

# An Improved Non-CSD 2-Bit Recursive Common Subexpression Elimination Method to Implement FIR Filter

---

Hassan Kamal, Joohyun Lee, and Bontae Koo

The number of adders and critical paths in a multiplier block of a multiple constant multiplication based implementation of a finite impulse response (FIR) filter can be minimized through common subexpression elimination (CSE) techniques. A two-bit common subexpression (CS) can be located recursively in a non-canonical sign digit (CSD) representation of the filter coefficients. An efficient algorithm is presented in this paper to improve the elimination of a CS from the multiplier block of an FIR filter so that it can be realized with fewer adders and low logical depth as compared to the existing CSE methods in the literature. Vinod and others claimed the highest reduction in the number of logical operators (LOs) without increasing the logic depth (LD) requirement. Using the design examples given by Vinod and others, we compare the average reduction in LOs and LDs achieved by our algorithm. Our algorithm shows average LO improvements of 30.8%, 5.5%, and 22.5% with a comparative LD requirement over that of Vinod and others for three design examples. Improvement increases as the filter order increases, and for the highest filter order and lowest coefficient width, the LO improvements are 70.3%, 75.3%, and 72.2% for the three design examples.

**Keywords:** FIR filter, common subexpression elimination, logical operators, logical depth.

---

Manuscript received Nov. 8, 2010; revised Jan. 28, 2011; accepted Feb. 14, 2011.  
Hassan Kamal (phone: +82 42 860 0765, email: hassankamal@etri.re.kr, hassan23kamal@yahoo.com), Joohyun Lee (email: juehyun@etri.re.kr), and Bontae Koo (email: koobt@etri.re.kr) are with the Broadcasting & Telecommunications Convergence Research Laboratory, ETRI, Daejeon, Rep. of Korea.  
<http://dx.doi.org/10.4218/etrij.11.0110.0642>

## I. Introduction

Dedicated hardwired implementation is required for high-speed digital filters. Flexibility of the multiplier is of no use if the filters are designed for a specific application such as channel equalization or pulse shaping. A highly efficient filter structure can be generated by fixing the filter coefficients with dedicated hardware elements such as adders and shifters. Therefore, dedicated filter structures have been designed and used in great detail during the past years. Shift and addition operations are used to implement the multiplier operation of a filter, out of which the addition operation is more complex than the shift operation as it can be hard wired. The total number of adders, termed as logical operators (LOs), which are required for computing the partial products of an input signal multiplied with coefficients will determine the hardware requirements, and the number of adder steps will dictate the logic depths (LDs). LOs and LDs determine the complexity of the system. Multiple constant multiplication (MCM) is a transformation in which multiple constants are multiplied with one variable at a time. We use a transposed form of a finite impulse response (FIR) structure so that a multiplier block (MB) transforms into an MCM structure. We also use common subexpression elimination (CSE) to eliminate the redundant computations that exist in an MB by locating the most common bit patterns, called common subexpression (CS), existing in the binary representation of coefficients. Most authors have used canonical sign digit (CSD) representation to represent the coefficients [1]-[10]. In the CSD code of a number, each bit is set to 0, 1, or -1 so as to minimize the total number of nonzero

bits. This is a unique representation of a binary number with a minimum number of 1 and  $-1$  digits [11]. For fixed point filter coefficients, this conversion has to be done once for each coefficient. However, for variable filter coefficients, as in adaptive filters, the conversion has to be repeated for each change, and will hence add delay due to the conversion. Due to its ternary nature, each nonzero digit in a CSD representation requires an extra bit for the sign. This requires not only extra space but also the reallocation of space which affects the overall speed. Another issue in dealing with CSD numbers is the need for both addition and subtraction operations instead of only addition, as in case of 2's complement multiplication. Although the number of nonzero bits are reduced to nearly half [12], the number of combinations increases after the CSD transformation. This is because in CSD representation we deal with  $-1, 0$ , and  $1$ , and in 2's complement we deal with  $0$  and  $1$ . An increase in the number of combinations may lead to an increase in the number of CSs.

In our present work, we used a non-CSD representation of coefficients and introduced an improved CSE method to remove the redundancy in the representation. We also compare our method with the existing CSE methods that employ CSD representation and show that with an improved CSE method a non-CSD representation of coefficients requires either less or a comparable number of LOs to realize the FIR filter.

The remainder of this paper is divided as follows. In section II, we explain the MCM structure. In section III, we give a further explanation of the common subexpression. In section IV, we focus on existing methods. In section V, the CSE method is detailed. In section VI, we compare our results with existing methods. Finally, in section VII, we offer some concluding remarks.

## II. Multiple Constant Multiplication

An FIR filter involves a calculation of the following form:

$$Y_i = a_{ij} X_j, \quad (i = 0, 1, \dots, N-1 \text{ and } j = 0, 1, \dots, M-1), \quad (1)$$

where  $X_i$  and  $Y_i$  are input and output vectors, respectively;  $a$  is a set of constant filter coefficients;  $N$  is the number of coefficients; and  $M$  is the word length. Figure 1 shows an FIR structure and its transposed form for  $N=4$ . Figure 1(b) (transposed form) shows that an input  $X(n)$  is multiplied with constant filter coefficients. We explore this property of FIR filters and perform the input multiplication with the constant filter coefficients using adders and shifters.

