Design of High-Speed Multiplierless Filters Using a Nonrecursive Signed Common Subexpression Algorithm

Marcos Martínez-Peiró, Eduardo I. Boemo, and Lars Wanhammar, *Member, IEEE*

***Abstract*—In this work, a new algorithm called nonrecursive signed common subexpression elimination (NR-SCSE) is discussed, and several applications in the area of multiplierless finite-impulse response (FIR) filters are developed. While the recursive utilization of a common subexpression generates a high logic depth into the digital structure, the NR-SCSE algorithm allows the designer to overcome this problem by using each subexpression once. The paper presents a complete description of the algorithm, and a comparison with two other well-known options: the graph synthesis, and the classical common subexpression elimination technique. Main results show that the NR-SCSE implementations of several benchmark circuits offer the best relation between occupied area and logic depth respect to the previous values published in the technical literature.**

***Index Terms*—Common subexpression elimination, finite-impulse response (FIR) filtering, multiplierless algorithm.**

I. Introduction

THE multiplication of a variable (data input) by a set of constants (finite-impulse response (FIR) filtering, DCT, FFT, etc.) is a central operation in video processing, digital television, data transmission, and wireless communications. The area-time optimization of this operation has often been accomplished by using a shift-and-add multiplication algorithm, combined with techniques to reduce the number of nonzero bits in the binary representation of the coefficients. For example, signed-digit (SD) code was efficiently applied in [1] and [2] to reduce circuit area. Moreover, an additional area saving can be obtained if the common subexpression elimination (CSE) method introduced in [1] is also utilized [2], [3]. Main ideas of the CSE method are shown in Fig. 1. In this example, the implementation of the coefficient 10100101 requires only one subexpression (101) and two additions [Fig. 1(b)] compared to a shift-and-add that requires three additions [Fig. 1(a)]. In terms of speed, the logic depth is diminished from three to two adders. This fact leads to an important reduction of the data propagation time through the whole filter structure. The use of a register, the triangle in a box in [Fig. 1(b)], before the coefficient implementation allows the subexpressions to be pipelined. The number of logical operators (LO) is further reduced if common subexpressions can be shared between various coefficients (Fig. 2).

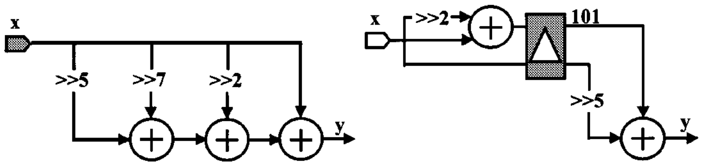

Fig. 1. Implementation of the coefficient 10100101 by using the CSE method.

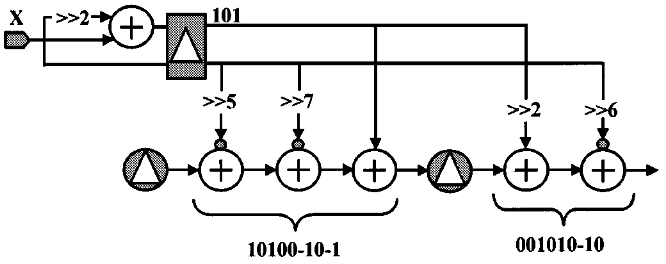

Fig. 2. CSE with a pipeline implementation of two coefficients.

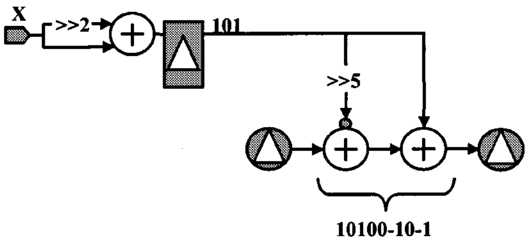

Fig. 3. Coefficient $10100\underline{1}0\underline{1}$ using a SCS.

The CSE idea has been also used to share signed subexpressions (SCSE). Fig. 3 shows the reduction in the previous implementation by using 101 or $\underline{1}0\underline{1}$ (where $\underline{1} = -1$) as signed common subexpression (SCS). Compared to the design in Fig. 2, the SCSE algorithm saves a logic operator in the implementation of the coefficient $10100\underline{1}0\underline{1}$.

Another alternative to save area is to use graph synthesis methods [4], [5], [7], [8]. These methods depart from a set of integer coefficients to create a dependence graph. The graph obtains each new coefficient from the previous ones and shift operations (powers of two) of the data input. For instance, the coefficient 93 is obtained as $(3 = 2 + 1, 93 = 3 \times 2^5 - 3)$ by using shifts of the data input, together with subtractions and

Manuscript received November 16, 2001; revised March 14, 2002. This work was supported in part by the Universidad Politécnica de Valencia and in part by the Spanish Ministry of Science and Technology under Contracts TIC2000-1151-C07-05 and TIC2001-2688-C03-03.

M. Martínez-Peiró is with the School of Telecommunication Engineering, Department of Electronic Engineering, Universidad Politecnica de Valencia, 46020 Valencia, Spain (e-mail: mpeiro@eln.upv.es).

E. I. Boemo is with the School of Computer Engineering, Universidad Autonoma de Madrid, 28049 Madrid, Spain (e-mail: eduardo.boemo@uam.es).

L. Wanhammar is with the Electronics Systems, Department of Electrical Engineering, Linkoping University, S-581 83 Linkoping, Sweden (e-mail: larsw@isy.liu.se).

