the shifts in ASIC). Fig. 2 shows the implementation of the filter tap in using PFP coded coefficients. It requires 85 FAs to implement the multiplier block adders (MBAs), (A_1 to A_5), in Fig. 2. Though this FA requirement is less than that of direct implementation (110 FAs), note that the PFP implementation needs more FAs when compared with the CSE method in Fig. 1. The span contributes significantly more to the wordlength requirement than the shift in PFP coding. Therefore, we propose to reduce the ranges of the span component using coefficient-partitioning.

C. Full Adder Reduction by Coefficient-Partitioning

The basic idea in this approach is to reduce the range of the span part of PFP by partitioning it into two sub-components, called *sub-filters.* We shall now show that the FA requirement can be drastically reduced by coding the sub-filters using PFP.

Definition 1 (Order): The most significant bit of a filter coefficient represented in CSD form is defined as the order of the coefficient. For instance, the order of a coefficient $h(n) = 2^{-6} + 2^{-8} + 2^{-11} + 2^{-14} + 2^{-16}$ is 2^{-6}.

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-filters) 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* 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 (4) 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-filters, $h_1(n)$ and $h_2(n)$, we have

$$h_1(n) = x_2 \text{ and } h_2(n) = 2^{-5}x_2 + 2^{-9}x_2 \tag{7}$$

where $h(n)$ is the sum of $h_1(n)$ (MSB half) and $h_2(n)$ (LSB half). The LSB sub-filter is further scaled by its order, 2^{-5}, and expressed as $h_2(n) = 2^{-5}(x_2 + 2^{-4}x_2)$. Fig. 3 shows the implementation of the filter tap using our coefficient-partitioning (CP) method. 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. The data along the signal paths b_is are shown in the table in Fig. 3. Our method requires only 49 FAs to implement the filter tap, which is a reduction of 17% compared with conventional CSE implementation [11]. The MB delay in our CP method and that of the CSE method are identical (3 adder-steps).

III. Optimization of Horizontal And Vertical Cse By Coefficient-Partitioning

There are two classes of CSE methods proposed to tackle the MCM problem in digital filters. The Horizontal Common Subexpression Elimination (HCSE) utilizes the most common horizontal subexpressions that occur *within* each coefficient to eliminate redundant computations [11]-[15]. In general, these methods use Hartley's [11] two most common horizontal subexpressions (HS), i.e., [1 0 1] and [1 0 –1] and their negated versions. The Vertical Common Subexpression Elimination (VCSE) [16] utilizes the vertical subexpressions (VS) that occur *across* the adjacent coefficients to tackle the MCM. The HCSE method offers better reduction of adders for filters whose coefficients are coded using relatively larger wordlengths ($\geq$ 16 bits) whereas in applications that require only shorter wordlengths, the VCSE method is found to be better. In this section, we show that our CP method can be

employed to optimize the HCSE (CP-HCSE technique) and VCSE (CP-VCSE technique) methods to achieve further reduction of FAs. We use Harley's HS, [1 0 1] and [1 0 –1] in our CP-HCSE method and the VS, [1 1], [1 –1], [1 0 1] and [1 0 –1] in CP-VCSE method. From the extensive examples we worked out on CSD representations of FIR and IIR filter coefficients, it has been observed that the above-mentioned subexpressions occur the most (around 70%). Note that other subexpressions such as [1 0 0 1] and [1 0 0 –1] also occur in CSD coefficients. However these subexpressions occur only fewer times. It has been shown in [11] that it is counter-productive to calculate and reuse such subexpressions, since it increases the complexity of the data-flow graph. On the other hand, using the most common subexpressions will have little adverse effect on routability.

A. The CP-HCSE Technique

In this section, we discuss the implementation of a LPFIR filter using the CP-HCSE technique and compare the number of FAs required with that of the HCSE method [11]. A 6-tap FIR filter designed using Parks-McClellan algorithm [23] is considered to illustrate our method. The pass-band and stop-band edges of the filter are 0.2π and 0.25π respectively. The 16-bit CSD form of the coefficients is shown in Fig. 4. The numbers in the first row of Fig. 4 represent the number of bitwise right shifts. The number of FAs required in direct method obtained using (1) and (2) is 317. The HS, [1 0 1] and [1 0 –1], shown inside the rectangles in Fig. 4 are given by:

$$x_2 = x_1 + x_1 >> 2 \text{ and } x_3 = x_1 - x_1 >> 2 \quad (8)$$

With these HS, the output of the filter in HCSE method [11] can be represented as

$$\begin{aligned} &2^{-2}x_1 + 2^{-6}x_3 + 2^{-10}x_2 + 2^{-14}x_3 + 2^{-2}x_3[-1] + 2^{-8}x_2[-1] + \\ &+ 2^{-12}x_3[-1] - 2^{-16}x_1[-1] + 2^{-2}x_1[-2] - 2^{-5}x_1[-2] + \\ &2^{-9}x_1[-2] - 2^{-15}x_1[-2] + 2^{-2}x_1[-3] - 2^{-5}x_1[-3] + 2^{-9}x_1[-3] \\ &- 2^{-15}x_1[-3] + 2^{-2}x_3[-4] + 2^{-8}x_2[-4] + 2^{-12}x_3[-4] \\ &- 2^{-16}x_1[-4] + 2^{-2}x_1[-5] + 2^{-6}x_3[-5] + \\ &2^{-10}x_2[-5] + 2^{-14}x_3[-5] \end{aligned} \quad (9)$$

The number of FAs required to implement the MBAs obtained using (1) and (2) is 227 using HCSE. Using CP-HCSE, the filter output (9) can be expressed as

$$\begin{aligned} &2^{-2}\left(x_1 + 2^{-4}x_3 + 2^{-8}(x_2 + 2^{-4}x_3)\right) + \\ &2^{-2}\left(x_3[-1] + 2^{-6}x_2[-1] + 2^{-10}(x_3[-1] - 2^{-4}x_1[-1])\right) + \\ &2^{-2}\left(x_1[-2] - 2^{-3}x_1[-2] + 2^{-7}(x_1[-2] - 2^{-6}x_1[-2])\right) + \\ &2^{-2}\left(x_1[-3] - 2^{-3}x_1[-3] + 2^{-7}(x_1[-3] - 2^{-6}x_1[-3])\right) + \\ &2^{-2}\left(x_3[-4] + 2^{-6}x_2[-4] + 2^{-10}(x_3[-4] - 2^{-4}x_1[-4])\right) + \\ &2^{-2}\left(x_1[-5] + 2^{-4}x_3[-5] + 2^{-8}(x_2[-5] + 2^{-4}x_3[-5])\right) \end{aligned} \quad (10)$$

The filter structure using the CP-HCSE method is shown in Fig. 4. Only 184 FAs are required using this technique. Note that the HCSE method offers a FA reduction of 28% over direct method whereas the reduction achieved using our CP-HCSE method is 42%. The critical path lengths are identical (3 adder-steps) in both methods.

