Design of Linear Phase FIR Filters in Subexpression Space Using Mixed Integer Linear Programming

Ya Jun Yu, *Member, IEEE*, and Yong Ching Lim, *Fellow, IEEE*

***Abstract*—In this paper, a novel optimization technique is proposed to optimize filter coefficients of linear phase finite-impulse response (FIR) filter to share common subexpressions within and among coefficients. Existing approaches of common subexpression elimination optimize digital filters in two stages: first, an FIR filter is designed in a discrete space such as finite wordlength space or signed power-of-two (SPT) space to meet a given specification; in the second stage, an optimization algorithm is applied on the discrete coefficients to find and eliminate the common subexpressions. Such a two-stage optimization technique suffers from the problem that the search space in the second stage is limited by the finite wordlength or SPT coefficients obtained in the first stage optimization. The new proposed algorithm overcomes this problem by optimizing the filter coefficients directly in subexpression space for a given specification. Numerical examples of benchmark filters show that the required number of adders obtained using the proposed algorithm is much less than those obtained using two-stage optimization approaches.**

***Index Terms*—Common subexpression sharing, finite-impulse response (FIR) filters, mixed integer linear programming (MILP), subexpression space.**

I. Introduction

FINITE-IMPULSE RESPONSE (FIR) filters have many attractive virtues such as exact linear phase property, guaranteed stability, free of limit cycle oscillations, and low coefficient sensitivity. However, the complexity of an FIR filter in terms of the implementation cost in VLSI is generally higher than that of a corresponding infinite-impulse response (IIR) filter meeting the same magnitude response specifications. Therefore, during the past decades, continuous effort has been made to design low complexity FIR filters. One of the most successful strategies is to optimize the filter coefficients in signed power-of-two (SPT) [1]–[3] space, where each coefficient is represented as a sum of a limited number of SPT terms. Thus, coefficient multiplications can be replaced by shifters and adders, so that the implementation of the filter is essentially multiplierless.

The number of adders to implement the FIR filter can be further reduced by extracting common subexpressions from the coefficients [4], [5]. The adders used to realize the common subexpressions can be shared within a coefficient as well as among coefficients. For example, when two coefficients with binary numbers 100101 and 10101 are implemented independently, two adders are required for each coefficient, as shown in Fig. 1(a). Thus, in total four adders are used for the coefficients. If the common subexpression of 101 is extracted and implemented first, only an additional adder is required for each coefficient, ending up with three adders in total, as shown in Fig. 1(b). Even in such a simple example, an adder is saved.

In Fig. 1, transposed direct form implementation is used to share the common subexpression. Since direct form is the dual of the transposed direct form, the subexpression sharing can always be implemented using the direct form in the same complexity as that using the transposed direct form. In transposed direct form implementation, the adders can easily be classified into two categories: 1) those implementing the coefficients; and 2) those inserted between consecutive delays. Since the number of adders between consecutive delays remains invariant for different coefficient implementation scheme, the number of adders used to implement the coefficients is commonly taken as the coefficient implementation complexity. This is also the coefficient implementation complexity measure adopted in this paper.

Various optimization techniques [5]–[16] have been proposed to identify and extract as many common subexpressions from coefficients as possible to reduce the number of adders. Traditionally, the filters are designed in two stages. First, the FIR filter is designed in a discrete space such as finite wordlength space or SPT space to meet a given specification. In the second stage, an optimization technique is applied on the discrete coefficients to find and eliminate the common subexpressions. Many of the optimization techniques in the second stage are based on a heuristic search or combined exhaustive search.

There is an obvious disadvantage in such a two-stage optimization technique. The filter designed in the first stage is an optimum or suboptimum solution in the given discrete space, but is not in the discrete space constructed by subexpressions. For example, assume that the optimum value of a coefficient is 101, and the discrete value obtained in the first stage optimization is 100 if only 1 SPT term is allowed per coefficient; it is impossible to obtain the optimum solution of 101 in the second stage search, since 101 is not a permissible value in the 1st stage optimization. A recent publication [13] alleviates this disadvantage by perturbing the first stage (sub)optimum solution during the second stage search. Such a perturbation is equivalent to a local search.