Publisher Item Identifier S 1057-7130(02)06232-8.

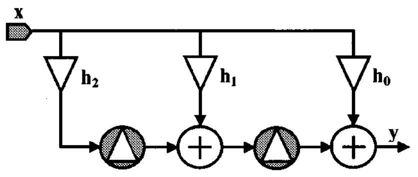

Fig. 4. Transposed structure of a FIR filter with three coefficients.

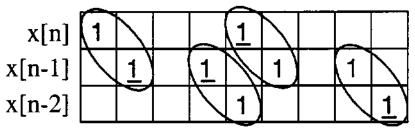

Fig. 5. Four occurrences of the same common subexpression in three coefficients FIR filter.

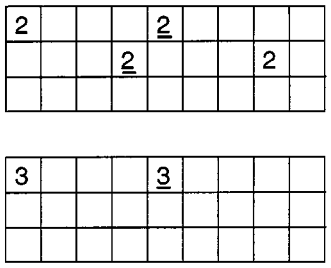

Fig. 6. Recursive use of the algorithm from R. Hartley [6] over the array in Fig. 5.

shifts of the previously calculated coefficient. However, these dependence graph usually results in structures with high logical depth.

Following the research ideas described above, the contribution of the algorithm proposed in this paper is to simplify both logic depth and number of logic operators. It can be accomplished by searching signed subexpressions that are used to create independent structures for each coefficient.

The paper is structured as follows. In Section II, the algorithms used to implement multiplication by constants and their most interesting characteristics are reviewed. Section III describes the new algorithm based on a computer array splitting reduction. Finally, in the fourth section, the results in terms of LO and logical depth (LD) are compared to previous related works [1], [2], [5], [11].

II. Review of Multiplierless Algorithms

The basic method to multiply by constants without using multiplier blocks is based on the power-of-two representation of these constants. A canonic signed-digit (CSD) representation can be employed to implement the basic shift-and-add algorithm, obtaining a reduction in the number of nonzero bits compared with the binary code. As an example, based on the CSD representation, Hartley [1] and [6] proposed a CSE algorithm that was refined by Potkonjak [2]. Both alternatives are based on the location of several common subexpressions between coefficients. The main idea is illustrated in Fig. 5, for a FIR filter with $y(n) = h_0x(n) + h_1x(n-1) + h_2x(n-2)$ (Fig. 4) where $h_0 = (1000\underline{1}000)_2$, $h_1 = (0\underline{1}0\underline{1}01\,010)_2$, and $h_2 = (00\,001\,000\underline{1})_2$.

Four occurrences of the same subexpression between three different coefficients are shown in Fig. 5. The leftmost subexpression can be expressed as $x_2 = x(n) - x(n-1)2^{-1} = x - x[-1] \gg 1$, where the term $[-1]$ represents a unit delay, the sign "$\gg n$" a n-step right shift, and the bar indicates a negative expression. The method proposed by Hartley to identify common subexpressions is then applied recursively. Fig. 6 shows the location of the previous subexpression into a new matrix, and its recursive use. From these figures, a second subexpression is obtained by using $x_3 = x_2 - x_2[-1] \gg 3$. Finally, the complete filter can be expressed as $y = x_3 - x_3 \gg 4$. Fig. 7 shows the final topology of the circuit. In the example, the new structure and the basic one have similar delays. But the method led to a time penalty if it is used to synthesize more complex filters.

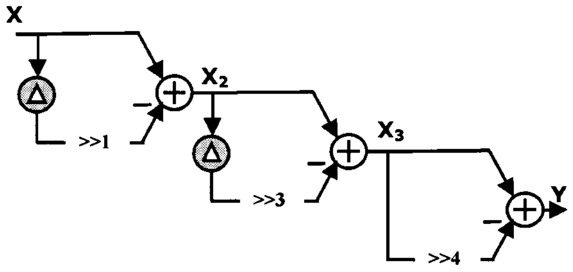

Fig. 7. Filter implemented using the Hartley algorithm and the decomposition done in Figs. 5 and 6.

TABLE I

a	815	110011000$\underline{1}$
b	621	1001101101
c	831	110100000$\underline{1}$
d	105	0001101001

TABLE II

	a	b	c	d
a	-	2	3	1
b		-	2	4
c			-	1
d				-

The ITM algorithm [2] describes another common subexpression-based technique. The method finds the maximum number of coincidences between two signed-digit (SD). For instance, Table I represent the SD coefficients while the binary coincidences between those coefficients is shown in Table II. The final structure by using additions, subtractions and shifts is shown in Fig. 8.

An alternative to the previously described CSE algorithms was developed by Bull [4]. This is one of the first works describing a graph dependence algorithm. The algorithm (named

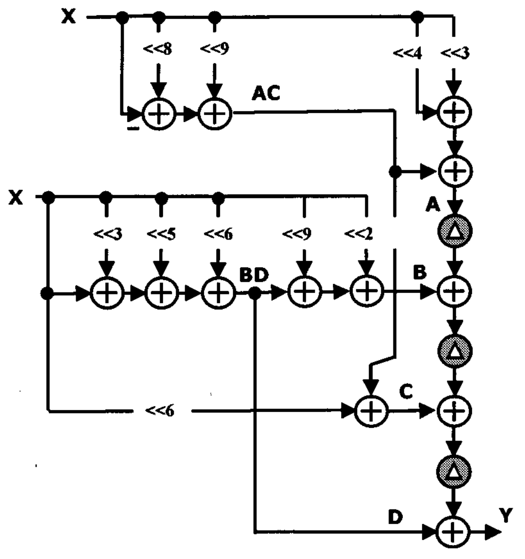

Fig. 8. ITM [2] representation of a four-coefficient filter.