The CP-HCSE procedure is 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 HS [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 HCSE.

Step 4) Express the HCSE output corresponding to $h(k)$ in PFP. Set $M = span$.

Step 5) Partition the span part into two sub-filters of length $M/2$ (or two sub-components of lengths $\lfloor M/2 \rfloor$ and $\lceil M/2 \rceil$ if M is odd). Scale the latter subfilter by its *order*.

Step 6) Increment k. If $k \neq N$, go to Step 3. Otherwise, terminate the program.

B. The CP-VCSE Technique

In this section, we apply our CP technique to VCSE method [16]. The filter coefficients in previous example (Fig. 3) are used here to illustrate the CP-VCSE technique. Fig. 5 shows the VS present in the coefficient set. The VS, [1 1] and [1 –1], shown inside the rectangles in Fig. 5 are given by:

$$x_4 = x_1 + x_1[-1] \text{ and } x_5 = x_1 - x_1[-1] \quad (11)$$

An inherent drawback of the VCSE method is its inability to completely exploit the symmetry of FIR filter coefficients for efficient implementation of the filter. In the case of HCSE method, since all the bits forming an HS exist within the coefficient, its symmetric counter- part can be easily implemented using delays and structural adders. However, the bits that form VS in VCSE method occur *across* the coefficients and hence the symmetry is destroyed when the bits are of opposite sign [12]. Hence in VCSE implementations, extra adders are required to obtain the symmetric part of the coefficients when more than one VS with bits of opposite sign exist. Due to this constraint, the VCSE implementation requires more adders (thirteen in this case) than the HCSE method (eleven), which will in turn increases the number of FAs and the MB delay. Using (1) and (2), it can be computed that 291 FAs are needed for VCSE implementation. Fig. 6 shows the filter structure using our CP-VCSE method. Our CP-VCSE method needs 254 FAs, which is a reduction of 20% over the direct method (317 FAs are required in direct method) whereas the reduction achieved using the VCSE method is only 8%. The critical path lengths are identical (5 adder steps) in above methods. The reduction of FAs as well as speed performance in vertical subexpression methods (both VCSE and our CP-VCSE) is less than that of the horizontal subexpression methods (HCSE and CP-HCSE). Therefore, the vertical subexpression methods offer better reduction over the horizontal subexpression methods in implementing FIR filters only when the coefficient wordlength is relatively smaller. However, in the case of IIR filters, the requirement of adders in horizontal subexpression methods are also higher since the filter coefficients are not symmetric. Hence, the reductions

offered by vertical subexpression methods are improved in implementing IIR filters than their FIR counterparts.

The CP-VCSE procedure is as follows.

Step 1) Design the filter of length N according to the desired specification.

Step 2) Obtain the CSD representation (in matrix form) of the N infinite-precision coefficients, $h(0)$ to $h(N-1)$, for a desired wordlength. For example, let us consider two cases, i.e., case-1: N is 10 (even number of coefficients, $h(0)$ to $h(9)$) and case 2: N is 9 (odd number of coefficients, $h(0)$ to $h(8)$). Set $k = 0$.

Step 3) Identify the VS [1 1], [1 –1], [1 0 1] and [1 0 –1] and their negated versions that exist across the coefficients, $h(k), h(k+1)$ and $h(k+2)$. (Note that the algorithm will identify the VS across the coefficients, $h(0), h(1)$ and $h(2)$ in the first iteration. In the second iteration, i.e., when *k* is incremented by one in Step 4, it will examine for the VS occur across the coefficients, $h(1), h(2)$ and $h(3)$, after excluding the VS already identified in $h(1)$ and $h(2)$). Express the filter output corresponding to these coefficients using VCSE.

Step 4) Set $k = k+1$. If $k \le (N/2-3)$ for even N and $k \le (\lfloor N/2-3 \rfloor)$ for odd *N*, go to Step 3. Otherwise, go to Step 5. Thus, at the completion of this iteration, the algorithm will identify all the VS present in the first half of the symmetric coefficient set. Considering the example, this first symmetric half set includes $h(0)$ to $h(4)$ for $N = 10$ (case-1) and $h(0)$ to $h(3)$ for $N = 9$ (case-2).

Step 5) Set $k = N-1$. Identify the VS that exist across the coefficients, $h(k)$, $h(k-1)$ and $h(k-2)$. (Considering case-1, the algorithm will identify the VS across the coefficients, $h(9)$, $h(8)$ and $h(7)$ in the first iteration. In the second iteration, i.e., when *k* is decremented by one in Step 6, it will examine for the VS occur across the coefficients, $h(8)$, $h(7)$ and $h(6)$, after excluding the VS already identified in $h(9)$ and $h(8)$). Express the filter output corresponding to these coefficients using VCSE.

Step 6) Set $k = k-1$. If $k \ge ((N/2)+2)$ for even N and $k \ge (\lfloor (N/2)+3 \rfloor)$ for odd *N*, go to Step 5. Otherwise, go to Step 7. Thus, the algorithm will identify all the VS present in the second half of the symmetric coefficient set. Considering the example, this second symmetric half set includes $h(5)$ to $h(9)$ for $N = 10$ (case-1) and $h(5)$ to $h(8)$ for $N = 9$ (case-2). Note that the central coefficient $h(\lfloor N/2 \rfloor)$ (i.e., $h(4)$ in case-2) is excluded when the number of coefficients is odd since it does not have a symmetric counter part.

Step 7) Examine the VCSE expressions obtained in Step 4 (former symmetric half) and Step 6 (latter symmetric half) for terms that have identical shifts and delays. Obtain the VCSE expressions of the latter part from the former part by using appropriate delay operations. Finally, add those terms that are left out (terms that do not have any common shift and delay).

Step 8) Obtain the PFP of each of the expressions obtained in Step 7. Set $M = span$. Partition the span part into two sub-filters of length $M/2$ (or two sub-components of lengths $\lfloor M/2 \rfloor$ and $\lceil M/2 \rceil$ if M is odd). Scale the latter subfilter by its *order*. Terminate the program when all the VCSE expressions are coded using CP-VCSE.