Two techniques [17], [18] were proposed to optimize the filter coefficient in subexpression space directly for given filter specifications. In [17], the dynamic range of each coefficient is determined; exhaustive search is then used to check all possible

Manuscript received October 16, 2006; revised March 12, 2007. This work was supported in part by Grant COE-SUG (1) and Temasek Laboratory@NTU. This paper was recommended by Associate Editor T. Hinamoto.

The authors are with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore (e-mail: eleyuyj@pmail.ntu.edu.sg; elelimyc@pmail.ntu.edu.sg).

Digital Object Identifier 10.1109/TCSI.2007.904599

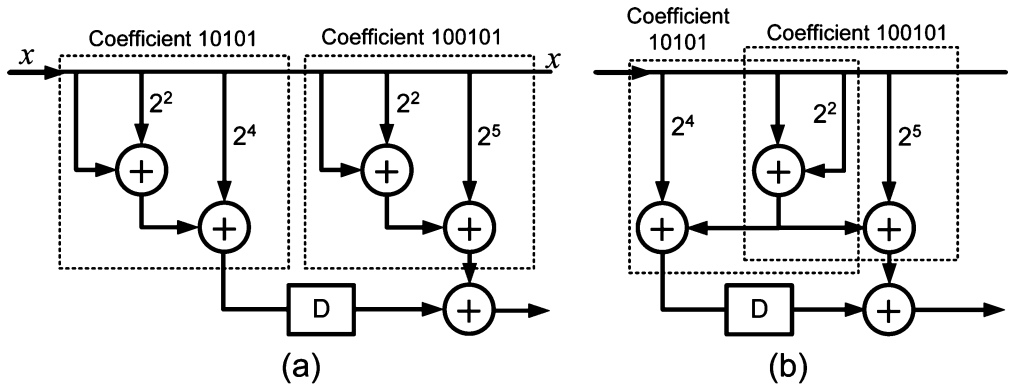

Fig. 1. Implementation of two coefficient multiplications. (a) Each coefficient is independently implemented by shifters and adders. (b) Common subexpression of 101 is extracted and implemented to be shared between two coefficients.

combinations to find the best solution. In [18], a 0/1 integer programming is used to minimize the number of adders subject to a given filter specification. Very good results were shown in both papers. However, the computation time of these two approaches are huge. They are only suitable to design short filters with short wordlength. For example, in [18], it is stated that the reasonable solving time to optimize a filter with 24 taps and 10 coefficient bits is within one day. Since the computation time of these optimization techniques increase at least exponentially with the filter length and bit width, it is impractical to design longer filters with wider bit width.

In this paper, a technique directly optimizing the filter coefficients in the discrete space constructed by a given set of subexpressions is proposed. Mixed integer linear programming (MILP) is the technique employed to optimize the filter coefficients to meet the given specifications. In this technique, the frequency response ripple is minimized subject to a given number of adders. The obtained results may not be the optimum in terms of the number of adders, but the saving in the number of adders achieved using the proposed technique is significant compared with those obtained using other techniques that can be used to design filters with respectable length and bit width. The computation time to optimize a filter with order 60 and coefficient bit width 12 is typically within a few minutes to a few hours depending on specific examples.

The remaining of the paper is organized as follows. Section II introduces the space constructed by a given set of subexpressions. An algorithm to find the best approximation of a real number in the subexpression space is proposed. Section III introduces the branch and bound MILP to optimize the filter coefficients in subexpression spaces. Numerical examples are shown in Section IV to illustrate the advantage of the proposed technique. Section V proposes a multi-step optimization approach to circumvent the difficulty of long filter design using MILP. Section VI shows the flexibility of MILP to design filters in subexpression space with other constraints such as limiting the maximum adder step of coefficients.

II. Construction of Discrete Subexpression Space

Since a finite word-length coefficient values can be converted into an integer value by multiplying a suitable integer power-of-two, it is sufficient to consider only coefficients with integer values.

A. Subexpression Space

It is well known that an integer n can be represented in SPT number space as [3], [19]

$$n = \sum_{i=0}^{K-1} y(i)2^{q(i)}, \qquad y(i) \in \{-1, 1\} \tag{1}$$

where K is the number of SPT terms, and $q(i)$ is a non-negative integer.