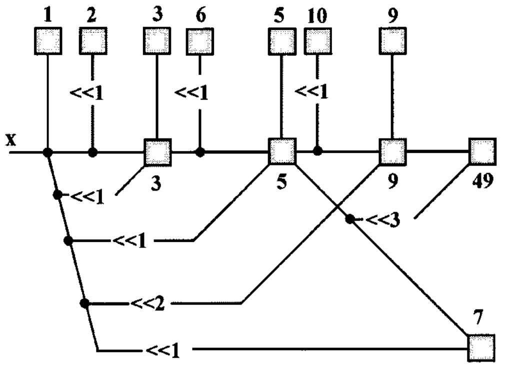

Fig. 9. Filter using the BHA graph synthesis algorithm.

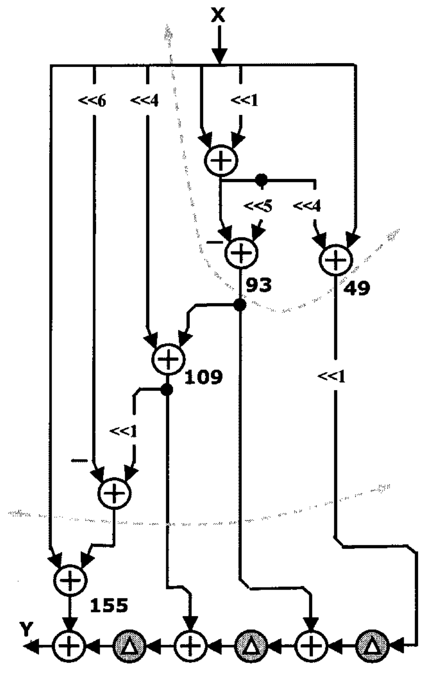

Fig. 10. Implementation of the filter $y[n] = 155x[n]+109x[n-1]+93x[n-2] + 98x[n-3]$ based on the BHM algorithm.

TABLE III

Algorithm	LO	LD
CSD	18	3
ITM	12	5
Hartley	14	3
BHM	11	6

as BHA in subsequent works) represents the filter as a graph that creates a new coefficient, depending on the previous one. In Fig. 9 the BHA algorithm has been used to generate nine unique coefficients (represented by boxes). The authors demonstrated in [7] that the use of adders combined with subtractors leads to a lower number of logic operators. A modified version of the BHA was presented in [5] and [8]. In these works, Dempster and Macleod demonstrated that the Bull-Horrocks modified algorithm (BHM) could reach the minimum number of logical operators. One of the advantages of the BHM is the use of integers with a larger magnitude than the coefficient, in order to reduce the number of operators. For example, the coefficient 7 is obtained as $7 = 8 - 1$ whereas BHA used the expression $7 = 4 + 2 + 1$. BHM allows the designer a 26% average reduction in the number of logical operators, compared with the simplest shift-and-add algorithm from the CSD representation. An example of a filter using BHM is presented in Fig. 10. The dotted lines represent possible frequency cuts to reduce the logic depth.

Although BHM leads to the lowest number of adders, the algorithm does not consider the logic depth. While the graph synthesis algorithms reduce the area, its dependence graph increases the logic depth. A comparative example between these algorithms is shown in Table III. The number of logic operators and logic depth required for a filter with coefficients {105, 621, 815, 831} is summarized. This case study was used in [2] to explain the ITM algorithm.

Main conclusions are that BHM algorithm uses the lowest number of LO, but present the highest LD. Hartley algorithm requires few additional logic operators respect to BHM alternative, having at the same time a minimum LD. In addition, the Hartley algorithm leads to a straightforward layout, thus simplifying hardware synthesis. Others algorithms to implement multiplierless structures like [9] and [10] (compared in [8]) are based on exhaustive search methods, that are time consuming and more difficult to translate into hardware.

III. NR-SCSE Algorithm

In this section, we propose a new array splitting algorithm that combines the advantages of previous methods: it reduces the logic depth obtained from Hartley algorithm, using approximately the same number of logic operators than BHM. The original array starts with a CSD representation of the coefficients, obtaining a layout similar to the Hartley description. The resulting structure can also be easily synthesized into hardware.

The proposed NR-SCSE algorithm makes use of a signed subexpression elimination method as [1] and [2], but subexpressions belonging to different coefficients are not shared. This modification leads to independent structures in the final hardware description, that reduce logic depth and increase the operation frequency. A complete description of the NR-SCSE algorithm is shown below.

The problem starts with the CSD coefficient array $Y_{\mathrm{CSD}}(c,n)$, where c and n are the two-dimensions of the array. c represents the number of coefficients and n is the number of bits, or precision. The algorithm is based on the following three points.