IV. Adder Complexity in Coefficient-Partitioning

The number of FAs required in CP method can be obtained by modifying (1) and (2) as follows. If x_2 and x_3 are the CS obtained from the input x_1, and x_{k_j} represents the data from the set $\{x_1, x_2, x_3\}$ that has to be shifted corresponding to the position of the *j*-th CSD bit, the general expression for filter output corresponding to a coefficient $h(n)$ of wordlength B is

$$y(n) = \sum_{j=1}^{z} (s_j 2^{-p_j})(x_{k_j}) \tag{12}$$

where $s_j \in \{-1, 0, 1\}$, $p_j \in \{0, 1, ...B\}$, and z is the number of nonzero digits. If p_{s_1} is the shift, (12) can be expressed in PFP form as

$$y(n) = 2^{-p_{s1}} \sum_{j=1}^{z} (s_j 2^{-(p_j - p_{s1})})(x_{k_j}) \tag{13}$$

Partitioning the coefficient into two sub-components of lengths $M_1 = \lfloor (p_z - ps1)/2 \rfloor$ and $M_2 = \lceil (p_z - ps1)/2 \rceil$, and scaling the later part by its order, $2^{-p_{s2}}$, (13) can be written as

$$y(n) = 2^{-p_{s1}} \left[\begin{array}{l} \sum_{j_1=1}^{M_1} s_{j_1} 2^{-(p_{j_1} - p_{s_1})} x_{k_{j_1}} + \\ 2^{-p_{s2}} \left(\sum_{j_2=1}^{M_2} s_{j_2} 2^{-(p_{j_2} - p_{s_1} - p_{s_2})} x_{k_{j_2}} \right) \end{array} \right] \tag{14}$$

where $(p_{j_1} - p_{s_1}) \in \{0,1,.....\lfloor (p_z - p_{s_1})/2 \rfloor\}$ and $(p_{j_2} - p_{s_1} - p_{s_2}) \in \{\lceil (p_z - p_{s_1})/2 \rceil, \lceil (p_z - p_{s_1})/2 \rceil + 1, ... (p_z - p_{s_1})\}$.

Let $\sum_{j_1=1}^{M_1} a_{j_1}$ and $\sum_{j_2=1}^{M_2} b_{j_2}$ represent the number of FAs required to compute the terms, $\sum_{j_1=1}^{M_1} s_{j_1} 2^{-(p_{j_1} - p_{s_1})} x_{k_{j_1}}$, and $\sum_{j_2=1}^{M_2} s_{j_2} 2^{-(p_{j_2} - p_{s_1} - p_{s_2})} x_{k_{j_2}}$ respectively, obtained using (1) and (2). The total number of FAs required to compute (14) is

$$\sum_{j_1=1}^{M_1} a_{j_1} + \sum_{j_2=1}^{M_2} b_{j_2} + F_n \tag{15}$$

where F_n is the range of the last term of (14), i.e., $s_j\, 2^{-(p_z - p_{s_1})} x_{k_z}$.

We have also examined the adder complexity reduction achieved by partitioning the coefficient into more than two sub-components. When the filter output expression (14) (corresponding to partitioning into two halves) is implemented using the parallel tree-structured addition, the inner shift operation $2^{-p_{s_2}}$, is performed just before the final-stage adder of the tree. Therefore, the widths of the adders in the preceding stages that compute the sum of the bracketed term of $2^{-p_{s_2}}$ are less and only the final-stage adder requires the highest width. Assume that the coefficient is partitioned into n sub-components, instead of two. The output expression (14) can be written as

$$y(n) = 2^{-p_{s_1}} \left[\sum_{j_1=1}^{M_1} s_{j_1} 2^{-(p_{j_1} - p_{s_1})} x_{k_{j_1}} + 2^{-p_{s_2}} \left(\sum_{j_2=1}^{M_2} s_{j_2} 2^{-(p_{j_2} - p_{s_1} - p_{s_2})} \right)_{x_{k_{j_2}}} + \cdots\cdots\cdots\cdots + 2^{-p_{s_n}} \left(\sum_{j_n=1}^{M_n} s_{j_n} 2^{-(p_{j_n} - p_{s_1} - p_{s_n})} x_{k_{j_n}} \right) \right] \qquad (16)$$

In this case, the widths of the adders in the intermediate-stages of the tree-structure are larger since the multiple inner shifts, $(2^{-p_{s_2}}, 2^{-p_{s_3}}, \ldots\ldots, 2^{-p_{s_n}})$, in (16) need to be performed prior to the intermediate additions. Hence, each of these intermediate-stage adders would require more FAs. Therefore, partitioning a coefficient into two halves offers the best FA reduction than partitioning into multiple parts.

The number of times each subexpression occurs is counted by an exhaustive search. In HCSE method, this requires a time complexity of $O(n^2)$ where n is the total number of nonzero terms in the coefficient set [11]. Assume that there are N coefficients and each coefficient has at the most B nonzero bits. Thus, the total number of nonzero bits is $(N \times B)$ and time complexity of HCSE algorithm is $O((N \times B)^2)$. When compared with HCSE, our method has an additional complexity of coefficient-partitioning. Assume that W is the coefficient wordlength, W_a is the span of the PFP representation and X is the position of the first nonzero bit from the MSB. The maximum value of W_a (i.e., $W_{a,\max}$) is W and the minimum value of X (i.e., $X_{\min}$) is 1 (this is the case when the MSB is a nonzero bit). In the case of a CSD number, since no adjacent bits are ones, if a nonzero bit occurs at $(W-2)$, then the only possibility of occurrence of the next nonzero bit is in the LSB position, i.e., at W. Note that if $X = W$, it means that there is only one nonzero bit in the coefficient and hence no adders are needed to compute the output for that tap. Therefore, the maximum value of X (i.e., $X_{\max}$) for any coefficient that has more than one nonzero bit is $(W-2)$. The additional complexity in our method is mainly on the computation of shift differences of the LSB half of the PFP coefficient. This is linearly related to the number of nonzero terms in the LSB half. A PFP coefficient can have a maximum of $\lfloor W_{a,\max} - X_{\min})/2 \rfloor = (W-1)/2$ nonzero terms in its LSB half after partitioning. Thus, the additional complexity of our method for N coefficients is the search time to determine X as well as to compute the shift differences of the LSB half, which is given by $O\left[\left[(W-2) + \lceil (W-1)/2 \rceil\right] \times N\right]$. The total time complexity is

$$O((N \times B)^2) + O\left[\left[(W-2) + \lceil (W-1)/2 \rceil\right] \times N\right] \qquad (17)$$

A comparison of the time complexity of our algorithm with that of the HCSE [11] can be made through an example. Consider the CSD coefficients in Fig. 3. In this case, N is 6, total number of nonzero bits (i.e., $N \times B$) is 36 and W is 16. The complexity of HCSE algorithm is $O(36^2) = 1296$. Using (27), the complexity of our method is 1428, which is only 10% more than the HCSE method.

V. Experimental Results

In this section, we present examples of implementing several FIR and IIR filters using our algorithm. The proposed CP-HCSE and CP-VCSE techniques are compared with HCSE [11] and VCSE [16] methods and reductions of FAs are determined.