Similarly, a discrete subexpressions space can be constructed by defining its element as

$$n = \sum_{i=0}^{K-1} y(i)2^{q(i)}, \qquad y(i) \in S \tag{2}$$

where S is a set of permissible subexpression bases. The set of subexpression bases is also called basis set in the remaining of this paper for expository convenience. In (2), $y(i)2^{q(i)}$ is a shifted version of a subexpression basis, named a subexpression term, and K is redefined as the number of subexpression terms.

Usually, a subexpression basis refers to a trinary string with more than 1 nonzero digit, such as 1011 or $100\bar{1}$, where $\bar{1}$ represents -1. In this paper, a single digit 1, 0, or $\bar{1}$ is also broadly referred to as a subexpression basis. Therefore, a typical basis set may be $S_1 = \{0, 1, \bar{1}, 101, 10\bar{1}, \bar{1}0\bar{1}, \bar{1}01\}$. Thus, we say that a subexpression space is constructed based on a basis set according to (2). We stipulate that if a digit string is in a basis set, its negative value must also be in the same set. This stipulation is reasonable because inclusion of the negative value into the set does not incur the use of additional adders. It is noted further that element 0 in S_1 has no contributions in the construction of subexpression space, but it is included in every subexpression basis set for the convenience of the description of algorithm, of which we will see in the following derivation.

The elements of a basis set can alternatively be represented by decimal numbers. For example, S_1 can be written as $\{0, \pm 1, \pm 3, \pm 5\}$. In this paper, these two representations are used interchangeably for expository convenience.

B. Some Definitions About Basis Set

The basis set is said to be contiguous if the basis set contains contiguous odd numbers starting from ± 1, i.e., any odd

number whose absolute value is smaller than the maximum element is an element of the set. Thus, S_1 is contiguous, but $S_2 = \{0, \pm 1, \pm 5, \pm 7\}$ is not contiguous.

The number of adders required to construct a basis is called the order of the basis. For example, $10\bar{1}$ is a first-order basis, whereas 10101 is a second-order basis.

The order of the basis set is defined as the number of adders required to generate the values of all elements in the set. It is obvious that the order of S_1 is 2 since 1, 0 and $\bar{1}$ do not require any adder to construct, 101 and $10\bar{1}$ are first-order bases requiring one adder each, whereas $\bar{1}0\bar{1}$ and $\bar{1}01$ are the negatives of 101 and $10\bar{1}$, respectively, where no additional adder is required.

It is apparent by inspection that the order of a contiguous basis set is equal to $(P-1)/2$, where P is the maximum elements of the set.

From the above definitions, it is interesting to note that SPT space is a special subexpression spaces whose basis set has order 0.

C. Finding the Best Approximation of a Number in Subexpression Space

Having the subexpression space constructed based on a basis set, we wish to find the best approximation of a real number x using a sum of not more than K subexpression terms $y(i)2^{q(i)}$, where $y(i)$ is an element of the set of permissible subexpression bases and $q(i)$ is a non-negative integer. Let $[x]_K$ be the K-term subexpression number that is a best approximation to x. Since the number of elements in the basis set is limited, an exhaustive search can be employed to find the combination of K pairs of $y(i)$ and $q(i)$ which minimize the error between x and the approximation.

When the order of the basis set and/or K are/is increasing, exhaustive search becomes computational impractical. A greedy algorithm for computing $[x]_K$ is proposed. In this algorithm, initially, the approximation to x, denoted as $[x]$, is set to 0. Subexpression terms are assigned to $[x]$ one at a time. At each time, the subexpression term which can minimize the difference between x and $[x]$ is assigned to $[x]$. The algorithm runs as follows.

Step 1) Initialize $m = 1$ and $z_0 = x$.
Step 2) Find $y(m)2^{q(m)}$ which minimizes $|z_{m-1} - y(m)2^{q(m)}|$.
Step 3) If either $y(m) = 0$ or $m = K$, go to Step 6; otherwise go to Step 4.
Step 4) Update $z_m = z_{m-1} - y(m)2^{q(m)}$.
Step 5) Increment m. Go to Step 2.
Step 6) $[x]_K = \sum_{i=1}^{K} y(i)2^{q(i)}$. Stop.