- There are two patterns of signed subexpressions, S1 and S2. S1 represents subexpressions with values $10\cdots01$ or its negative $\underline{1}0\cdots0\underline{1}$, and S2 subexpressions with values $10\ldots0\underline{1}$ or $\underline{1}0\cdots01$. The number of zeros in S2 is between 1 and $n-2$. The nonzero digits in subexpression S1 have the same sign while in S2 1 and -1 are always present.
- Each coefficient C has one associated subexpression model matrix (SMM) P_C with two rows and $n-2$ columns. Each position in the first row represents an occurrence of the subexpression S1, whereas the occurrences of the subexpression S2 are represented in the second row. The column position shows the number of zeros between ones for each subexpression model. For instance, the cell $P_C(2,4) = 3$ represents three occurrences of the subexpression 10000$\underline{1}$ or $\underline{1}$00001 in the CSD representation of the coefficient C.
- The number of common subexpression (S) to be found out in the original CSD array can be selected by the designer. In Section IV, it is demonstrated that selecting the maximum number of common subexpressions can lead to filters with poor properties in terms of area and speed compared to those filters obtained from a reduced number of the most common subexpressions.

The search of S common subexpressions among C coefficients can be implemented in several steps as described in the following algorithm.

Step 1) Obtain the P_C^S SMM arrays for each subexpression s belonging to the interval (1, S) and each coefficient c in (1, C). The algorithm must take into account the overlaps between subexpressions in the same coefficient. As an example, in the pattern 10$\underline{1}$01, the subexpression S2 of value 10$\underline{1}$ or its signed associated subexpression $\underline{1}$01 has two occurrences. However, only one is accepted as valid.

Step 2) Calculate the sum-array $PT\{s\}$ by using (1).

$$PT\{s\} = \sum_{c=1}^{C} P_c^s. \tag{1}$$

Step 3) Select the most common subexpression defined as the maximum value in the $PT\{s\}$ array.

For instance, the NR-SCSE algorithm is applied on the Y_{CSD} array in (2) to generate the sum-array $PT\{s\}$ for the first most common subexpression (MCSE$\{1\}$) in (3).

$$Y_{\mathrm{CSD}} = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & \bar{1} & 0 & \bar{1} \\ 1 & 0 & \bar{1} & 0 & 0 & 0 & 1 & 0 \\ \bar{1} & 0 & 0 & 1 & 0 & 1 & 0 & \bar{1} \\ \bar{1} & 0 & 1 & 0 & 0 & 1 & 0 & \bar{1} \end{pmatrix} \tag{2}$$

$$PT\{1\} = \begin{pmatrix} 3 & 1 & 0 & 0 & 1 & 2 \\ 4 & 2 & 2 & 5 & 0 & 1 \end{pmatrix}. \tag{3}$$

In (3), the $\mathrm{MCSE}\{1\} = \mathrm{PT}\{2,4\}$ is located in the second row—fourth column. Thus, the first subexpression selected that appears five times in the original CSD array is 10 000$\underline{1}$ or its negative value $\underline{1}$00 001. Using the $\mathrm{MCSE}\{1\}$, the SMM obtained for each coefficient are shown in (4)–(7)

$$P_1^1 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \end{pmatrix} \tag{4}$$

$$P_2^1 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{5}$$

$$P_3^1 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \tag{6}$$

$$P_4^1 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \end{pmatrix}. \tag{7}$$

Step 4) Eliminate the previous selected subexpression from the P_C^S arrays, i.e., the nonzero bits in the subexpression are replaced by zeros.

Step 5) Generate the partial residual array $Y'\{s\}$ from the original Y_{CSD} array without the occurrences of the subexpression selected in Step 3).

In (8), we have the resulting partial residual array $Y'\{s\}$ after the selection of $\mathrm{MCSE}\{1\}$

$$Y'\{1\} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & \bar{1} & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & \bar{1} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}. \tag{8}$$

Step 6) If either $s = S$ or the maximum value into the array $PT\{s\} \leq 1$, go to Step 7). Otherwise, increment s and return to Step 1).

Step 7) Obtain as a result the subexpressions, the final residual array, number of adders, and the logic depth of the structure. These values are explained in the next paragraphs.

The iteration of the NR-SCSE algorithm over the last residual array (8) leads to the second subexpression $\mathrm{MCSE}\{2\} = \mathrm{PT}\{2,3\}$ from the new sum-array $\mathrm{PT}\{2\}$

$$\mathrm{PT}\{2\} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 2 & 0 & 0 & 0 \end{pmatrix}. \tag{9}$$

In (9) the subexpression 1000$\underline{1}$ (or its signed value 1000$\underline{1}$) has two occurrences. Finally, the last residual array obtained is

$$Y'\{2\} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}. \tag{10}$$

The filter can be expressed as the union of the last residual array (10) and the SMM arrays obtained for the first and the second MCSE (3)–(6) and (8). Generalizing, the original array Y_{CSD} can be represented as

$$Y_{\text{CSD}} = \Omega_{s=1}^{S}\Omega_{c=1}^{C}P_c\{s\} + Y'. \tag{11}$$

Ω symbolizes the position for each MCSE obtained in the original CSD array. The number of logical operators (adder/subtractors) to implement the final structure can be expressed as

$$\text{LO} = S + \sum_{s=1}^{S}\left(\sum_{c=1}^{C}P_c^s\right) + \sum_{c=1}^{C}Y'\{c\} - 1. \tag{12}$$

Equation (12) represents the addition of the number of subexpressions S, the number of nonzero elements from the SMM array and the number of nonzero elements from the residual array. Equation (13) represents the logic depth obtained for each coefficient

$$\text{LD}(c) = \sum_{s=1}^{S}\left(\sum_{i=1}^{2}\sum_{j=1}^{N-2}P_c^s(i,j)\right) + \sum_{n=1}^{N}Y'(c,n). \tag{13}$$