A. FIR Filters

The FIR filters are designed using Parks-McClellan algorithm. The normalized pass-band and the stop-band edges of the filter are 0.2π and 0.25π respectively. The CSD representation of the coefficients using 8, 12, 16, 20 and 24 bits are considered. *Direct method* means the CSD implementation without employing any CSE technique. Table I shows the number of FAs required to implement the multipliers of the filters of different lengths (N) using the direct method. Comparison of the number of FAs required in HCSE method [11] and our CP-HCSE method is shown in Table II. The FA requirement in VCSE [16] and our CP-VCSE methods are listed in Table III. Fig. 7 shows the percent reduction of FAs achieved using various methods over the direct method in designing the FIR filter of length 50, for different wordlengths. Note that the VCSE [16] and the proposed CP-VCSE methods offer better reduction than their horizontal counterparts (HCSE [11] and our CP-HCSE respectively) only when the coefficient wordlength is smaller (8 bits). For wordlengths larger than 8 bits, both the horizontal subexpression techniques result in higher reduction than the vertical subexpression techniques. The average reductions of VCSE and CP-VCSE are 26.5% and 42% respectively whereas these figures are 34.8% and 51.7% for HCSE and CP-HCSE methods. This shows that our methods offer a significant reduction of FAs in MB implementations.

The reduction of FAs over the direct method in designing the FIR filter whose coefficients are coded using 16-bit CSD, for different number of filter taps are shown in Fig. 8. The average reduction offered by the CP-HCSE method is 54%, which is the

highest among all the techniques. The average reduction achieved using the CP-VCSE method is 44.2% where as that of the HCSE and the VCSE methods are 36.4% and 22.4% respectively. Furthermore, Fig. 8 shows that our methods offer higher reduction of FAs for higher-order filters. For example, in the case of the filter with 10 taps, the reduction achieved using our CP-HCSE technique (46.5%) is higher than that of the HCSE method (35.2%) by 11.3%. On the other hand, for the filter with 400 taps, these reductions are 60.6% and 38.7% respectively. Thus, in the case of 400-tap filter, our CP-HCSE method offers a reduction of 21.9% over the HCSE method. This reduction is almost twice that for the filter with 10 taps.

The reductions of FAs achieved using our CP-HCSE technique over the HCSE method for different numbers of filter taps are shown in Fig. 9. For wordlengths of 12 bits and above, our method results in higher reduction as the filter order increases. This illustrates that the use of shorter shifts by partitioning the coefficient results in significant reduction of FAs required to implement the low-magnitude end-coefficients (end-coefficient here being defined as the first $N/4$ coefficients, $h(0)$ to $h(\lceil N/4 \rceil - 1)$, of an FIR filter) of higher order filters. Our CP-HCSE technique offers average FA reductions of 26.5% and 32.8% over the HCSE method for the 16-bit and 24-bit cases respectively.

We have examined the reduction of FAs for FIR filters of various specifications. Our simulation results show that reductions are identical to the above-mentioned design example for filters of different pass-band and stop-band edges. Furthermore, it has been observed that slightly better FA reductions are achieved for filters with relatively wider transition bands. As the transition band of the filter becomes wider, the side-lobes of the impulse response decrease and hence the magnitudes of the end-coefficients of $h(n)$ will also decrease. Hence, most of the nonzero bits of the CSD representations of end-coefficients occur in the LSB part and the use of shorter *shifts* in our method results in considerable reduction of FAs. We illustrate this using an example of an FIR filter whose transition band is wider than that in the previous design example. The normalized pass-band and the stop-band edges of the filter are 0.2π and 0.35π respectively. The reduction of FAs over the direct method in designing the filter whose coefficients are coded using 16-bit CSD, for different number of filter taps are shown in Fig. 10. The average reductions offered by the CP-HCSE and the CP-VCSE methods are 58% and 47.2% respectively, whereas these reductions were 54% and 44.2% for the filter in the previous example. The reductions achieved using HCSE and VCSE methods are almost identical in both cases. Thus, our coefficient-partitioning methods offer improved FA reductions when employed to implement filters with relatively wider transition bands. It may also be noted that the critical path lengths of filter structures obtained using our CP methods are same as that of the CSE methods.

Though we use the CSE techniques for comparison throughout the paper, it must be noted that our CP algorithm can also be applied to minimum-adder multipliers designed using other methods. Basically, the CP algorithm can optimize the coefficient multipliers designed using any other complexity reduction techniques to further minimize the number of FAs. We have examined the reduction of FAs achieved when the CP method is applied to the nonrecursive signed common subexpression (SCSE) method [15]. The filters, FIR2 (order 16, wordlength 16), FIR4 (order 25, wordlength 9) and FIR6 (order 60, wordlength 14), used in [15] are used here for comparison. Table IV shows the comparison of the number of FAs (NFA) and critical path lengths (CPL). Note that the SCSE method optimized using our CP algorithm (CP-SCSE method) offers an average FA reduction of 15% over the SCSE method. The critical path lengths of these two methods are identical. These examples illustrate that our approach offers a more general solution for multiplier complexity reduction.

B. IIR Filters

We consider the implementation of elliptic IIR filters since for a given set of specifications, it has a lower order than any other IIR filter type. The normalized cutoff frequency (ω_n) of the filter is 0.2, the pass-band ripple is 0.1dB and the stop-band ripple is 50dB. The order of an IIR filter required to meet a given frequency response specification is considerably less than its FIR counterpart. However, IIR filters are more sensitive to coefficient quantization as it may result in change in pole locations that eventually causes higher quantization error and limit cycle effect [23]. Therefore, more number of bits are required in the representation of the coefficients of IIR filters as compared to their FIR counterparts. Hence, the CSD representation of the coefficients using 12, 16, 20, 24 and 32 bits are considered in this example. Table V shows the number of FAs required to implement the multipliers of the filters of different orders (*N*) using direct method. Comparison of the number of FAs required in HCSE method and our CP-HCSE method is shown in Table VI. The FA requirement in VCSE and our CP-VCSE methods are listed in Table VII. Fig. 11 shows the reduction of FAs achieved using various methods over the direct method in designing the elliptic IIR filter of order 13, for different wordlengths.

Since the IIR filter coefficients are not symmetric, the horizontal subexpression techniques do not have the advantage of exploiting the coefficient symmetry as in FIR filter designs. Hence, both the vertical subexpression methods (VCSE [16] and our CP-VCSE) offer higher reduction than the horizontal subexpression techniques for wordlengths of 12 and 16 bits. However, for larger wordlengths (20, 24 and 32 bits), the horizontal subexpression techniques (HCSE [11] and our CP-HCSE) results in better reduction than their vertical counterparts. It can be seen that the reduction of FAs achieved using the HCSE method decreases for larger wordlengths, whereas our CP-HCSE technique results in a better reduction in such cases. In the case of VCSE method, the FA reduction decreases considerably for larger wordlengths (from 44% for 12 bits to 19% for 32 bits). Though our CP-VCSE also shows a similar behavior for larger wordlengths, note that the reduction of FAs is only minimal (from 49.2% for 12 bits to 43% for 32 bits). Our CP-VCSE technique offers the best reduction of FAs among all other methods for 12 and 16 bit implementations. In the case of wordlengths larger than 16 bits, our CP-HCSE technique offers the best reduction. From Fig. 11, it can be seen that the HCSE and VCSE method offer identical average FA

reductions (31%) for different wordlengths, over the direct method. The average FA reductions offered by our CP-HCSE and CP-VCSE techniques are almost identical (44.4% and 44.8% respectively).