## III. Common Subexpression

If implemented by adders and shifters, an MB of an FIR

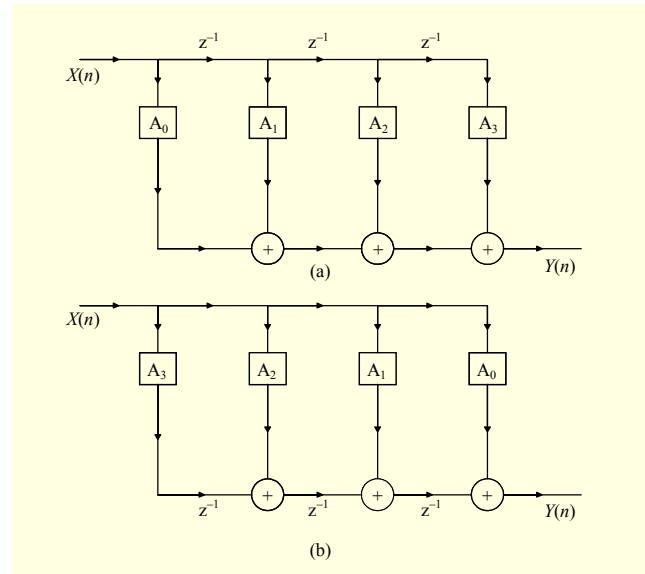


Fig. 1. (a) FIR filter and its (b) transposed form.

filter, as shown in Fig. 1(b), may have some redundant addition operations. Redundancy in the multiplier block of the FIR filter shown in Fig. 1(b) is to be removed by locating the common subexpressions and eliminating them from the design. The CSE method described here is based on finding a common subexpression of a 2-bit pattern recursively. Consider a four-tap FIR filter, as shown in Fig. 1, with filter coefficients as follows:

$$\begin{aligned} A_0 &= 00111011 = 2^0 + 2^1 + 2^3 + 2^4 + 2^5, \\ A_1 &= 00101011 = 2^0 + 2^1 + 2^3 + 2^5, \\ A_2 &= 10110011 = 2^0 + 2^1 + 2^4 + 2^5 + 2^7, \\ A_3 &= 11001010 = 2^1 + 2^3 + 2^6 + 2^7. \end{aligned}$$

Filter output  $Y(n)$  for any input  $X(n)$  can be written as

$$Y(n) = X(n)(2^0 + 2^1 + 2^3 + 2^4 + 2^5) + X(n)(2^0 + 2^1 + 2^3 + 2^5) + X(n)(2^0 + 2^1 + 2^4 + 2^5 + 2^7) + X(n)(2^1 + 2^3 + 2^6 + 2^7). \quad (2)$$

Select the two subexpressions as follows:

$$\begin{aligned} \text{Subexpression 1: } & 2^0 + 2^1, \\ \text{Subexpression 2: } & 2^3 + 2^5. \end{aligned}$$

Using the two subexpressions written above, a third subexpression is chosen as

$$\text{Subexpression 3: } 2^0 + 2^1 + 2^3 + 2^5.$$

Figure 2 shows the implementation of an MB of a 4-tap FIR filter using adders and shifters. We used the three subexpressions shown above to reduce the number of adders in building the MB. Without using the subexpressions, 14 adders were required to calculate the MB output, but using the

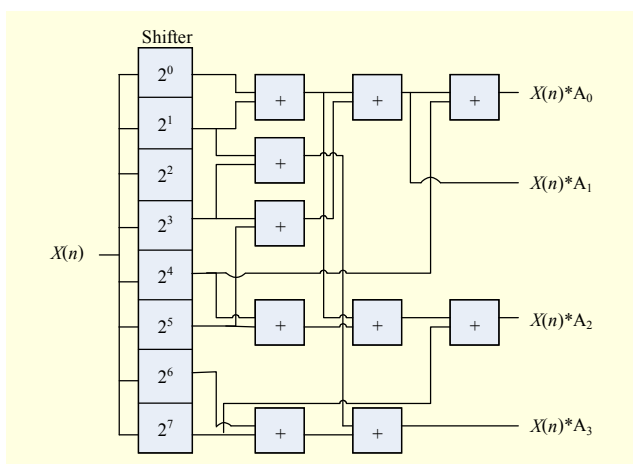


Fig. 2. Implementation of multiplier block of FIR filter.

subexpressions required only 10 adders; therefore, we could save 4 adders by eliminating the common subexpressions.

#### IV. Existing Methods

Most researchers have tried to minimize the complexity of an FIR filter by reducing the number of LOs and LDs of the multiplier block. The author in [4] uses the technique of 2-bit subexpression sharing and considers the number of delay latches in the circuit. The author attempts to minimize the number of adder and delay combinations. In [2], an algorithm is proposed for searching the nonzero 2-bit subexpression from the CSD representation of the coefficients.

In [13], the authors use differential coefficients to implement a multiplierless digital filter. While they used a graph-based approach, in terms of complexity reduction, CSE methods show better results. In [9], the authors reordered the computations and searched out common computations to maximize the sharing, but this method results in an increase in delay, corresponding to a delay of one adder step on average. The CSE algorithm used in [14] considers the redundancy in both the coefficients and MB, but the improvement of LO and LD over existing techniques was very small. Based on an ingenious graph synthesis approach, a CSE algorithm, called a contention resolution algorithm for weight-two horizontal subexpressions (CRA-2), has been developed [15]. In comparison to NR-SCSE [6], CRA-2 saves 1% to 2% more LOs. A graph representation of an MB used to reduce the number of LOs was utilized by the Bull-Horrocks algorithm (BHA) [16]. The Bull-Horrocks modified (BHM) algorithm and  $n$ -dimensional reduced adder graph (RAGn) algorithm, presented in [17], reduce the number of LOs further. Graph-representation-based techniques reduce the number of LOs required to implement an MB compared to the CSE methods