In step 2, $y(m)$ is selected from the subexpression basis set. $y(m) = 0$ is also a possible choice. When $y(m) = 0$ minimizes $|z_{m-1} - y(m)2^{q(m)}|$, it implies that any further subexpression terms from the given basis set will not reduce the approximation error. For this reason, 0 is included in every basis set, as stated earlier. In the following, 0 is omitted from the basis set for expository convenience.

From our experience derived from a large number of examples when $K = 2$ and $K = 3$, and when the basis set is contiguous, the approximation obtained by using this greedy algorithm is the best approximation. Unfortunately, no mathematic proof is available so far to prove the optimality of the algorithm.

However, it is possible to find counter examples showing that the approximation obtained using the greedy algorithm is not the best approximation when the basis set is not contiguous. An example of which is to round the integer 23 to a subexpression space using two subexpression terms from the noncontiguous basis set of $S_3 = \{\pm 1, \pm 5, \pm 7, \pm 9\}$; the approximation obtained using the greedy algorithm is $5 \times 2^2 + 1 \times 2^2 = 24$, but the best approximation is $1 \times 2^4 + 7 \times 2^0 = 23$.

III. Optimizing Filter Coefficients Using Branch-and-Bound MILP

The zero-phase frequency response of a linear-phase FIR filter of length N can be expressed as

$$H(\omega) = \sum_{n=0}^{\lfloor \frac{N-1}{2} \rfloor} h(n)\,\mathrm{Trig}(\omega, n) \tag{3}$$

where, $h(n)$'s are the filter coefficients and $\mathrm{Trig}(\omega, n)$ is an appropriate trigonometric function depending on the parity of N and the symmetry of the impulse response [20].

For a given set of filter specifications, such as passband edge ω_p, stopband edge ω_s, passband and stopband ripple ratio δ_p/δ_s, and filter length N, the optimum continuous coefficients minimizing the magnitude ripple can be found by formulating the problem as a linear programming problem as follows:

$$\begin{aligned} &\text{minimize}: \quad \delta \\ &\text{subject to}: \quad 1 - \delta \le H(\omega) \le 1 + \delta, \qquad \text{for } \omega \in [0, \omega_p] \\ &\qquad -\frac{\delta_s \delta}{\delta_p} \le H(\omega) \le \frac{\delta_s \delta}{\delta_p}, \qquad \text{for } \omega \in [\omega_s, \pi] \end{aligned} \tag{4}$$

To find the coefficient values in a discrete space for the same set of filter specifications, the linear programming problem is replaced by a MILP problem. Branch and bound algorithm is one of the most efficient techniques to solve MILP problem.

First we review the depth-first search branch and bound algorithm [21]–[23] which has successfully optimized filter coefficients in integer and SPT spaces.

In the branch and bound (B&B) algorithm, after the continuous optimum solution is obtained, a coefficient x_k is selected for branching. Suppose that x_2 is selected and that the integer space is the desired discrete coefficient space. Suppose also that the continuous optimum value of x_2 is 3.4. Two subproblem P_1 and P_2 are created by imposing the bounds $x_2 \le 3$ and $x_2 \ge 4$, respectively (see Fig. 2). Problem P_2 is stored and P_1 is solved. Another coefficient (say x_1) is then selected for partitioning into two subproblems P_3 and P_4 by imposing bounds on the selected coefficient (say $x_1 \le \lceil x_1 \rceil$ and $x_1 \ge \lfloor x_1 \rfloor$, respectively, where $\lceil x \rceil$ is the permitted discrete value immediately larger than or equal to x and $\lfloor x \rfloor$ is the permitted discrete value of x immediately less than or equal to x). P_4 is stored and P_3 is solved. P_3 is then further partitioned into P_5 and P_6. In a similar way, P_6 is stored and P_5 is solved. Suppose that P_5 yields a discrete solution. P_5 is then fathomed and P_6 is solved. In Fig. 2, a line underneath a node indicates that no further exploration from that node can be profitable. Such a node is said to be fathomed. If P_6 yields a discrete solution, P_6 is fathomed and

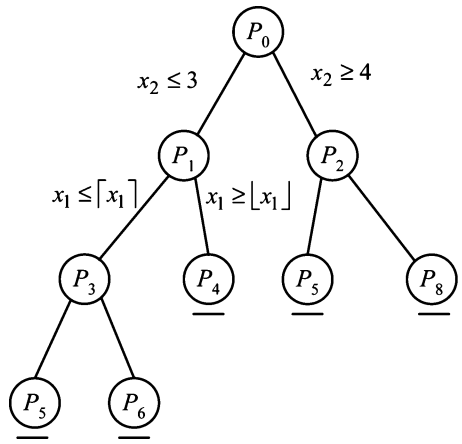

Fig. 2. Example of a branch-and-bound tree.