We have examined the reduction of FAs for IIR filters with different pass-band and pass-band edges as well as various stop-band ripples. In addition to elliptic filters, Butterworth and Chebyshev IIR filters were also examined. It has been noted that our CP methods result in considerable reduction of FAs in all these cases irrespective of the filter specifications. Thus, the simulation results of FIR and IIR filters clearly illustrate that our coefficient-partitioning techniques (CP-HCSE and CP-VCSE) offer the best reduction of FAs than minimum-adder implementation methods using HCSE [11] and VCSE [16]. Based on the simulation results, the following guidelines for choosing the best implementation method from our CP-HCSE and CP-VCSE methods can be formulated.

1. The CP-HCSE technique offers the best FA reduction for designing FIR filters than the CP-VCSE method except in the case where the coefficient wordlength is smaller (8 bits). In most practical filter applications, the frequency response of the filter will deteriorate if the coefficient is coded using a wordlength of 8 bits. This deterioration is minimal only for filters with fewer taps (typically, less than 15 taps). Therefore, in such cases the CP-VCSE technique can be adopted since it offers better reduction than the CP-HCSE method when the coefficient wordlength is 8 bits. For filters with taps more than fifteen, larger wordlengths (12, 16 and 24 bits) are required to meet the desired magnitude response. Hence, the CP-HCSE method would be the best choice in such cases.
2. In the case of IIR filters, the CP-VCSE method offers the best reduction for coefficient wordlengths of 12 and 16 bits. For larger wordlengths, the CP-HCSE method offers a slightly better reduction than the CP-VCSE method.

VI. Conclusions

We have presented a coefficient-partitioning method to implement low-complexity digital filters with a minimum number of FAs. While the optimization criterion in conventional low-complexity filter implementation methods is the number of adders, the focus of our method is to minimize the number of FAs required for each adder. Our coefficient-partitioning algorithm is combined with the pseudo floating-point coefficient-coding scheme and applied to optimize the common subexpression elimination methods. Design examples show that our methods offer considerable reduction of FAs when compared with HCSE [11] and VCSE [16] methods. The FA reduction achieved using our methods is substantially higher for higher order filters.

References

[1] G. K. Ma and F. J. Taylor, "Multiplier policies for digital signal processing," *IEEE ASSP Magazine,* vol. 7, no. 1, pp. 6-20, Jan. 1990.

[2] D. Li, "Minimum number of adders for implementing a multiplier and its application to the design of multiplierless digital filters," *IEEE Trans. Circuits Syst.,* vol. 42, no. 7, pp. 453-460, July 1995.

[3] H. Samueli, "An improved search algorithm for the design of multiplierless FIR filters with powers-of-two coefficients," *IEEE Trans. Circuits Syst.,* vol. 36, no. 7, pp. 1044-1047, July 1989.

[4] Y. C. Lim and S. R. Parker, "Design of discrete coefficient-value linear phase FIR filters with optimum normalized peak ripple magnitude," *IEEE Trans. Circuits Syst.,* vol. 37, no. 12, pp. 1480-1486, Dec. 1990.

[5] C.-L. Chen and A. N. Willson, "A trellis search algorithm for the design of FIR filters with signed-powers-of-two coefficients," *IEEE Trans. Circuits Syst. II,* vol. 46, no. 1, pp. 29-39, Jan. 1999.

[6] C. F. Chen, "Implementing FIR filters with distributed arithmetic," *IEEE Trans. Acoustics, Speech, and Signal Processing,* vol. 33, no. 5, pp. 1318-1321, Oct. 1985.

[7] S. A. White, "Applications of distributed arithmetic to digital signal processing: a tutorial review," *IEEE ASSP Magazine,* vol. 6, no. 13, pp. 4-19, July 1989.

[8] T. Tjahjadi and W. Steenart, "Non-recursive digital FIR filter implementation using stored square ROM multipliers," in proc. *IEEE ICASSP,* vol. 8, pp. 1196-1199, April 1983.

[9] J. I. Guo, C. M. Liu, and C. W. Jen, "The efficient memory-based VLSI arrays for DFT and DCT," *IEEE Trans. Circuits Syst. II,* vol. 39, no. 10, pp. 723-733, Oct. 1992.

[10] M. Mehendale, S. D. Sherlekar, and G. Venkatesh, "Synthesis of multiplierless FIR filters with minimum number of additions," in *Proceedings of the 1995 IEEE/ACM International Conference on Computer-Aided Design,* Los Alamitos, CA: IEEE Computer Society Press, 1995, pp. 668-671.

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

[12] M. Yagyu, A. Nishihara, and N. Fujii, "Fast FIR digital filter structures using minimal number of adders and its application to filter design," *ICICE Trans. Fundam. Electron. Commun. Comput. Sci.,* no. 8, pp. 1120-1129, E79-A, 1996.

[13] R. Pasko, P. Schaumont, V. Derudder, S. Vernalde, and D. Durackova, "A new algorithm for elimination of common subexpressions," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Syst.,* vol. 18, no. 1, pp. 58-68, January 1999.

[14] R. Pasko, P. Schaumont, V. Derudder, and D. Durackova, "Optimization method for broadband modem FIR filter design using common subexpression elimination," in *Proceedings on the 10th International Symposium on System Synthesis,* pp. 100-106, 1997.

[15] 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.

[16] Y. Jang and S. Yang, "Low-power CSD linear phase FIR filter structure using vertical common sub-expression," *Electronics Letters,* vol. 38, no. 15, pp. 777-779, July 2002.

[17] D. R. Bull and D. H. Horrocks, "Primitive operator digital filters," *IEE Proceedings – G,* vol. 138, no. 3, pp. 401-412, June 1991.

[18] A. G. Dempster and M. D. Mcleod, "Use of minimum adder multiplier blocks in FIR digital filters," *IEEE Trans. Circuits Syst. II,* vol. 42, pp. 569-577, Sept. 1995.

[19] N. Shakarayya, 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. 488-497, June 1997.

[20] H. Choo, K. Muhammad and K. Roy, "Complexity reduction of digital filters using shift inclusive differential coefficients" *IEEE Transactions on Signal Processing,* vol. 52, no. 6, pp. 1760-1772, June 2004.

[21] 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 Syst.,* vol. 24, no. 2, pp. 295-304, February 2005.