in [2], [4], [5], and [9]. However, the LDs of the resulting coefficient multipliers are considerably high. The algorithms in [15], [17], and [11] produce multipliers with large LDs which increase the delay of the multiplier substantially. The algorithms in [11], [16]-[18], consider one coefficient at a time and do not consider the effect on the rest of the coefficients while synthesizing them. A multiple adder graph algorithm (MAG) [19] minimizes the adder cost by considering the effect on the remaining coefficients. The algorithm in [20] proposed techniques for CSE optimization that are based on the extension of conventional 2-bit CSs to form three and four nonzero-bit super-subexpressions. The number of super subexpressions grows with the word length; hence, it is useful only for filter coefficients of higher word lengths. Routing complexity is also high in [20] as compared to [2], [4]-[6], and [9]. Common subexpression elimination (SBE) comes in two varieties, a horizontal CS (HCS) type existing within a coefficient, and a vertical CS (VCS) type existing across the coefficients. HCS elimination (HCSE) and VCS elimination (VCSE) can be combined to show better results than in [20]. In [8], the CSE method incorporating both VCSE and HCSE is shown to reduce the average area by 6.4% and 3.8% over the methods in [13] and [21], respectively.

The LD is reduced to 17.6% and 3.2%, respectively, compared to the results in [3] and [10]. By considering the implementation of the first half of the coefficients only, and assuming that the remaining half of the coefficients are symmetric to the first half so that their output can be shared, the authors in [3], [8], and [10], ignored the fact that VCSs put constraints on this sharing of output [1]. Hence, additional LOs are required to implement the second half of the coefficients as symmetry cannot be exploited fully if VCSs are used for CSE. Hence, the hardware requirement and logical depth shown in [3], [8], and [10] are not sufficiently accurate [1]. The constraints discussed in this paper are not applicable to anti-symmetric filters.

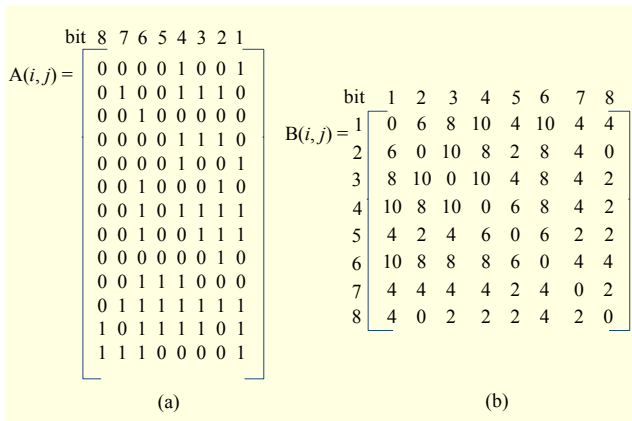
#### V. Proposed CSE Method

In this section, we explain our proposed CSE algorithm by using a design example. Consider a Parks-McClellan design of a low-pass FIR filter with a 26-filter order, passband, and stopband edges at  $0.2\pi$  and  $0.25\pi$ , respectively. The coefficient word length is 9 bits, where the most significant bit (MSB) is a sign bit and the remaining 8 bits are used for magnitude. Infinite-precision filter coefficients of the above mentioned specifications are obtained by using Matlab and their symmetric half is shown in Table 1.

**Step 1.** Binary matrix comprising filter coefficients is constructed:

**Table 1.** Coefficients of 26-tap Parks-McClellan linear phase FIR filter.

$h(0) = -0.00933078669575$	$h(1) = 0.07628237421426$
$h(2) = 0.03135623682714$	$h(3) = 0.01374432164657$
$h(4) = -0.00948598843682$	$h(5) = -0.03358586396879$
$h(6) = -0.04680063247432$	$h(7) = -0.03819695824263$
$h(8) = -0.00271831937636$	$h(9) = 0.05563093697248$
$h(10) = 0.12420551537587$	$h(11) = 0.18473033065671$
$h(12) = 0.22024453765020$	



**Fig. 3.** Generated matrices A and B from the filter coefficients shown in Table 1.

$$A(i, j) \quad (i=1, 2, 3, \dots, m \text{ and } j=1, 2, 3, \dots, n), \quad (3)$$

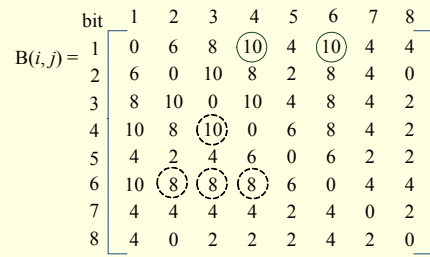
where  $m$  and  $n$  are the number of coefficients and bit width of the coefficients except for the sign bit. Rows stand for filter coefficients and columns for each coefficient bit. For the filter coefficients shown in Table 1, binary matrix A is shown in Fig. 3(a). Only the symmetric half of matrix A is shown.

**Step 2.** By using matrix A (refer to (3)), another diagonally symmetric square matrix B is constructed:

$$B(i, j) = \begin{cases} \sum_{k=1}^m A(k, i) * A(k, j), & \text{for } (i \neq j), \\ 0, & \text{for } (i = j), \end{cases} \quad (4)$$

$(i = 1, 2, 3, \dots, n; j = 1, 2, 3, \dots, n)$ ,

where  $m$  and  $n$  are the number and bit width of the coefficients except for the sign bit, and ‘\*’ represents a multiplication sign. The equation yields a square matrix B in which an element  $B(i, j)$  for ‘ $k$ ’ varying from 1 to  $m$ , is the number of ‘1’  $i$ -th and  $j$ -th bits (from all coefficients) simultaneously. Figure 3(b) shows a diagonally symmetric  $8 \times 8$  square matrix  $B(i, j)$ , ( $i=1, 2, 3, \dots, 8; j=1, 2, 3, \dots, 8$ ) from coefficient matrix A (step 1),



