A Greedy Common Subexpression Elimination Algorithm for Implementing FIR Filters

S. Vijay$^+$, A.P.Vinod*, Edmund M-K.Lai$^@$
$^+$Dept. of Instrumentation and Control Engg., *School of Computer Engg., $^@$Institute of Information Sciences and Tech.
$^+$National Institute of Technology, Trichy, India, *Nanyang Technological University, Singapore,
$^@$Massey University, New Zealand, Email: $^+$imvijays@gmail.com, *asvinod@ntu.edu.sg, $^@$E.Lai@massey.ac.nz

Abstract— **The complexity of Finite Impulse Response (FIR) filters is dominated by the number of adders (subtractors) used to implement the coefficient multipliers. A greedy Common Subexpression Elimination (CSE) algorithm with a look-ahead method based on the Canonic Signed Digit (CSD) representation of filter coefficients for implementing low complexity FIR filters is proposed in this paper. Our look-ahead algorithm chooses the maximum number of frequently occurring common subexpressions and hence reduces the number of adders required to implement the filter. This adder reduction is achieved without any increase in critical path length. Design examples of FIR filters show that the proposed method offers an average adder reduction of about 20% over the best known CSE method.**

I. Introduction

FIR filters find extensive application in mobile communication systems due to its linear phase property and absolute stability. Low complexity and high speed digital filtering for mobile computing and communication applications require dedicated hard wired implementation of the filters. The number of additions (subtractions) used to implement the coefficient multiplier determines the complexity of FIR filters. Many approaches including coefficient coding using efficient arithmetic schemes, coefficient optimization techniques, distributed arithmetic techniques, read-only memory (ROM) - based designs, and common subexpression elimination (CSE) techniques have been proposed. Among these, the CSE techniques in [1]–[4] produced the best hardware reduction since it deals with multiplication of one variable (input signal) with multiple constants (coefficients). The goal of CSE is to identify multiple occurrences of identical bit patterns that are present in the CSD representation of coefficients, and eliminate these redundant multiplications. In [1], a graphical algorithm was proposed to identify and eliminate 2-bit subexpressions. A more efficient method was proposed in [2] which eliminated the most commonly occurring 2-bit subexpressions. As an additional criterion in the subexpression identification process, an estimation of a latch count improvement was also considered in [2]. A modification of the 2-bit CSE technique presented in [1] for identifying the “proper” patterns for elimination of common sub expression and to maximize the optimization impact was proposed in [3]. In [4], the technique in [2] was modified to minimize the logic depth (critical path length) into the digital structure. In [5], it has been shown that the Horizontal Common Subexpression Elimination (HCSE) technique offered better reduction of adders and Logic Depth (LD) than Vertical Common Subexpression Elimination (VCSE) in FIR implementations. The Bull-Horrock’s (BH) algorithm [6] and Reduced Adder Graph-*n*-dimensional (RAG-*n*) [7], used for the synthesis of filter coefficients are graph dependent, which produced multipliers with large logic depth (critical path length). A new Binary Subexpression Elimination (BSE) method was proposed in [8] using binary representation of filter coefficients that offered better adder reductions than previous CSD-based CSE methods. However the subexpression elimination in BSE [8] is not done by the most resourceful method because of the sequential checking and formation of binary bit patterns. This results in many bits being ungrouped and additional adders being required to implement them.

In this paper, we propose a CSE algorithm based on CSD representation of coefficients, which combines three techniques - the HCSE, the VCSE and the look-ahead technique. Our method provides adder reductions in case of filter coefficients with both smaller and larger word lengths. Moreover our technique does not increase the LD of the filter.

The rest of the paper is organized as follows. In Section II, we briefly review the BSE [8]. Section III shows an illustrative example of our method. Our CSE algorithm is presented in Section IV. In Section V, design examples of FIR filters and their comparisons are presented. Section VI provides our conclusions.