[22] A.P. Vinod, A.B. Premkumar and E. M-K. Lai, "An optimal Entropy coding scheme for efficient implementation of pulse shaping FIR filters in digital receivers," *Proceedings of IEEE International Symposium on Circuits and Systs*, vol. 4, pp. 229-232, May 2003.

[23] J. G. Proakis and D. G. Manolakis, *Digital Signal Processing Principles, Algorithms, and* Applications. Prentice Hall, 1998.

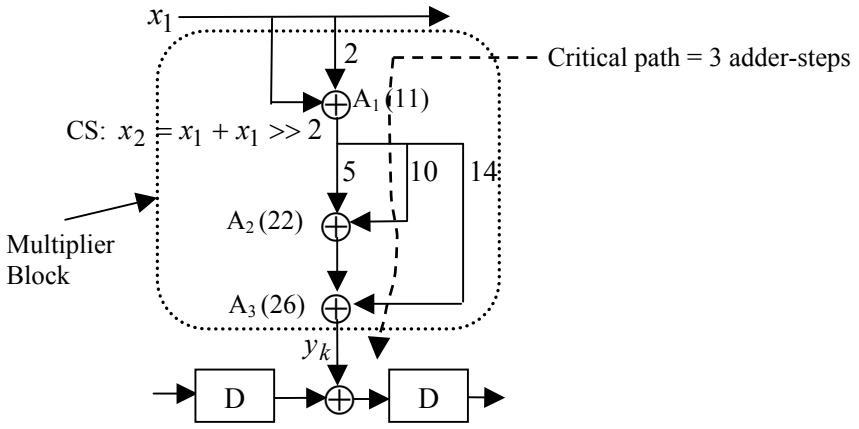
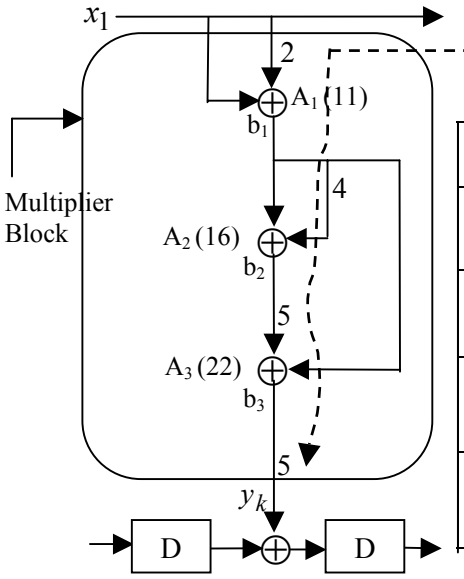


Fig. 1. FIR filter implementation using CSE.



Signal path	Data
b_1	$x_2 = x_1 + x_1 >> 2$ (CS)
b_2	$x_2 + x_2 >> 4 = x_2 + 2^{-4} x_2$
b_3	$x_2 + x_2 >> 5 + x_2 >> 9$ $= x_2 + 2^{-5} x_2 + 2^{-9} x_2$
y_k	$x_2 >> 5 + x_2 >> 10 + x_2 >> 14$ $2^{-5} x_2 + 2^{-10} x_2 + 2^{-14} x_2$

Fig. 2. Filter implementation using CP method.

Bit shift / $h(n)$	-1	-2	-3	-4	-5	-6	-7	-8	-9	-10	-11	-12	-13	-14	-15	-16
$h(0)$	0	1	0	0	0	1	0	-1	0	1	0	1	0	1	0	-1
$h(1)$	0	1	0	-1	0	0	0	1	0	1	0	1	0	-1	0	-1
$h(2)$	0	1	0	0	-1	0	0	0	1	0	0	0	0	0	-1	0
$h(3)$	0	1	0	0	-1	0	0	0	1	0	0	0	0	0	-1	0
$h(4)$	0	1	0	-1	0	0	0	1	0	1	0	1	0	-1	0	-1
$h(5)$	0	1	0	0	0	1	0	-1		1	0	1	0	1	0	-1

Fig. 3. Horizontal Subexpressions in 6-tap FIR filter.

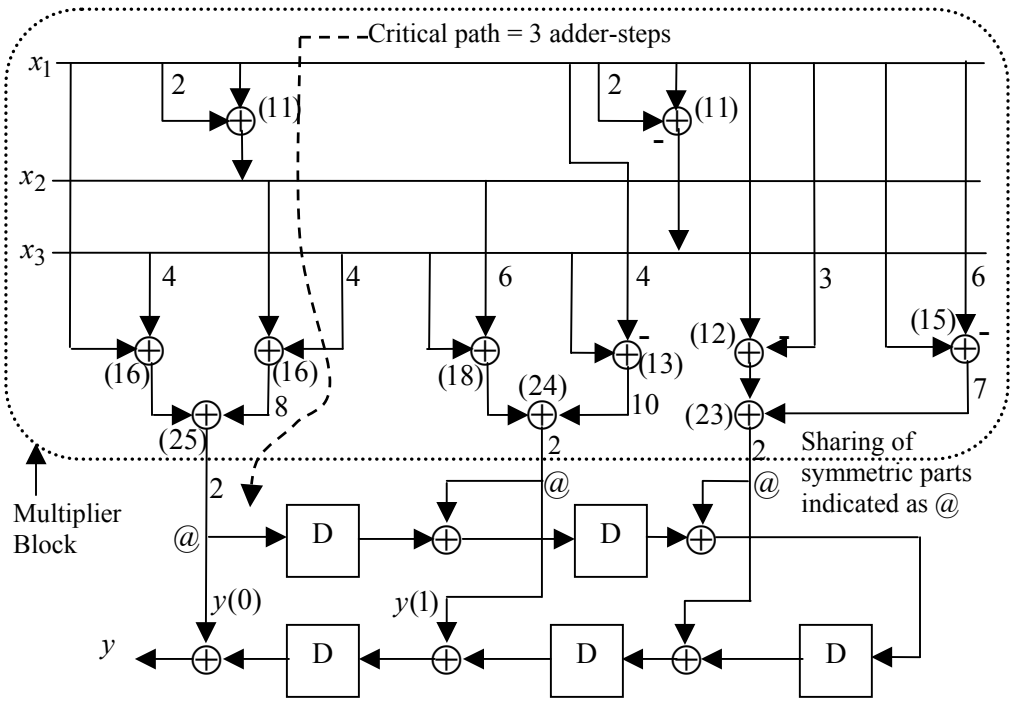


Fig. 4. Proposed filter structure using CP-HCSE of Fig. 3.

Bit shift	-1	-2	-3	-4	-5	-6	-7	-8	-9	-10	-11	-12	-13	-14	-15	-16
$h(n)$																
$h(0)$	0	1	0	0	0	1	0	-1	0	1	0	1	0	1	0	-1
$h(1)$	0	1	0	-1	0	0	0	1	0	1	0	1	0	-1	0	-1
$h(2)$	0	1	0	0	-1	0	0	0	1	0	0	0	0	0	-1	0
$h(3)$	0	1	0	0	-1	0	0	0	1	0	0	0	0	0	-1	0
$h(4)$	0	1	0	-1	0	0	0	1	0	1	0	1	0	-1	0	-1
$h(5)$	0	1	0	0	0	1	0	-1		1	0	1	0	1	0	-1

Fig. 5. Vertical Subexpressions in 6-tap FIR filter.