**Fig. 4.** Matrices B of filter coefficients shown in Table 1.

where  $m=26$  and  $n=8$ . Each element of matrix B is the weightage to establish a subexpression by two bits (corresponding to the row and column of that element in matrix B) of matrix A.

$$\text{Step 3. } [i, j] = \text{findmag}(\max(B)); \quad (5)$$

where  $i$  and  $j$  are the row and column indices of the highest integer in matrix B, respectively, ‘findmag’ is an operator used to find the magnitude, and ‘max’ is the maximum operator, meaning it locates the largest integers. By searching the highest integer in matrix B, we locate any two columns that together have the maximum number of ‘1’s (both columns have ‘1’ for the same row, also referred to as occurrences) with respect to other columns in matrix A. If ‘ $i$ ’ and ‘ $j$ ’ are two such columns, then we combine bits ‘ $i$ ’ and ‘ $j$ ’ of the coefficients to establish a subexpression. Matrix B in Fig. 3(b) has a maximum magnitude of 10. However, there is a possibility that any column of matrix A (for example, ‘ $i$ ’) might have the maximum number of ‘1’s with more than one column. As can be noticed in matrix B (refer to (4)) for  $i=1$ , the maximum magnitude is 10 when  $j=4$  and  $j=6$ . Similarly, for  $i=3$  and  $j=(2, 4)$ , the maximum magnitude is 10. This means bit 1 can be used to make a subexpression with bit 4 and bit 6, and bit 3 with bit 2 and bit 4. However, we can combine two bits at a time to make a pair; hence, we need to select any one bit to establish a subexpression with bit ‘ $i$ ’. This process is repeated until the largest magnitude of matrix B remains greater than or equal to 2. If the largest magnitude is less than 2, the algorithm is terminated.

**Step 4.** Suppose  $P(1, 2, \dots, k)$  are such  $k$  bits that claim to build a subexpression with bit ‘ $i$ ’ on the basis of the highest integer of matrix B.  $P(1, 2, \dots, k)$  can be located using

$$[i, P(1, 2, \dots, k)] = \text{findindex}(\max(B(i, 1:n))), \quad (6)$$

where ‘findindex’ is used as an operator to find the index, ‘max’ is the maximum operator, and ‘ $n$ ’ is the bit width of the coefficients except for the sign bit.

Referring to Fig. 4, if  $i = 1$ , then  $k = 2$  and  $P(1, 2) = (4, 6)$ .  $B(1, P(1)) = 10$  and  $B(1, P(2)) = 10$  (shown by the full circle). Now referring to (7), we locate the largest occurrences of bits

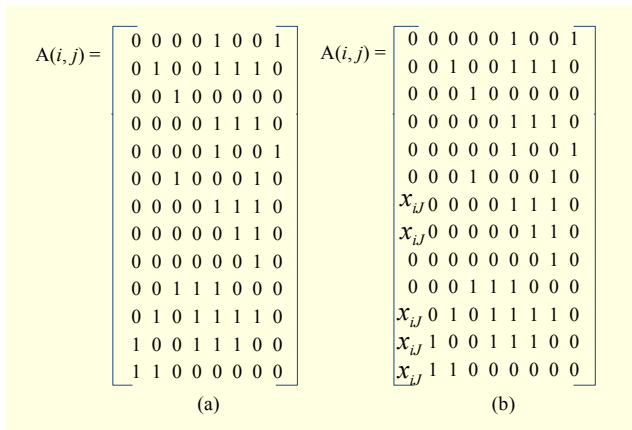


Fig. 5. Matrices A after the steps 4 and 5, respectively.

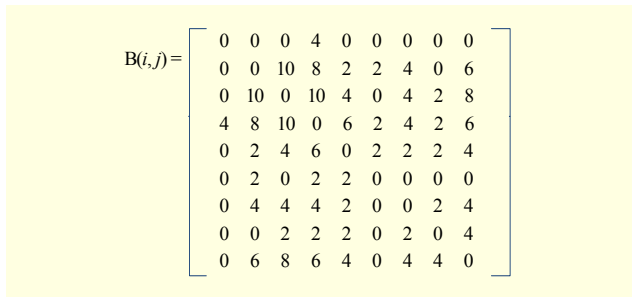


Fig. 6. Matrix B generated in second iteration at step 2.

$P(1, 2, \dots, k)$  with other bits except bit 'i' by the help of matrix B.

$$Q(1, 2, \dots, k) = \text{findmag}(\max(\text{B}(P(1, 2, \dots, k), t))), \quad (7)$$

where  $t \neq i$  and varies from 1 to  $n$ . In matrix B, shown in Fig. 4,  $Q(1) = 10$  for  $P(1) = 4$  and  $Q(2) = 8$  for  $P(2) = 6$  where  $Q(1)$  and  $Q(2)$  are shown in the dotted circles. Once the largest occurrences of bits  $P(1, 2, \dots, k)$  with remaining bits (except bit 'i') are located in Q, we search the index of minimum Q in P.

$$\begin{aligned} R &= \text{findindex}(\min(Q(1, 2, \dots, k))), \\ J &= P(R). \end{aligned} \quad (8)$$

By searching the index of minimum Q in P, we search for a bit whose occurrences with bit 'i' is highest and lowest with any other bit in matrix B (see (8)). In Fig. 4, for  $Q(1, 2)=[10, 8]$ , we get  $R=2$  and  $J=6$ . Hence, bit  $i=1$  can make a subexpression with bit  $J=6$ . Referring to matrix A in step 1, bits  $i$  and  $J$  are selected to make a subexpression. Once a subexpression is chosen, matrix A is updated.