TABLE I
NUMBER OF SUPPORTED INTEGERS BETWEEN 0 TO 1 023 (INCLUSIVE) WITH NOT MORE THAN 2 SUBEXPRESSION TERMS USING THE FIRST-, SECOND-, THIRD-, AND FOURTH-ORDER SUBEXPRESSION BASIS SETS, RESPECTIVELY. THE BASES AVAILABLE FOR USE ARE $\pm1, \pm3, \pm5, \ldots, \pm17$

	The number of supported integers between 0 and 1,023 (inclusive)		
Basis set	Supported by contiguous basis set	The largest number supported by other basis sets	The other basis set supporting the largest number of integers
1st order	232	221	$\{\pm1, \pm7\}$
2nd order	362	362	$\{\pm1, \pm3, \pm7\}$ $\{\pm1, \pm5, \pm7\}$
3rd order	512	498	$\{\pm1, \pm3, \pm5, \pm15\}$
4th order	608	608	$\{\pm1, \pm3, \pm5, \pm7, \pm11\}$ $\{\pm1, \pm3, \pm5, \pm7, \pm13\}$ $\{\pm1, \pm3, \pm5, \pm7, \pm15\}$

the algorithm backtracks to P_3 and switches to solve P_4. The branching, backtracking and searching process continues until all the nodes are fathomed. The algorithm searches the tree in a depth-first manner and earns its name "depth-first" search.

When the discrete space has a highly nonuniform grid as in the case of SPT space or the subexpression space, it is more appropriate to minimize the normalized peak ripple magnitude [22]. In order to optimize the normalized peak ripple, the passband gain must be left as a continuous variable. Technique for determining the optimal passband gain has been reported in [22].

In applying the depth-first search B&B algorithm to optimize the coefficient in subexpression space, it is necessary to:

1) determine $\lceil x \rceil$ and $\lfloor x \rfloor$ for any given coefficient value x, subject to a given basis set and a given number of subexpression terms used;
2) determine the basis set of the subexpression space; and
3) determine the number of subexpression terms used for each coefficient.

The basis set and the number of subexpression terms used for each coefficient determine the number of adders required for the implementation of the filter.

In the following, all the above are addressed.

A. Determination of $\lceil x \rceil$ and $\lfloor x \rfloor$

The algorithm starts with determining $[\bar{x}]_K$ for any given real number x, where $\bar{x}$ is the integer nearest to x. $[\bar{x}]_K$ may be determined by exhaustive search or greedy algorithm.

Step 1) Find $[\bar{x}]_K$ by exhaustive search or by greedy algorithm.
Step 2) if $[\bar{x}]_K = x$, let $\lceil x \rceil = \lfloor x \rfloor = [\bar{x}]_K$, go to Step 7; otherwise, go to Step 3.
Step 3) If $[\bar{x}]_K > x$, let $\lceil x \rceil = [\bar{x}]_K$ and $i = -1$; otherwise, let $\lfloor x \rfloor = [\bar{x}]_K$ and $i = 1$.
Step 4) Let $x' = x$.
Step 5) Update $x' = x' + i$. If $[\bar{x'}]_K = [\bar{x}]_K$, repeat Step 5; otherwise, go to Step 6.
Step 6) If $i > 0$, let $\lceil x \rceil = [\bar{x'}]_K$; otherwise, let $\lfloor x \rfloor = [\bar{x'}]_K$.
Step 7) Stop.

In the above algorithm, the most time consuming operation is to find $[\bar{x}]_K$. If the subexpression basis set is static, i.e., if it is not growing or updated during the optimization, a look-up table can be created to store $[x]_K$ for all integer x within a given range. Then, using the above algorithm, the upper bound and lower bound of any continuous numbers can be obtained efficiently.