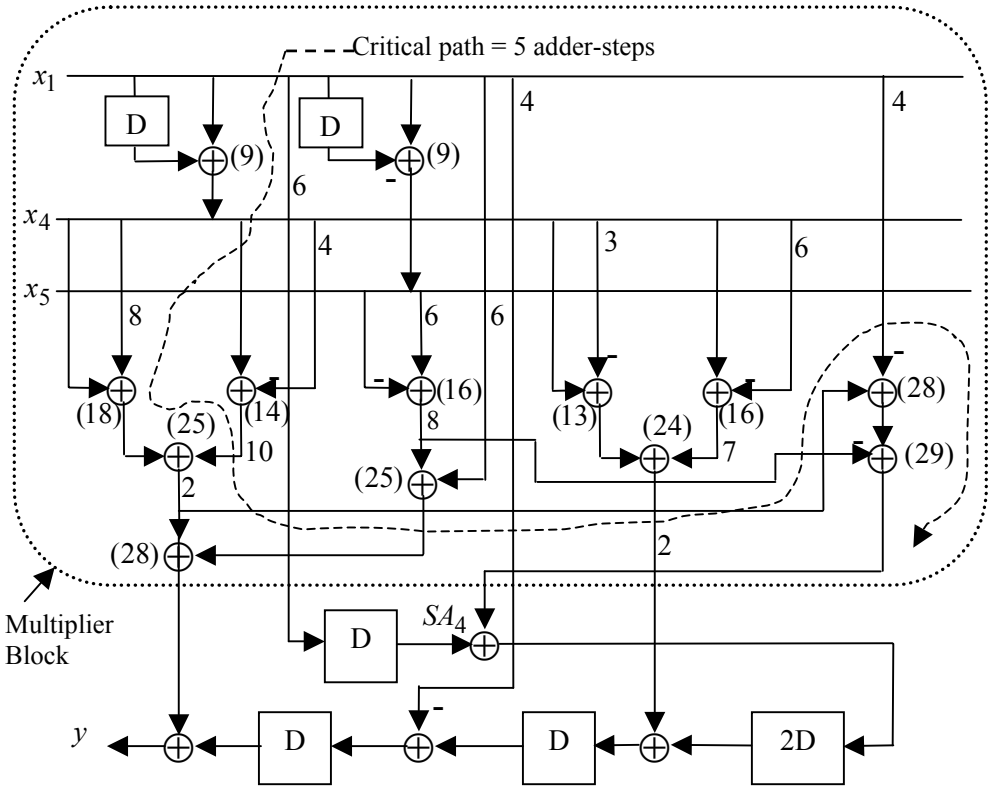


Fig. 6. Proposed filter structure using CP-VCSE of Fig. 5.

Table I Number of FAs required to implement the FIR filters using *direct method*

Filter length *(N)*	Number of FAs in Direct Method				
	8-bit	12-bit	16-bit	20-bit	24-bit
10	162	341	550	648	972
30	222	588	1141	1662	2260
50	303	940	1690	2774	3832
80	327	1429	2617	4022	5820
120	414	1581	3462	5766	8205
250	663	2189	5294	9810	14948
400	947	2672	6414	11936	19717

Table II Number of FAs required to implement the FIR filters using HCSE and proposed CP-HCSE methods

N	Number of FAs in HCSE Method [11]					Number of FAs in CP-HCSE Method				
	8-bit	12-bit	16-bit	20-bit	24-bit	8 -bit	12-bit	16-bit	20-bit	24-bit
10	122	217	357	421	618	114	181	292	335	468
30	213	372	775	1197	1557	182	311	597	856	1128
50	182	615	1097	1778	2444	162	473	831	1243	1671
80	237	839	1586	2482	3719	211	612	1154	1725	2468
120	297	746	2181	3748	5506	255	560	1499	2444	3495
250	467	1294	3319	6004	9627	399	930	2118	3767	5740
400	827	1510	3919	7413	12402	706	1050	2540	4607	7256

Table III Number of FAs required to implement the FIR filters using VCSE and proposed CP-VCSE methods

N	Number of FAs in VCSE Method [16]					Number of FAs in CP-VCSE Method				
	8-bit	12-bit	16-bit	20-bit	24-bit	8 -bit	12-bit	16-bit	20-bit	24-bit
10	104	233	467	566	804	95	225	356	411	539
30	177	389	950	1419	2000	149	375	679	979	1546
50	154	624	1306	2272	3119	137	491	990	1731	2521
80	203	885	2007	3097	4732	174	870	1395	2156	3329
120	247	787	2597	4612	6531	217	764	1818	3096	4726
250	391	1357	3865	6916	11839	334	1353	2753	4827	7967
400	653	1598	4662	8689	15773	495	1601	3188	6159	10588

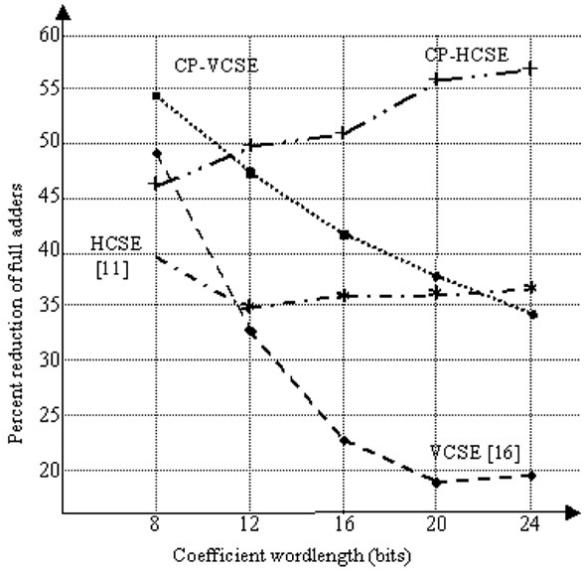


Fig. 7. Reduction of full adders achieved using various methods over the direct method in designing the FIR filter of length 50, for different wordlengths.