**Step 5.** Matrix A is updated as follows: If  $A(t, i) = A(t, J) = 1$ ; for  $t=1$  to  $m$ .

Then,  $A(t, i) = A(t, J) = 0$ ; for  $t=1$  to  $m$ . From the design example, we have  $i=1$ ,  $J=6$ , and  $m=8$ . From step 1, matrix A is updated as shown in Fig. 5(a).

**Step 6.** At  $n = n+1$ , a new column is created in matrix A as

shown in Fig. 5(b). If  $A(t, i) = A(t, J) = 1$ , then  $A(t, n) = x_{i,J}$ , for  $t=1$  to  $m$ , where  $x_{i,J} = 2^{(i-1)} + 2^{(J-1)}$ . The new column represents the subexpression generated at step 4. Once a new column is appended to matrix A, the process is restarted from step 2. Hence, the subexpressions are searched recursively until all possible subexpressions are formed. For the design example, a subexpression identified in terms of bit 1 and bit 6 can be written as  $2^0 + 2^5$ . We move to step 2 to search the next subexpression until the highest magnitude in matrix B is greater than 1. A newly formed matrix B is shown in Fig. 6.

The highest magnitude in matrix B is 10. Bit 2 has magnitude 10 with bit 3 and vice versa. Bit 3 has magnitude 10 with bit 4 and vice versa. We can either take bit 2 or bit 3 first and move to step 4.

## VI. Comparison

In [1], Vinod and others analyzed the impact of HCSE and VCSE in exploiting the symmetry of FIR coefficients. The authors demonstrated an optimization algorithm to reduce the LO and LD requirements. They claimed that the algorithm produced the best reduction in number of LOs compared to the best known algorithm in the literature without increasing the LD requirement. The results in [1] (in terms of LOs and LD) have been compared with several CSE methods, such as Hartley [4], Pasko [5], BHM [17], C1 algorithm [22], MAG [19], NR-SCSE [6], HCUB [18], CSDC [21], and CRA-2 [15], [9]. The authors in [1] also presented design examples of several FIR filters using the proposed common subexpression algorithm. As in [1], we also used the Parks-McClellan algorithm to design FIR filters and compared our results with [1] for the design of FIR filters described in examples 1, 2, and 3 [1]. By doing so, our results can also be compared with other CSE methods mentioned in [1]. Filter coefficients of the specifications mentioned in the following examples were obtained through the help of Matlab.

**Example 1.** Here, we compare our results (in terms of LOs and LDs) with [1] (example 1) for filters FIR1 to FIR5. FIR1 has passband and stopband frequencies of  $0.15\pi$  and  $0.25\pi$ , respectively [23]. For FIR2, the passband and stopband frequencies are  $0.021\pi$  and  $0.07\pi$ , respectively [23]. FIR3 is a high-pass filter L1 with stopband and passband frequencies of  $0.37\pi$  and  $0.5\pi$  [24], respectively. FIR4 is employed in a digital advanced mobile phone system (D-AMPS) receiver with passband and stopband frequencies of  $0.6173\pi$  and  $0.6276\pi$ , respectively. FIR5 is employed in a personal digital cellular (PDC) receiver with passband and stopband frequencies of  $0.6836\pi$  and  $0.6973\pi$ , respectively. The authors in [1] showed an average LO reduction of 10.2% over the NR-SCSE [6] method. Over Hartley [4], Pasko [5], BHM [17], C1

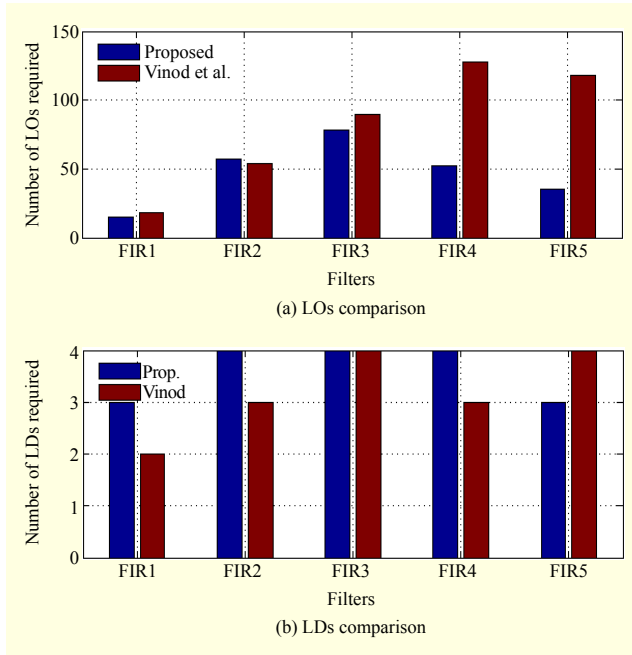


Fig. 7. Comparison of LOs and LDs for example 1.

algorithm [22], and MAG [19], the average LO reductions are 22.4%, 16.1%, 12.9%, 6.7%, and 4.4%, respectively. The LD in [1] is either less than or comparable with other methods in

Table 2. Comparison of proposed method with that of Vinod and others in terms of LOs and LDs for example 1.

Filter	N	W	Vinod et al. 2009		Proposed method		Improvement	
			LOs	LDs	LOs	LDs	LOs (%)	LDs number
FIR1	25	9	18	2	15	3	16.6	+1
FIR2	59	14	54	3	57	4	-5.0	+1
FIR3	120	17	90	4	78	4	13.3	0
FIR4	200	13	128	3	52	4	59.0	+1
FIR5	230	12	118	4	35	3	70.3	-1

most cases. These comparison results with their filter specifications (as mentioned above) are directly taken from [1]. Table 2 shows the comparison results, with an average 30.8% improvement in LO reduction by our algorithm over that in [1]. The LDs of the proposed filters are comparable to those in [1] in most cases. Figure 7 gives a clearer view of the higher order filters (FIR4 and FIR5), which showed an average improvement of 56.8% in the reduction of LOs with the same average LDs.