II. Binary Subexpression Elimination (BSE)

In BSE [8], three techniques are combined for reducing the number of adders - Binary Horizontal Subexpression

Elimination (BHSE), the Binary Vertical Subexpression Elimination (BVSE) and the hardwiring of the final stage adder. In the BHSE technique, the Binary Horizontal Common Subexpressions (BHCSs), x_6 to x_9, are formed from the binary representation of coefficients as follows.

$$[0\ 1\ 1] = x_6 = 2^{-1}x_1 + 2^{-2}x_1 \quad (1)$$

$$[1\ 0\ 1] = x_7 = x_1 + 2^{-2}x_1 \quad (2)$$

$$[1\ 1\ 0] = x_8 = x_1 + 2^{-1}x_1 \quad (3)$$

$$[1\ 1\ 1] = x_9 = x_1 + 2^{-1}x_1 + 2^{-2}x_1 \quad (4)$$

A direct realization of the BHCSs (1)-(4) would require 5 adders. But as x_8 can be obtained from x_6 by a shift operation and x_9 from x_8 using an adder, only 3 adders are required to realize the BHCSs (1)-(4) as shown by (5) and (6).

$$x_6 = 2^{-1}x_1 + 2^{-2}x_1 = 2^{-1}\,(\,x_1 + 2^{-1}x_1\,) = 2^{-1}x_8 \quad (5)$$

$$x_9 = x_1 + 2^{-1}x_1 + 2^{-2}x_1 = x_8 + 2^{-2}x_1 \quad (6)$$

In the BVSE technique, the basic BVCS [1 1] is taken and implemented as

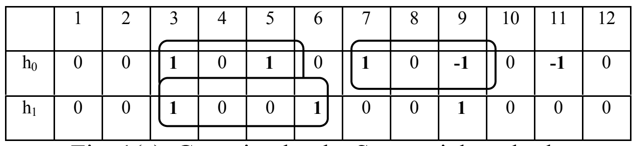
$$x_{10} = x_1 + x_1[-1] \quad (7)$$

The main disadvantage of the BSE [8] is that BHCSs are formed without a look-ahead and therefore many bits are left ungrouped after obtaining the BHCSs. Moreover, as the number of nonzero bits in binary representation is considerably larger than CSD representation, the BSE has the disadvantage of starting with a large number of nonzero bits in the optimization space compared to CSD-based CSE methods.

III. New Common Subexpression Method

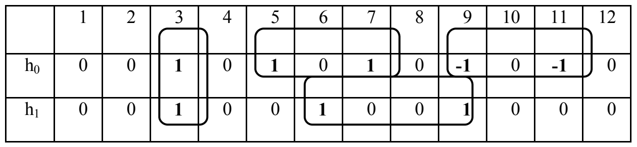
We propose an improved CSE method based on the CSD representation of the filter coefficients that maximizes the elimination of common subexpressions. The basic idea of our method is searching and selecting patterns with a look ahead to eliminate redundant Horizontal and Vertical Common Subexpressions (HCSs and VCSs) so that the subexpressions are maximized and least number of nonzero bits are left ungrouped. The grouping is done in such a way that the logic depth of the multiplier is kept minimal. As our CSE uses CSD, the initial optimization space has fewer numbers of nonzero bits unlike BSE [8].

The proposed CSE method can be illustrated using the example of a 12-tap FIR filter coefficients given in Table I. The patterns are selected based on a look-ahead method, as shown in Figures 1(a) and 1(b). Fig. 1(a) shows the conventional sequential subexpression formation for an example filter h_0 and h_1, whereas Fig. 1(b) shows the same fusing our look-ahead method. Note that there are two ungrouped bits in Fig 1(a), whereas all the bits are grouped in Fig 1(b), which minimizes the number of adders. The HCSs $x_3 = 3 = [1\ 0\ 1]$, $x_4 = 4 = [1\ 0\ \text{-}1]$, $x_5 = 5 = [1\ 0\ 0\ 1]$, $x_6 = 6 = [1\ 0\ 0\ \text{-}1]$ and VCS $x_2 = 2 = [1\ 1]$ and their negated versions are indicated inside rectangles in Table I.



	1	2	3	4	5	6	7	8	9	10	11	12
h_0	0	0	**1**	0	**1**	0	**1**	0	**-1**	0	**-1**	0
h_1	0	0	**1**	0	0	**1**	0	0	**1**	0	0	0

Fig. 1(a). Grouping by the Sequential method



	1	2	3	4	5	6	7	8	9	10	11	12
h_0	0	0	**1**	0	**1**	0	**1**	0	**-1**	0	**-1**	0
h_1	0	0	**1**	0	0	**1**	0	0	**1**	0	0	0