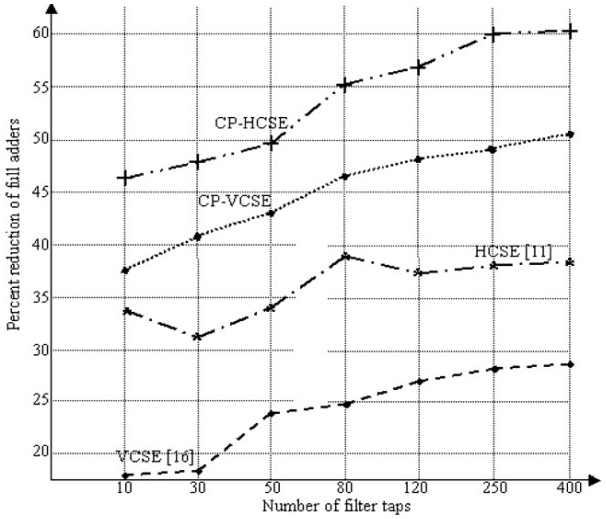


Fig. 8. Reduction of full adders over the direct method in designing the FIR filter with coefficient wordlength of 16 bits, for different number of filter taps.

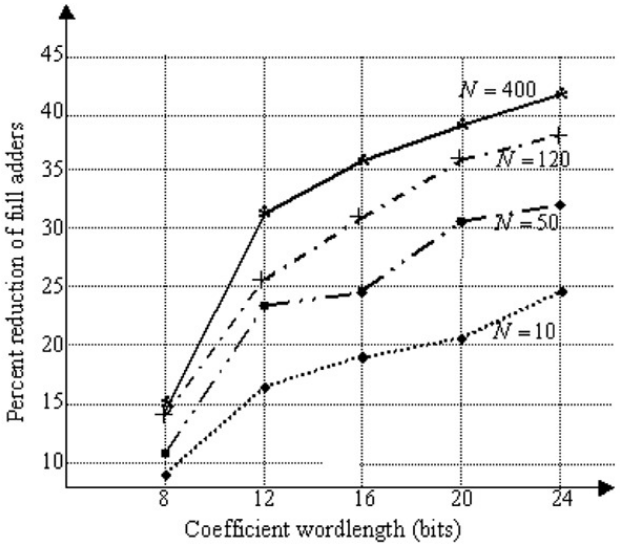


Fig. 9. Reduction of full adders achieved using proposed CP-HCSE technique over the HCSE method [11] for different number of FIR filter taps.

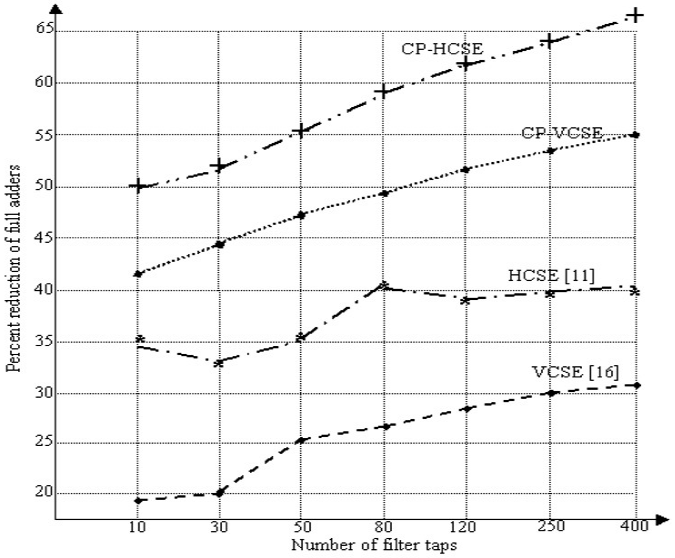


Fig. 10. Reduction of full adders over the direct method in designing the FIR filter of wider transition band with coefficient wordlength of 16 bits, for different number of filter taps.

Table IV Comparison of the reduction of FAs and critical path lengths in realizing the FIR filters in [15]

	FIR2			FIR4			FIR6		
Method	[11]	[15]	CP-SCSE	[11]	[15]	CP-SCSE	[11]	[15]	CP-SCSE
NFA	460	390	300	660	620	570	1234	1042	904
CPL	4	3	3	3	2	2	4	4	4

Table V Number of FAs required to implement the Elliptic IIR filters using *direct method*

Filter order *(N)*	Number of FAs in Direct Method				
	12-bit	16-bit	20-bit	24-bit	32-bit
5	320	589	858	1222	2007
7	437	805	1043	1589	2636
9	716	1117	1657	2295	3596
11	1075	1635	2331	3001	4485
13	1462	1956	2796	3514	5218
15	1826	2543	3339	4283	6624

Table VI Number of FAs required to implement the Elliptic IIR filters using HCSE and proposed CP-HCSE

N	Number of FAs in HCSE Method [11]					Number of FAs in CP-HCSE Method				
	12-bit	16-bit	20-bit	24-bit	32-bit	12-bit	16-bit	20-bit	24-bit	32-bit
5	261	475	637	875	1423	227	375	484	650	1022
7	314	588	798	1119	1866	278	498	615	820	1312
9	516	751	1067	1487	2402	455	623	837	1106	1722
11	789	1115	1559	2017	3104	706	943	1259	1558	2292
13	917	1297	1907	2425	3872	845	1112	1552	1912	2849
15	1200	1752	2324	2827	4769	1076	1647	1867	2214	3524

Table VII Number of FAs required to implement the Elliptic IIR filters using VCSE and proposed CP-VCSE

N	Number of FAs in VCSE Method [16]					Number of FAs in CP-VCSE Method				
	12-bit	16-bit	20-bit	24-bit	32-bit	12-bit	16-bit	20-bit	24-bit	32-bit
5	212	417	722	1078	1722	163	327	489	725	1158
7	267	525	883	1302	2206	213	423	664	906	1429
9	430	658	1276	1866	3035	365	593	901	1265	2000
11	659	956	1655	2278	3821	547	842	1366	1615	2565
13	823	1082	2170	2681	4206	737	972	1613	1954	2953
15	1057	1460	2468	3349	5571	887	1246	1990	2377	3796

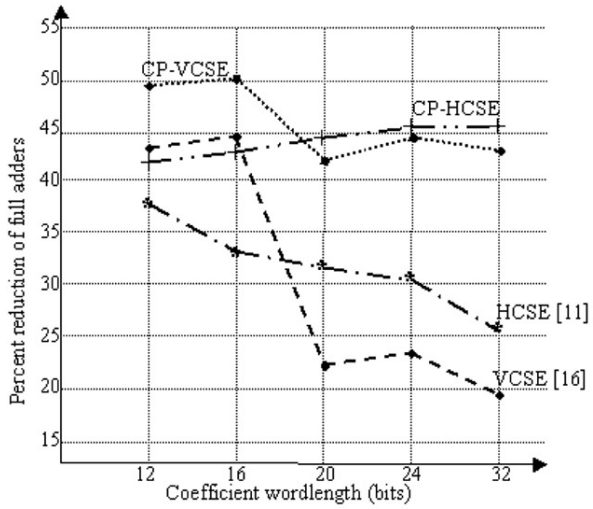


Fig. 11. Reduction of full adders achieved using various methods over the direct method in designing the elliptic IIR filter of order 13, for different wordlengths.