**Example 2.** Here, we compare our results (in terms of LOs

Table 3. Comparison of number of logical operators needed to realize the FIR filter in example 2.

Filter length	12 bit			16 bit			20 bit			24 bit		
	Vinod et al. 2009	Proposed method	Improvement (%)	Vinod et al. 2009	Proposed method	Improvement (%)	Vinod et al. 2009	Proposed method	Improvement (%)	Vinod et al. 2009	Proposed method	Improvement (%)
20	24	22	8.3	28	36	-28.6	34	48	-41.0	46	69	-50.0
50	37	32	13.51	55	59	-7.0	66	85	-28.8	84	109	-29.8
80	52	35	32.7	67	72	-7.5	92	111	-20.1	120	151	-20.4
120	76	36	52.6	106	94	11.32	150	144	4.0	184	203	-10.3
200	102	34	66.7	150	103	31.33	198	200	-1.0	260	278	-6.9
400	150	37	75.3	270	121	55.19	368	269	26.9	494	434	12.1

Table 4. Comparison of number of logical operators needed to realize the FIR filter in example 3.

Filter length	PSR (dB)	16 bit			20 bit			24 bit		
		Vinod et al. 2009	Proposed method	Improvement (%)	Vinod et al. 2009	Proposed method	Improvement (%)	Vinod et al. 2009	Proposed method	Improvement (%)
200	-24	176	121	31.30	229	208	9.17	290	291	-0.34
460	-48	316	139	56.00	430	320	25.58	549	500	8.93
610	-65	376	143	61.97	538	361	32.90	706	591	16.29
940	-85	481	144	70.06	720	400	44.44	980	721	26.43
1,180	-96	536	149	72.20	856	416	51.40	1,170	777	33.59

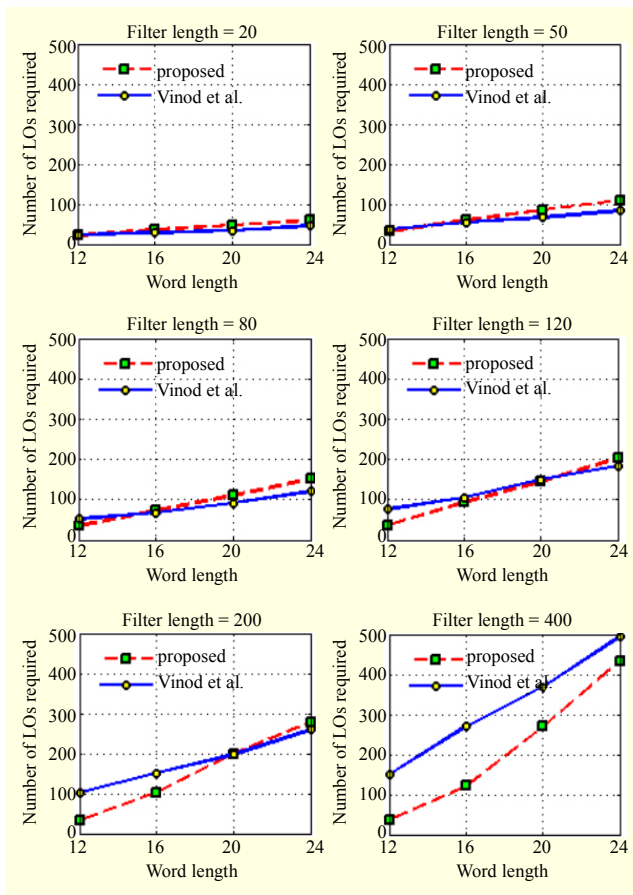


Fig. 8. Comparison of LOs for example 2.

and LDs) with those of Vinod and others [1] (example 2) for an FIR filter with passband and stopband frequencies of  $0.2\pi$  and  $0.22\pi$ , respectively. For different filter lengths (20, 50, 80, 120, 200, 400), the coefficient word lengths were varied (12, 16, 20, 24 bits) to compare the LO and LD requirements. For filter lengths of 20, 50, 80, 120, 200, and 400, the average reductions in LO achieved by [1] as compared with [9] is 9.7%, while reductions of 12%, 13%, 18.5%, 5.6%, and 30% were achieved for CRA-2 [15], NR-SCSE [6], CSDC [21], MAG [19], and C1 [22], respectively. These comparison results are directly taken from [1]. The comparison between the proposed CSE algorithm and [1] is shown in Table 3. The average LO reduction achieved using our method over that of Vinod and others is 5.5%. For 12-bit coefficient word lengths, our algorithm showed an improvement of 41.5% (averaged for all filter lengths) over [1], but this improvement increases gradually as the filter length increases. The highest improvement is 75.3% for a filter length of 400 and 12-bit coefficient word lengths. Figure 8 shows a comparison of the LOs needed to implement the FIR filters for different filter lengths. For higher order filters of 120, 200, and 400, our CSE algorithm shows an average improvement of 26.44% over [1].