The final logic depth in the structure is calculated as the maximum value of (13) for all coefficients. Thus, the Y_{CSD} array used as example is implemented with $\text{LO} = 9$ and $\text{LD} = 2$. The LD value is

$$\begin{aligned}\text{LD} &= \max\begin{pmatrix}P_1\{1\}+P_1\{2\}+Y_1'\{2\}\\ P_2\{1\}+P_2\{2\}+Y_2'\{2\}\\ P_3\{1\}+P_3\{2\}+Y_3'\{2\}\\ P_4\{1\}+P_4\{2\}+Y_4'\{2\}\end{pmatrix}\\ &= \max\begin{pmatrix}2+0+0\\0+1+1\\1+1+0\\2+0+0\end{pmatrix} = \max\begin{pmatrix}2\\2\\2\\2\end{pmatrix} = 2.\end{aligned} \tag{14}$$

The final structure obtained from the application of the NR-SCSE algorithm over the filter Y_{CSD} is shown in Fig. 11. The two most common subexpressions are pipelined. Each coefficient is obtained using its SMM array, that represents the connections from the $\text{MCSE}\{2,4\}$, $\text{MCSE}\{2,3\}$, and the residual array Y'.

Fig. 11 shows the final result: 9 logic operators (plus the last vector merging adder), and two adders of logic depth. A pseudo-HDL description code obtained from the NR-SCSE algorithm is

$$\begin{aligned}\{2,4\} &= x + \overline{x} \gg 4\\ \{2,3\} &= x + \overline{x} \gg 3\\ Y' &= x\\ y_0 &= \{2,4\} + \{2,4\} \gg 2\\ y_1 &= \overline{\{2,3\}} \gg 2 + Y'\\ y_2 &= \overline{\{2,4\}} + \{2,3\} \gg 3\\ y_3 &= \overline{\{2,4\}} + \{2,4\} \gg 2.\end{aligned} \tag{15}$$

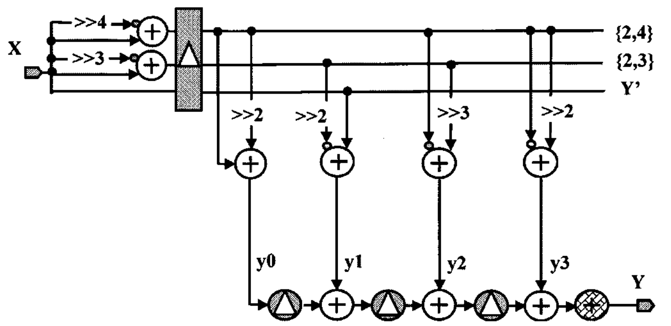

Fig. 11. Transposed FIR filter structure obtained from the NR-SCSE algorithm application.

Equation (16) shows the direct structure of the filter obtained by applying the transposition theorem to (15). A straightforward solution could be obtained from the transposed structure (Fig. 11) using the same theorem. The direct structure requires less registers than the transposed version. However, the resulting logic depth is always better for the transposed representation. The idea of the NR-SCSE algorithm is to create high-speed structures from the lowest logic depth option. Thus, the transposed version is the structure of choice when NR-SCSE is applied. Otherwise, the BHM algorithm offers the minimum number of logic operators to implement the direct structure along with a sometimes poor logic depth.

$$\begin{aligned}\{2,4\} &= y_0 + y_0 \gg 2 + \overline{y_2} + \overline{y_3} + \overline{y_3} \gg 2;\\ \{2,3\} &= \overline{y_1} \gg 2 + y_2 \gg 3;\\ Y' &= y1;\\ y &= \{2,4\} + \overline{\{2,4\}} \gg 4 + \{2,3\} + \overline{\{2,3\}} \gg 3 + Y'.\end{aligned} \tag{16}$$

The number of patterns to be searched in Step 1) reduces dramatically the run time of the above algorithm when compared to optimal run time algorithms as [12], where subexpressions with more than two nonzero bits are examined. These algorithms increase the complexity by the creation of pattern frequency statistics. On the contrary, in this paper we propose to search pattern, S1 and S2 with only two nonzero bits as proposed in [3]. In this way, the process that identifies the most common subexpression in the original CSD array is simplified. In addition, the overlaps between subexpressions can easily be computed. For instance, in the 8-bit coefficient $10\underline{1}00\underline{1}01$, the occurrences of subexpression models S1 and S2 are computed in Table IV. It can be observed that subexpressions a and c have the same occurrences. The final selection is based on the minimum number of zeros between nonzero bits. The pattern a is selected: it leads to a reduction in the number of bits of the adder that implements this subexpression.

Although each selected subexpression modifies the statistics of all subexpressions; we propose to select only the most common subexpression once on each iteration of the algorithm, without taking into account the variation on the statistics. That option generates an algorithm that can be easily included into a synthesis tool with no penalty in the run time operation. To test the proposed method, filters S1 and S2 in [12] were implemented, obtaining the same number of adders.