Fig 1(b). Grouping by the Look-Ahead method

TABLE 1. CSD REPRESENTATION OF FILTER COEFFICIENTS

	1	2	3	4	5	6	7	8	9	10	11	12
h_0	0	0	**1**	0	**1**	0	0	**-1**	0	**1**	**0**	**1**
h_1	0	0	**1**	0	**1**	0	**1**	0	0	**-1**	0	0
h_2	0	0	0	0	**1**	0	0	0	0	**1**	0	**1**
h_3	0	0	0	**1**	0	**1**	0	**1**	0	0	**1**	0
h_4	0	0	0	**1**	0	**1**	0	**-1**	0	0	**1**	0
h_5	0	**1**	0	0	**1**	0	**1**	0	0	**1**	0	0

Table II is obtained from Table I by substituting the respective pattern numbers in the respective bit positions, i.e., HCSs, [1 0 0 -1] = 6, [1 0 1] = 3, [1 0 0 1] = 5 and VCS, [1 1] = 2. Further, multiple occurrences of two HCSs with identical shifts between them or an HCS and a nonzero bit with identical shifts between them are grouped to form super-subexpressions (SSs). In Table II, the SS 8 is formed from the HCS [1 0 1] and the bit '1' with a shift difference of one between them (as in h_3) and the SS 9 is formed from the HCS [1 0 1] and the bit '-1' with a shift difference of one between them (as in h_4).

TABLE II. FINAL REPRESENTATION OF FILTER COEFFICIENTS

	1	2	3	4	5	6	7	8	9	10	11	12
h_0	0	0	**2**	0	**6**	0	0	0	0	**3**	0	0
h_1	0	0	0	0	**2**	0	**6**	0	0	0	0	0
h_2	0	0	0	0	0	0	0	0	0	**3**	0	0
h_3	0	0	0	**8**	0	0	0	0	0	0	**2**	0
h_4	0	0	0	**9**	0	0	0	0	0	0	0	0
h_5	0	**5**	0	0	0	0	**5**	0	0	0	0	0

From Table II, we can express the output of the example as :

$$y_k = 2^{-3}x_2 + 2^{-5}x_6 + 2^{-10}x_3 + 2^{-5}x_2[-1] + 2^{-7}x_6[-1] + 2^{-10}x_3[-2]$$
$$+ 2^{-4}x_9[-3] + 2^{-11}x_2[-3] + 2^{-4}x_9[-4] + 2^{-2}x_5[-5] + 2^{-7}x_5[-5] \quad (8)$$

The number of Multiplier Block Adders (MBAs) required to implement the filter using the direct method (method using shifts and adds) in Table I is 18. The proposed Greedy CSE method needs only 11 MBAs (6 for the subexpressions and 5 for the actual realization), which is a reduction of 39% over the direct method. The reduction percentage is larger when higher order filters are considered.

IV. The Proposed Greedy Algorithm

In this section, we explain the proposed CSE method. We make use of both HCSs and VCSs, but we take into account only [1 1] and [-1 -1] as the VCSs as we are able to completely exploit the symmetry of the coefficients (due their same sign). Our CSE 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 coefficients for desired word length.

Step 3: The algorithm checks for nonzero bits at (z, w), (z+1, w) , (z, w+2) and (z+1, w+2) where 'z' is the coefficient and 'w' is the bit position.

Case 1: When there is a HCS and VCS at (z, w), and when the nonzero bits at (z, w) and (z+1, w) are of the same sign:

(a) First the VCS at (z, w) is considered - the number of subexpressions present and the non pairable bits are found for the rest of the bits in the (z) coefficient.

(b) Then the HCS at (z, w) is considered and the same procedure as (a) is followed.

(i) The number of subexpressions and non-grouped bits are compared for both these procedures and the one with the largest number of patterns is chosen as the method to pair up the rest of that (z) coefficient.

(ii) If the number of subexpressions and non-grouped bits is the same, then the procedure considering the HCS is implemented as they are easier to realize.

(c) Depending on whether the VCS or HCS at (z, w) is chosen to group and form subexpressions, increment w by 1 or 3 respectively. If w <= N-1, go to step 3. Otherwise go to step 4.

(d) If the nonzero bits at (z, w) and (z+1, w) are not of the same sign, then the HCS at (z, w) is selected. Increment w. If w <= N-1, go to step 3. Else go to step 4.