B. Selection of Basis Set

Higher order basis set includes more subexpression bases requiring more adders to generate these bases, but may result in requiring less subexpression terms to approximate a given set of filter coefficients leading to a reduction in the total number of adders for the entire filter. Therefore, in the implementation, the objective is to determine the basis set that would result in the minimum number of adders required to implement the entire filter. Since the subexpressions 1, and $\bar{1}$ do not require any adders, they are automatic members of all basis sets.

Further investigations show that, for a given order of basis set, generally, contiguous set supports a denser discrete coefficient grid space. For example, if basis, $\pm1, \pm3, \pm5, \ldots, \pm17$ are available, $\{\pm1, \pm3\}, \{\pm1, \pm5\}, \{\pm1, \pm7\}, \{\pm1, \pm9\}$, $\{\pm1, \pm15\}$, and $\{\pm1, \pm17\}$ are all first-order basis sets, but only $\{\pm1, \pm3\}$ is a contiguous set. For 10 bits integers, corresponding to integers between 0 and 1 023 (inclusive), the number of integers that are supported by those basis sets with not more than 2 subexpression terms are 232, 220, 221, 207, 206 and 190, respectively. Thus, the basis set $\{\pm1, \pm3\}$ which supports 232 integers between 0 and 1 023 (inclusive) has the densest discrete coefficient grid space.

Table I lists the number of integers that are supported by contiguous first-, second-, third-, and fourth-order basis set, as well as the most number of integers that can be supported using any other first-, second-, third-, and fourth-order basis sets. The integer ranges from 0 to 1 023 and $K = 2$. From Table I, it is seen that for all the order of basis sets, the contiguous sets support the most number of integers.

Base on this observation, the contiguous basis set is a good choice for constructing the subexpression space. Furthermore, the greedy algorithm can be used to find the nearest discrete value to any number where the basis set is the contiguous basis set.

C. Number of Subexpression Terms

The statistics for the number of subexpression terms used by each coefficient is such that larger magnitude coefficients use more subexpression terms than smaller magnitude coefficients. A given frequency response specification can be met by either increasing the number of subexpression terms per coefficient or by enlarging the number of basis set.

Experience shows that, for the minimum number of adders used, the number of subexpression terms per coefficient should not be more than two; the size of the basis set should be enlarged to meet specifications instead of increasing the number of subexpression terms per coefficient beyond two.

When the maximum number of subexpression terms is limited to not more than 2, generally, the required order of the basis set depends on the coefficient effective wordlength (EWL). If the effective wordlength is not larger than 9, usually, second-order basis set is a good choice to minimize the total number of adders. For effective wordlength larger than 9, each additional bit requires about 3 more subexpression bases. Thus, the suggested order of basis set is

$$\begin{cases} 2, & \text{for EWL} \ \le 9 \\ 2 + (\text{EWL} - 9) \times 3, & \text{for EWL} \ > 9. \end{cases}$$

IV. Numerical Examples

In this section, several FIR filters are designed using the proposed optimization techniques. The qualities of the results, in terms of the number of adders used to generate the coefficients, are compared with those obtained using two-stage approaches.

A. Example 1

This example is taken from example 2 of [2] (and labeled as $L2$ in Table IV). The specification is a lowpass filter with filter length $N = 63$, passpand and stopband edges $\omega_p = 0.2\pi, \omega_s = 0.28\pi$, passband and stopband ripples $\delta_p = 0.028$ and $\delta_s = 0.001$, and the effective wordlength is 11 bits excluding sign bit.

The subexpression space for the design is constructed based on a seventh-order contiguous basis set $\{\pm1, \pm3, \pm5, \pm7, \pm9, \pm11, \pm13, \pm15\}$. In the 32 distinct coefficients, the 13 coefficients with the largest magnitude in the continuous design are allocated with a maximum of 2 subexpression terms each, whereas the remaining are assigned with a maximum of 1 term each. A total of not more than 45 subexpression terms will be used. The depth-first branch and bound MILP produces a set of coefficients meeting the given specifications using only 42 subexpression terms with 10 coefficients requiring 2 subexpression terms. Therefore, the total number of adders used to produce the coefficients is 17, which can be broken down into 7 adders for constructing the basis set and 10 adders for combining the bases into coefficient values. This design yields a 25% saving in the number of adders when compared with the previously obtained best results [7]–[11]. The filter coefficients expressed in the sum of subexpression terms are listed in Table II.