TABLE IV

	Subexpression	Pattern Model	Occurrences
a	$10\underline{1}$	S2	2
b	$\underline{1}00\underline{1}$	S1	1
c	$10000\underline{1}$	S2	2
d	10000001	S1	1

IV. Algorithm Evaluation and Comparison Results

In this section, two methods are used to evaluate the advantages of the presented NR-SCSE algorithm. First, the main algorithms summarized in Section II are compared with the NR-SCSE approach, using several filter structures. Later results are generalized using a test bench of 100 filters. Both parameters LO and LD are obtained as a function of filter characteristics such as the order (T), the coefficient precision (n) and the required number of common subexpressions (S).

Several parameters must be defined to compare the NR-SCSE algorithm. In Table V, Σ symbolizes the required number of adders. The optimization ratio related to the use of a particular algorithm is usually reflected as $\vartheta = \Sigma/T$; i.e, the number of adders per tap coefficient. The improvement ratio, the number of LO compared with the CSD direct implementation, is defined as $\Lambda = \vartheta_{\text{CSD}}/\vartheta_X$. Finally, the Λ/LD ratio is introduced to measure improvements in both area and logic depth.

The filters implemented in Table V have different order and coefficient precision. FIR1 represents the topology used in Section II to describe the algorithms. FIR2 is a sixteenth-order antisinc filter. FIR3 is the structure used in [2] to describe the ITM algorithm. FIR4 and FIR5 are the filters F1 and F2 introduced in [11] and used in [12] as S1 and S2 to compare different algorithms. Finally, FIR6 is the filter designed in [13].

Table V shows that NR-SCSE obtains always a lower number of logic operators respect to the Hartley algorithm. The reason is that our algorithm searches for the best most common subexpression for each filter specification. To the contrary, Hartley just uses the two statistically most common subexpressions [6]. In addition, the NR-SCSE algorithm is designed to obtain the best logical depth, as shown in Table V. Although BHM algorithm often leads to the least number of logic operators (the proposed algorithm offers the minimum in FIR4 on Table V), the NR-SCSE algorithm has always the best compromise between logic depth and number of logic operators. This result allows the designer to get a matrix-vector product with the best relation between frequency and area.

The advantages of the NR-SCSE algorithm can be illustrated with a set of filters designed using the Park and McClellan algorithm [14]. These filters have different ripples in passband and stopband ranging from 0.001 to 0.0001, and from 0.025 to 0.1, respectively. Moreover, the set includes both narrow and wide band filters.

In Fig. 12 it can be observed that there exists a maximum number of shared subexpressions (S) that yields the minimum number of adders. The number of logical operators does not decrease in most of the filters, even sharing $S \geq 6$ subexpressions. This fact is an important difference between our algorithm and the approach presented in [12]. The effect of sharing too many subexpressions is to increase the final size of the filter. However, the use of few subexpressions implies extra wiring with high fanout that leads to a power consumption increment and a speed reduction in current VLSI technologies. Table VI summarizes the results from Fig. 12. The average of logical operators (LO_m) is 2 with $S = 3$. The same result is obtained using $S = 6$ subexpressions. However, the logic depth average is $\text{LD}_m = 3$ for the entire structures, even using different number of shared subexpressions. This fact is a consequence of the dependence

TABLE V

FILTER	ALGORITHM	Σ	LD	ϑ	Λ	Λ/LD
FIR1	CSD	15	4	3.75	1	0.25
T = 4	BHM	9	7	2.25	1.6	0.23
n = 8	Hartley	10	3	2.5	1.5	0.5
	SCSE (S=2)	9	2	2.25	1.67	0.83
FIR2	CSD	29	5	1.81	1	0.2
T = 16	BHM	19	7	1.18	1.53	0.22
n = 16	Hartley	21	4	1.31	1.38	0.34
	SCSE (S=2)	20	3	1.25	1.45	0.48
FIR3	CSD	18	4	4.5	1	0.25
T = 4	Potkonjak	12	6	3	1.5	0.25
n = 12	BHM	11	7	2.75	1.64	0.23
	Hartley-M	13	3	3.25	1.38	0.46
	SCSE (S=2)	13	3	3.25	1.38	0.46
FIR4	Samueli	23	3	1.92	1	0.33
T = 25	BHM	19	6	1.46	1.31	0.22
n=9	Hartley-M	21	3	1.61	1.19	0.40
	SCSE (S=2)	18	2	1.38	1.39	0.69
FIR5	Samueli	87	4	2.9	1	0.25
T = 59	Hartley	70	4	2.33	1.24	0.31
n = 14	SCSE (S=7)	60	2	2	1.45	0.72
FIR6	CSD	114	6	3.83	1	0.16
T = 60	BHM	61	8	2.03	1.88	0.23
n = 14	Hartley	85	4	2.83	1.35	0.34
	SCSE (S=6)	75	4	2.5	1.53	0.38

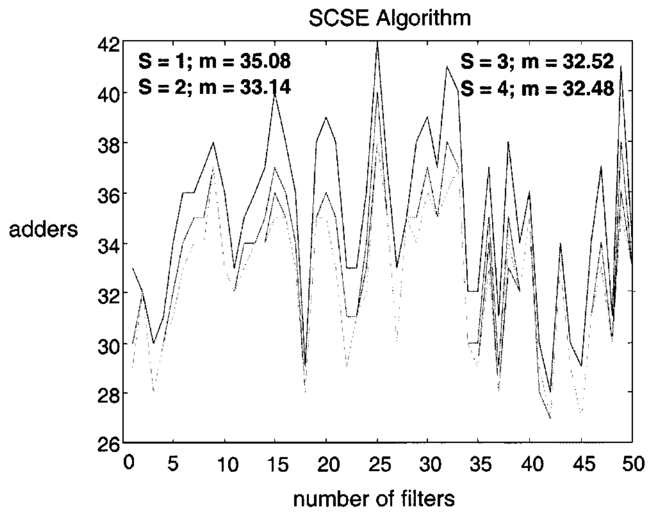

Fig. 12. Number of adders in 50 different filters with $T = 20$, $n = 12$, and S varying from 1 to 4.