Case 2: A similar procedure as illustrated above is used when there is a HCS at (z+1, w) and a VCS at (z, w), the nonzero bits at (z, w) and (z+1, w) being of the same sign:

(a) First, the VCS at (z, w) is considered. The number of subexpressions present and non grouped bits are found for the rest of the bits in the (z+1) coefficient.

(b) Then the HCS at (z, w) is considered and the same procedure as (b) is followed.

(i) The number of subexpressions and non-grouped bits are compared for both these procedures and the one with the largest number of patterns is chosen as the method to pair up the rest of that (z+1) coefficient.

(ii) If the number of subexpressions and non-grouped bits is the same, then the procedure considering the HCS is implemented as they are easier to realize.

(c) Depending on whether the VCS at (z, w) or HCS at (z+1, w) is chosen to group and form subexpressions, increment w by 1 or 3 respectively. If w <= N-1, go to step 3. Otherwise go to step 4.

(e) If the nonzero bits at (z, w) and (z+1, w) are not of the same sign, then the HCS at (z+1, w) is selected. Increment w. If w <= N-1, go to step 3. Else go to step 4.

Case 3: When only a HCS exists at (z, w), then select the HCS. Increment w. If w <= N-1, go to step 3. Otherwise go to step 4.

Case 4: When only a VCS exists at (z, w), then select the VCS. Increment w by 1. If w <= N-1, go to step 3. Otherwise go to step 4.

Step 4: When w > N-k, where k is the length of the pattern that is checked, set w = 1 and increment z by 1, go to step 3. When w > N-k, and z = (number of filter taps)/2, go to step 5.

Step 5: Once the HCSs and the VCSs are grouped, the coefficients are now checked for SSs like [1 0 1 0 1], [1 0 1 0 -1], [1 0 -1 0 1], [1 0 -1 0 -1] and their negated versions. Implement these SSs only if they occur at least twice in the coefficient matrix. This keeps a check on the LDs. When w > N-k, where k is the length of the pattern that is checked, set w = 1 and increment z by 1, go to step 5. When w > N-k, and z = (number of filter taps)/2, terminate the program.

V. Design Examples

In this section, we present examples of implementing several FIR filters of different length and frequency response specifications using the proposed algorithm and provide comparisons with the CSE [2] and the BSE [8] methods. FIR filters are designed using the Parks–McClellan algorithm.

Example 1: In this example, the filter pass-band and stop-band frequencies are 0.2π and $0.22\ \pi$ respectively. The comparison is done for different filter lengths of 20, 50, 80, 120, 200, 400 and 800 and for different wordlengths of 12, 16, 20 and 24 bits. Our proposed CSE gives a significant reduction of adders. Fig. 2 shows the comparison of reductions of adders achieved using our CSE method, the NR-SCSE [4] and the BSE [8] method, over the Hartley's CSE method [2], when the filter tap is 120 for wordlengths of 12, 16, 20 and 24 bits. The average adder reduction achieved using our method is 50.9% over the NR-SCSE [4] and 11.24% over BSE [8]. Overall, for all the filters in example 1, our method offers an average adder reduction of 56.6% over the CSE [2], 50.2% over the NR-SCSE [4] and 17.8% over the BSE [8]. The LDs of filters realized using our method are almost identical to that of BSE [8].

Example 2: In this example, the FIR filters employed in the filter bank channelizer of D-AMPS are considered as in [9]. Note that the decimation is moved to the left of the band-pass filters using the noble identity and the sampling rate chosen is 34.02MHz. The channel filters extract 30 kHz DAMPS channels from the input signal after down sampling by a factor of 350. The pass-band and stop-band edges are 30 kHz and 30.5 kHz respectively. The peak pass-band ripple is chosen as 0.1 dB. Table III shows the comparison of adders and LDs needed to implement the 610-tap filter corresponding to a stop-band attenuation of -65 dB for word lengths 12, 16, 20 and 24 bits, using our CSE method and the methods in NR-SCSE [4] and BSE [8].