B. Example 2

Three other filters, labeled as $S1, S2$, and $L3$ are also designed. $S1$ and $S2$ are the filters originally given in Example 1 and 2 of [24], whereas $L3$ is the filter originally given in Example 3 of [2]. The filter specifications for $S1$ and $S2$ were not explicitly given in [24]. From the coefficient values given in [24], the frequency responses of the filters were evaluated; the specifications were recovered and tabulated in Table III for easy reference. The total number of adders used to produce the subexpression coefficients as well as the subexpression basis set for each design are listed in Table IV. The best results obtained in the previous literatures are tabulated for comparison. The filter

TABLE II
Impulse Response of a Subexpression Coefficient Filter Meeting the Specification of Example 1, $h(n) = h(62 - n)$ for $32 \le n \le 62$

Passband gain 5466.7263							
Impulse Response $\times 4096$:							
$h(31) =$	5×2^8	$-$	15×2^0	$h(15) =$	$-$	7×2^3
$h(30) =$	9×2^7	$+$	1×2^0	$h(14) =$	$-$	5×2^1
$h(29) =$	13×2^6	$+$	11×2^1	$h(13) =$		1×2^5
$h(28) =$	7×2^6	$+$	3×2^2	$h(12) =$		13×2^2
$h(27) =$	11×2^3			$h(11) =$		11×2^2
$h(26) =$	-11×2^4	$+$	1×2^3	$h(10) =$		9×2^1
$h(25) =$	-1×2^8	$-$	7×2^0	$h(9) =$	$-$	3×2^2
$h(24) =$	-13×2^4	$-$	7×2^0	$h(8) =$	$-$	1×2^5
$h(23) =$	-11×2^3	$+$	1×2^1	$h(7) =$	$-$	9×2^2
$h(22) =$	3×2^4			$h(6) =$	$-$	13×2^1
$h(21) =$	1×2^7			$h(5) =$	$-$	5×2^1
$h(20) =$	1×2^7	$+$	1×2^1	$h(4) =$		1×2^2
$h(19) =$	9×2^3	$+$	1×2^1	$h(3) =$		3×2^2
$h(18) =$	-1×2^2			$h(2) =$		13×2^0
$h(17) =$	-1×2^6			$h(1) =$		9×2^0
$h(16) =$	-5×2^4			$h(0) =$		1×2^2

TABLE III
Filter Specifications for $S1$ and $S2$ [24]

Filters	N	ω_p	ω_s	δ_p	δ_s	EWL*
$S1$	25	0.3π	0.5π	0.0157	0.0066	9
$S2$	60	0.042π	0.14π	0.012	0.001	13

* EWL: Effective WordLength (excluding sign bit)

TABLE IV
Total Number of Adders used to Produce the Filter Coefficients, Denoted as M

Filters	N	The proposed technique			Best Published	
		M	Basis set used	EWL	M	EWL
$S1$	25	**4**	$\{\pm1, \pm3, \pm5\}$	9	**6** [8]–[11]	9
$S2$	60	**19**	$\{\pm1, \pm3, \pm5, \pm7, \pm9, \pm11, \pm13, \pm15\}$	11	**26** [8], [10]	13
$L1$	121	**44**	$\{\pm1, \pm3, \pm5, \pm7, \pm9, \pm11, \pm13, \pm15, \pm29, \pm31, \pm53, \pm57, \pm59, \pm89, \pm113, \pm125, \pm129, \pm145, \pm183, \pm271\}$	14	**52** [8], [10]	14
$L2$	63	**17**	$\{\pm1, \pm3, \pm5, \pm7, \pm9, \pm11, \pm13 \pm 15\}$	11	**22** [8], [10]	11
$L3$	36	**3**	$\{\pm1, \pm3, \pm5\}$	9	**5** [7]–[11]	9