TABLE VI

S	LO$_m$	LD$_m$	S	LO$_m$	LD$_m$
1	35.08	3.74	4	32.48	3.00
2	33.14	3.36	5	32.48	2.98
3	32.52	3.06	6	32.48	2.98

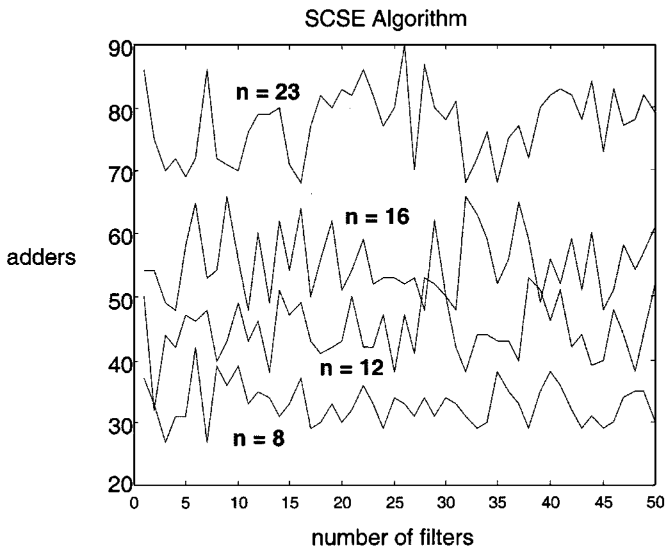

Fig. 13. Number of adders in four groups of 50 filters with different coefficient word length (n) and $T = 30$.

between n and LD, while LD is independent of the T (number of coefficients) and S.

Fig. 13 shows the dependence between the coefficient word length (n) and the number of adders. As the NR-SCSE algorithm searches common subexpressions in each individual coefficient, the word length n directly influences the required number of adders (LO). The figure shows that the average value of LD, for $n = 8$, 12, 16, and 23 is LD$_m$ = 2.1, 2.9, 4, and 5.6, respectively, thus evidencing the dependence between LD and n. Furthermore, LO depends on the order of the filter T. Figs. 14 and 15 present the results in terms of LD and LO for 100 filters with $n = 12$, $S = 4$, and $T = 10$ (Fig. 14) or $T = 40$ (Fig. 15). These figures show that LD does not depend on the order of the filter (the value of this parameter is approximately 3 in both pictures) whereas LD presents important variations (from 17 to 55 as average for $T = 10$ and $T = 50$, respectively).

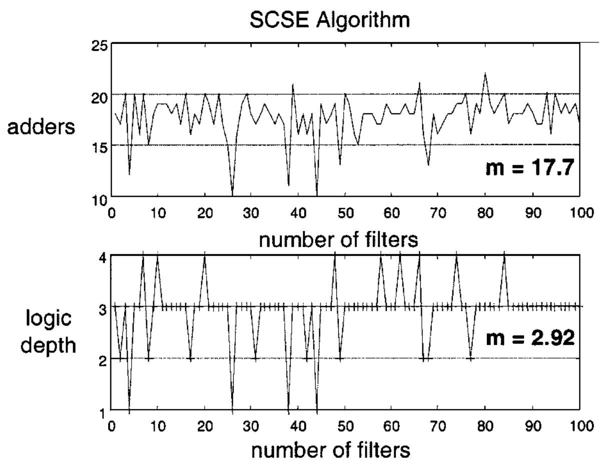

Fig. 14. Parameters LO and LD in 100 filters with $T = 10$, $n = 12$, and $S = 4$.

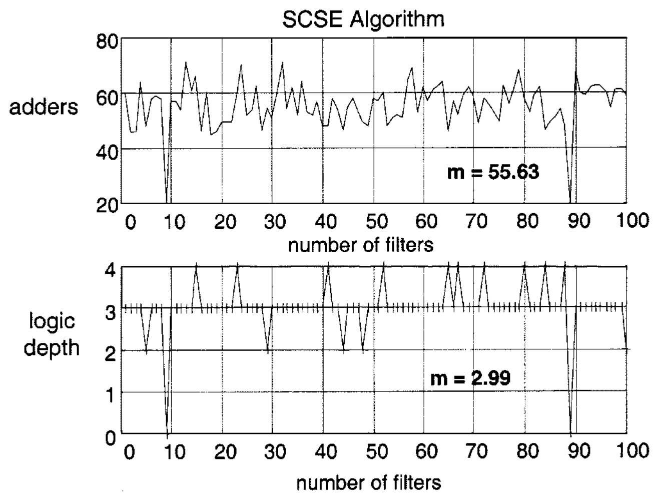

Fig. 15. Number of adders and logic depth in 100 filters with $T = 40$, $n = 12$, and $S = 4$.

From the above results we can conclude the following.

a) The logical depth (LD) depends neither on T nor S. It only depends on the word length n.
b) The number of required logical operators (LO) is function of T and n, but it is not highly dependent on S.