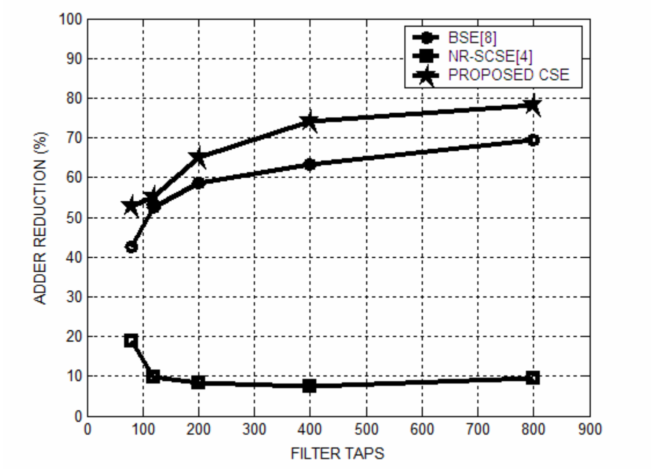


Fig. 2. Reduction of adders in designing the filters in example 1 for 16-bit word length.

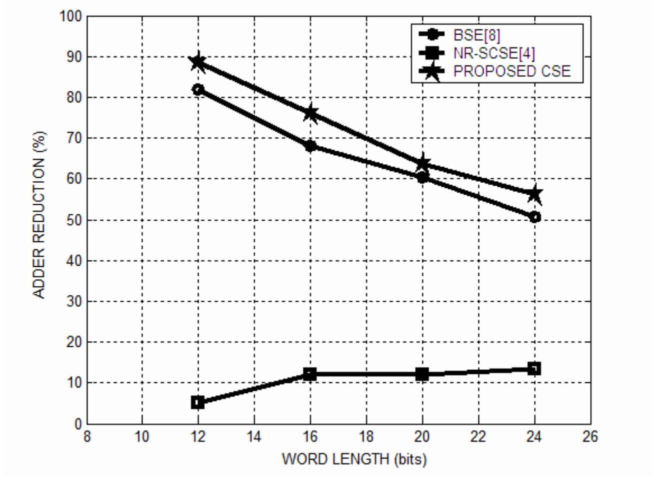


Fig. 3. Reduction of adders in designing the 610-tap D-AMPS filter for different word lengths.

From Table III, it can be seen that our method offers considerable reduction in the number of adders with almost no increase in the LDs.

TABLE III. COMPARISON OF THE D-AMPS CHANNEL FILTER WITH 610 TAPS FOR DIFFERENT METHODS

Word Length	CSE [2]		NR-SCSE [4]		BSE [8]		Proposed CSE	
	LO	LD	LO	LD	LO	LD	LO	LD
12	260	3	247	2	47	3	30	3
16	525	4	462	4	168	3	125	3
20	774	4	680	4	306	4	282	4
24	1007	4	872	5	496	4	441	5

The reduction of adders for the 610-tap filter for different word lengths is shown in Fig. 3. Overall, considering all the D-AMPS filters in example 2, our method offers an average adder reduction of 62.3% over the NR-SCSE [4] and 19.6% over the previously BSE [8]. The LDs of our method are almost same as that of BSE [8].

VI. CONCLUSIONS

We have presented a greedy CSE algorithm based on CSD representation of coefficients to implement low-complexity FIR filters. We have shown that the look-ahead method proposed by us maximizes the grouping of the subexpressions, thus leaving minimum number of unpaired nonzero bits. The average reduction of adders using our method is 20% over the best known CSE method (BSE [8]). The logic depths of filters implemented using our method is almost identical to that of BSE [8].

REFERENCES

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

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

[3] 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.

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

[5] A. P. Vinod and E. M-K. Lai, "Comparison of the horizontal and the vertical common subexpression elimination methods for realizing digital filters," in *Proc. of IEEE International Conference on Circuits and Systems,* 2005, pp. 496 – 499.

[6] D. R. Bull and D. H. Horrocks, "Primitive operator digital filters," *Proc. Inst. Elect. Eng.*, vol. 138, pt. B, no. 3, pp. 401–412, Jun. 1991.

[7] A. G. Dempster and M. D. Mcleod, "Use of minimum adder multiplier blocks in FIR digital filters," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 42, no. 9, pp. 569–577, Sep. 1995.

[8] R. Mahesh and A. P. Vinod, " A New Common Subexpression Elimination Algorithm For implementing Low Complexity FIR Filters in Software Defined Radio Receivers," in *Proc. IEEE International Symposium on Circuits and Systems,* vol. 4, pp. 4515-4518, May 21-24, 2006, Island of Kos, Greece.

[9] K.C. Zangi and R.D.Koilpillai, "Software radio issues in cellular base stations," *IEEE J. Select. Areas Commun.*, vol. 17, no.4, pp. 561-573, Apr.1999.