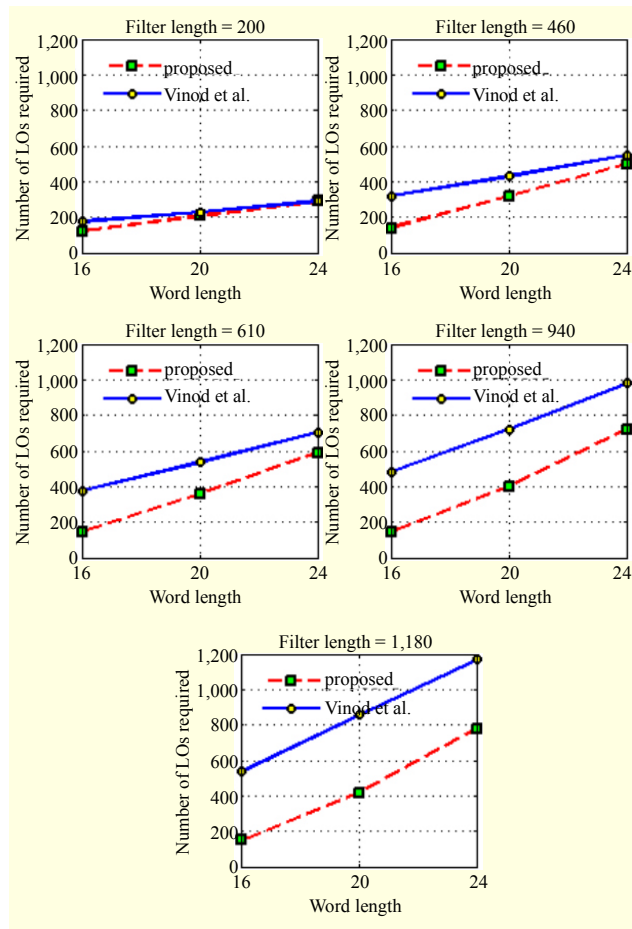


Fig. 9. Comparison of LOs for example 3.

For low filter order and higher coefficient widths, [1] shows better results (LO reduction) than our algorithm.

A comparison of LDs is not shown here due to a lack of space, but the reduction of LDs achieved by our CSE algorithm is comparable to [1] in all cases.

**Example 3.** We compared the results of example 3 [1] with the results produced by our method. We considered FIR filters employed as channel filters with a large number of taps. The sampling rates chosen were 34.02 MHz as in [18]. The channel filters extracted 30 kHz D-AMPS channels from the input signal after down-sampling by a factor of 350. The passband and stopband edges were 30 kHz and 30.5 kHz, respectively. The peak pass band ripple was chosen as 0.1 dB. Filters of lengths 200, 460, 610, 940, and 1,180 were chosen corresponding to peak stopband ripple (PSR) specifications of  $-24$  dB,  $-48$  dB,  $-65$  dB,  $-85$  dB, and  $-96$  dB, respectively. All these specifications mentioned above are directly taken from [1]. The method in [1] offers an average LO reduction of 17.6% over [9], 17.9% over CRA-2 [15], 20.2% over NR-SCSE [6], 26.9% over CSDC [21], and 10% and 12.8% over [19] and [22], respectively. Thus, [1] offers the best tradeoff in

terms of the number of LOs and LDs when compared to other existing CSE methods. We compared the LO reductions of an FIR filter between [1] and our method, and our method showed an average improvement of 22.5% over [1] in terms of LO requirements. A comparison between our proposed CSE algorithm and [1] in terms of LOs is shown in Table 4. Referring to Table 4, for coefficient widths of 16 bits, 20 bits, and 24 bits, improvement over [1] increases from 31.3% to 72.2%, 9.17% to 51.4%, and -0.34% to 33.6% for filter lengths of 200 to 1,180, respectively. Referring to Fig. 9, for almost all of the filter lengths and coefficient word lengths mentioned, our CSE algorithm consumes less LOs than [1] when implementing an FIR filter.

## VII. Conclusion

In our present work, we compared the adder (LO) and critical path (LD) reduction obtained by our non-CSD 2-bit recursive CSE algorithm used for implementing FIR filters with a method proposed in [1]. It was earlier suggested that a CSD representation of filter coefficients with a vertical and horizontal common subexpression elimination method helps in reducing hardware costs. Further, we represented a method that uses a non-CSD representation of filter coefficients with a recursive 2-bit CSE scheme to remove the redundancy in the multiplier block of a filter design. We compared our algorithm with that of Vinod and others [1]. Our method produces an average of 22.5%, 5.5%, and 30.8% improvement over [1] in terms of LO reductions for examples 3, 2, and 1, respectively. This improvement increases as the filter order increases, and for the highest filter order and lowest coefficient width, LO improvements over [1] were 70.3%, 75.3%, and 72.2% for the three design examples, respectively. Our algorithm is highly effective over [1] for higher filter orders and lower coefficient widths. The LD reduction achieved by our algorithm is comparable with [1] in most cases.