V. Conclusion

The work describes an algorithm obtained from [1] that presents some improvements such as a lower number of adders and logic depth. The new algorithm has been called NR-SCSE because it searches for the nonrecursive signed common subexpressions that must be eliminated from the original CSD array.

Compared with the most popular multiplierless algorithms, NR-SCSE offers the best relation area—frequency operation, or number of adders—logical depth along with optimal runtime operation due to its simplicity. Moreover, the structure of the filter obtained from the algorithm can easily be described using a HDL. Finally several interesting properties in both parameters LD and LO have been studied by using groups of 50 and 100 random filters.

REFERENCES

[1] R. Hartley, "Optimization of canonic signed digit multipliers for filter design," in *Proc. IEEE Int. Symp. Circuits and Systems*, Singapore, June 1991, pp. 1992–1995.

[2] M. Potkonjak *et al.*, "Multiple constant multiplication: Efficient and versatile framework and algorithms for exploring common subexpression elimination," *IEEE Trans. Computer-Aided Design*, vol. 15, no. 2, pp. 151–165, Feb. 1996.

[3] M. Mehendale, S. D. Sherlekar, and G. Vekantesh, "Synthesis of multiplierless FIR filters with minimum number of additions," in *Proc. 1995 IEEE/ACM Int. Conf. Computer-Aided Design*, Los Alamitos, CA, 1995, pp. 668–671.

[4] D. R. Bull, "Primitive operator digital filter synthesis using a shift biased algorithm," in *Proc. IEEE Int. Symp. Circuits and Systems*, vol. 2, Helsinki, Finland, pp. 1529–1532.

[5] A. Dempster and M. D. Macleod, "Constant integer multiplication using minimum adders," *Proc. Inst. Elec. Eng. Circuits and Systems*, vol. 141, no. 5, pp. 407–413, Oct. 1994.

[6] R. Hartley, "Subexpression sharing in filters using canonic signed digit multipliers," *IEEE Trans. Circuits Syst. II*, vol. 43, Oct. 1996.

[7] D. R. Bull and D. H. Horrocks, "Primitive operator digital filter," *Proc. Inst. Elec. Eng. Circuits, Devices and Systems*, vol. 138, pt. G, pp. 401–412, June 1991.

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

[9] D. Li, "Minimum number of adders for implementing a multiplier and its application to the design of multiplierless digital filters," *IEEE Trans. Circuits and Syst. II*, vol. 42, pp. 453–460, July 1995.

[10] A. Chatterjee *et al.*, "Greedy hardware optimization for linear digital circuits using number splitting and refactorization," *IEEE Trans. VLSI Syst.*, vol. 1, pp. 423–431, Dec. 1993.

[11] H. Samueli, "An improved search algorithm for the design of multiplierless FIR filters with powers-of-Two coefficients," *IEEE Trans. Circuits Syst.*, vol. 36, pp. 1044–1047, July 1989.

[12] R. Pasko, P. Schaumont, V. Derudder, S. Vernalde, and D. Durackova, "A new algorithm for elimination of common subexpressions," *IEEE Trans. Computer-Aided Design*, vol. 18, pp. 58–58, Jan. 1999.

[13] M. Martínez-Peiró and L. Wanhammar, "High-speed, low-complexity FIR filter using multiplier block reduction and polyphase decomposition," in *Proc. IEEE Int. Symp. Circuits and Systems*, vol. III, Geneva, Switzerland, May 2000, pp. 367–370.

[14] J. H. McClellan, T. W. Parks, and L. R. Rabiner, "A computer program for designing optimum FIR linear phase digital filters," *IEEE Trans. Audio Electroacoust.*, vol. AU-21, pp. 506–526, Dec. 1973.

Marcos Martínez-Peiró was born in Oliva, Spain. He received the M.Sc. and Ph.D. degrees from the Universidad Politécnica de Valencia, Spain, in 1993 and 2000, respectively.

Since 1993, he has been a Lecturer in the Department of Electronics at the Universidad Politécnica de Valencia. During the first half of 1999, he was a researcher student with the Department of Electrical Engineering, Linköping University, Sweden. Currently, he is Titular Professor at Telecommunications Engineering School of the Universidad Politécnica de Valencia, Spain. His areas of research interest include VLSI signal processing, digital filter design and custom digital signal processing for video applications.

Eduardo I. Boemo received the electrical engineering degree from the Universidad Nacional de Mar del Plata, Argentine, and the Ph.D. degree in telecommunication engineering from the Universidad Politécnica de Madrid, Spain, in 1985 and 1996, respectively.

Currently, he is Titular Professor and Vice Director of research at the School of Computer Engineering, Universidad Autónoma de Madrid, Spain. His current research interests include the design of FPGA-based systems, low-power techniques, computer arithmetics, self-timed circuits, and electrical engineering education.

Lars Wanhammar (S'74–M'81) was born in Vansbro, Sweden, on August 19, 1944. He received the Tekn. Mag., Civ.Ing., Tekn. Dr., and Docent degrees from Linköping University, Sweden, in 1970, 1980, 1981, and 1986, respectively.

Currently, he is a Professor of electronics systems, in the Department of Electrical Engineering, Linköping University. His research interests include theory and design of communication and digital signal processing systems, particularly digital filters and fast transforms, computational properties of digital signal processing algorithms, computer-aided design tools, and very large scale integration circuit techniques. He is the author and coauthor of eight textbooks.