## References

- [1] A.P. Vinod et al., "An Improved Common Subexpression Elimination Method for Reducing Logic Operators in FIR Filter Implementations without Increasing Logic Depth," *INTEGRATION, VLSI J.*, vol. 43, 2010, pp. 124-135.
- [2] M. Mehendale, S.D. Sherlekar, and G. Venkatesh, "Synthesis of Multiplier-less FIR Filters with Minimum Number of Additions," *Proc. IEEE/ACM Int. Conf. Comput.-Aid. Design*, 1995, pp. 668-671.
- [3] A.P. Vinod et al., "FIR Filter Implementation by Efficient Sharing of Horizontal and Vertical Common Subexpressions," *Electron. Lett.*, vol. 39, no. 2, Jan. 2003, pp. 251-253.
- [4] R.I. Hartley, "Subexpression Sharing in Filters Using Canonic Signed Digit Multipliers," *IEEE Trans. Circuits Syst. II*, vol. 43, no. 10, Oct. 1996, pp. 677-688.
- [5] R. Pasko et al., "A New Algorithm for Elimination of Common Subexpressions," *IEEE Trans. Comput.-Aid. Design Integ. Circuits Syst.*, vol. 18, no. 1, Jan. 1999, pp. 58-68.
- [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, Mar. 2002, pp. 196-203.
- [7] Y. Takahashi and Michio Yokoyama, "A Comparison of Multiplierless Multiple Constant Multiplication Using Common Subexpression Elimination Method," *51st Midwest Symp. Circuits Syst.*, 2008, pp. 298-301.
- [8] Y. Takahashi and M. Yokoyama, "New Cost-Effective VLSI Implementation of Multiplierless FIR Filter using Common Subexpression Elimination," *Proc. Int. Symp. Circuits Syst.*, vol. 2, May 2005, pp. 1445-1448.
- [9] H. Choo, K. Muhammad, and K. Roy, "Complexity Reduction of Digital Filters Using Shift Inclusive Differential Coefficients," *IEEE Trans. Signal Process*, vol. 52, no. 6, June 2004, pp. 1760-1772.
- [10] Y.B. Jang and S.J. Yang, "Low-Power CSD Linear Phase FIR Filter Structure Using Vertical Common Sub-expression," *Electron. Lett.*, vol. 38, no. 15, July, 2002, pp. 777-779.
- [11] F. Xu, C.H. Chang, and C.C. Jong, "Modified Reduced Adder Graph Algorithm for Multiplierless FIR Filters," *Electron. Lett.*, vol. 41, no. 6, Mar. 2005, pp. 302-303.
- [12] R. Hashemian, "A New Method for Conversion of a 2's Complement to Canonic Signed Digit Number System and its Representation," *30th Asilomar Conf. Signals Syst. Comput.*, vol. 2, 1996, pp. 904-907.
- [13] K. Muhammad and K. Roy, "A Graph Theoretic Approach for Synthesizing very Low-Complexity High-Speed Digital Filters," *IEEE Trans. Comput.-Aid. Design Integ. Circuits*, vol. 21, no. 2, Feb. 2002, pp. 204-216.
- [14] C.Y. Yao et al., "A Novel Common-Subexpression-Elimination Method for Synthesizing Fixed-Point FIR Filters," *IEEE Trans. Circuits Syst. I*, vol. 51, no. 11, Nov. 2004, pp. 2215-2221.
- [15] F. Xu, C.H. Chang, and C.C. Jong, "Contention Resolution Algorithm for Common Subexpression Elimination in Digital Filter Design," *IEEE Trans. Circuits Syst. II*, vol. 52, no. 10, Oct. 2005, pp. 695-700.
- [16] D.R. Bull and D.H. Horrocks, "Realisation Techniques for Primitive Operator Infinite Impulse Response Digital Filters," *Proc. Int. Symp. Circuits Syst.*, vol. 1, May 1993, pp. 607-610.
- [17] A.G. Dempster and M.D. Mcleod, "Use of Minimum-Adder Multiplier Blocks in FIR Digital Filters," *IEEE Trans. Circuits Syst. II*, vol. 42, no. 9, Sept. 1995, pp. 569-577.
- [18] Y. Voronenko and M. Pushcel, "Multiplierless Multiple Constant

Multiplication,” *ACM Trans. Algorithms*, vol. 3, no. 2, 2007.

- [19] J.H. Han and I.C. Park, “FIR Filter Synthesis Considering Multiple Adder Graphs for a Coefficient,” *IEEE Trans. Comput.-Aid. Design Integrat. Circuits Syst.*, vol. 27, no. 5, May 2008, pp. 958-962.
- [20] 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 Trans. Comput.-Aid. Design Integ. Circuits Syst.*, vol. 24, no. 2, Feb. 2005, pp. 295-304.
- [21] Y. Wang and K. Roy, “CSDC: A New Complexity Reduction Technique for Multiplierless Implementation of Digital FIR Filters,” *IEEE Trans. Circuits Syst. I*, vol. 52, no. 9, Sept. 2005, pp. 1845-1853.
- [22] A.G. Dempster, S.S. Dimirsoy, and I. Kale, “Designing Multiplier Blocks with Low Logic Depth,” *Proc. IEEE Int. Symp. Circuits Syst.*, vol. 5, May 2002, pp. 773-776.
- [23] H. Samuelli, “An Improved Search Algorithm for the Design of Multiplierless FIR Filters with Powers-of-Two Coefficients,” *IEEE Trans. Circuits Syst.*, vol. 36, no. 7, July 1989, pp. 1044-1047.
- [24] Y.C. Lim and S.R. Parker, “Discrete Coefficient FIR Digital Filter Design Based upon LMS Criteria,” *IEEE Trans. Circuits Syst.*, vol. 30, no. 10, Oct. 1983, pp. 723-739.



**Bontae Koo** received the BS and MS in electrical engineering from Korea University, in 1989 and 1991, respectively. He was with Hyundai Electronics, Rep. of Korea, from 1991 to 1997, and ASPEC, in San Jose, USA, from 1997 to 1999. He joined ETRI, Rep. of Korea, in 1999. Currently he serves as the team leader for the Application SoC Development Team.



**Hassan Kamal** received his MS from the Department of Instrumentation Engineering from Indian Institute of Science (IISc), Bangalore, India, in 2009. Since 2009, he has been working at ETRI, Daejeon, Rep. of Korea. His research interests include digital circuit design, biomedical signal processing, and LTE based FemtoCell technology.



**Joohyun Lee** received his MS in electrical engineering from Pohang University of Science and Technology (POSTECH), Rep. of Korea, in 1998. In 1998, he joined advanced DRAM design team of Hynix in Korea. Since 2000, he has been with ETRI, Daejeon, Rep. of Korea. His research interests include synchronization of OFDM receivers, multimedia broadcasting, and LTE-based FemtoCell technology.