TABLE V
IMPULSE RESPONSE OF SUBEXPRESSION COEFFICIENT FILTERS OF $S1, S2$ AND $L3$

Filter $S1, h(n) = h(24-n)$ for $13 \le n \le 24$

Passband gain 485.2682

Impulse Response $\times 512$:

$h(12) = 3\times 2^6 - 1\times 2^0$	$h(7) = -3\times 2^0$	$h(2) = -1\times 2^1$
$h(11) = 5\times 2^5 - 1\times 2^4$	$h(6) = 1\times 2^4$	$h(1) = 3\times 2^0$
$h(10) = 3\times 2^4$	$h(5) = 5\times 2^1$	$h(0) = 1\times 2^1$
$h(9) = -3\times 2^3$	$h(4) = -1\times 2^2$	
$h(8) = -1\times 2^5$	$h(3) = -1\times 2^3$	

Filter $S2, h(n) = h(59-n)$ for $30 \le n \le 59$

Passband gain 10945.3361

Impulse Response $\times 8192$:

$h(29) = 7\times 2^7 + 5\times 2^2$	$h(14) = -3\times 2^5 - 3\times 2^0$
$h(28) = 7\times 2^7 - 1\times 2^2$	$h(13) = -3\times 2^5 - 3\times 2^0$
$h(27) = 13\times 2^6 + 5\times 2^1$	$h(12) = -11\times 2^3 - 3\times 2^0$
$h(26) = 3\times 2^8 + 1\times 2^2$	$h(11) = -5\times 2^4$
$h(25) = 11\times 2^6 - 9\times 2^1$	$h(10) = -1\times 2^6$
$h(24) = 9\times 2^6 + 11\times 2^0$	$h(9) = -3\times 2^4$
$h(23) = 15\times 2^5 + 1\times 2^1$	$h(8) = -1\times 2^5$
$h(22) = 3\times 2^7 - 1\times 2^3$	$h(7) = -5\times 2^2$
$h(21) = 9\times 2^5 - 15\times 2^0$	$h(6) = -5\times 2^1$
$h(20) = 11\times 2^4 + 1\times 2^1$	$h(5) = -1\times 2^1$
$h(19) = 3\times 2^5$	$h(4) = 3\times 2^0$
$h(18) = 7\times 2^2$	$h(3) = 5\times 2^0$
$h(17) = -13\times 2^1$	$h(2) = 3\times 2^1$
$h(16) = -1\times 2^6$	$h(1) = 5\times 2^0$
$h(15) = -11\times 2^3$	$h(0) = 5\times 2^0$

Filter $L3, h(n) = h(35-n)$ for $18 \le n \le 35$

Passband gain 816.8370

Impulse Response $\times 1024$:

$h(17) = 3\times 2^6 + 1\times 2^2$	$h(11) = -1\times 2^5$	$h(5) = -1\times 2^0$
$h(16) = 5\times 2^5$	$h(10) = -3\times 2^2$	$h(4) = -5\times 2^1$
$h(15) = 3\times 2^5$	$h(9) = 3\times 2^1$	$h(3) = -3\times 2^2$
$h(14) = 1\times 2^5$	$h(8) = 5\times 2^2$	$h(2) = -3\times 2^1$
$h(13) = -1\times 2^4$	$h(7) = 5\times 2^2$	$h(1) = 1\times 2^0$
$h(12) = -5\times 2^3$	$h(6) = 3\times 2^1$	$h(0) = 1\times 2^3$

coefficients of $S1, S2$ and $L3$ are listed in Table V in the form of sum of subexpression terms.

It should be noted that in filter $S2$, 7 adders are required to construct the basis set (see Table IV), and 13 adders are used to combine the bases into coefficient values (see Table V). Thus, the total number of adders required to generate the coefficient should be 20. However, it is observed from the coefficient values of $S2$ in Table V that coefficients $h(13)$ and $h(14)$ have the same value. Therefore, an adder to sum the subexpression term can be saved, resulting in a total of 19 adders required, as shown in the entries of $S2$ in Table IV.

The results in Table IV show that the proposed algorithm not only use less adders to meet the same specifications, but also may produce filter coefficients with smaller dynamic ranges, as in the case of filter $S2$.

V. LONG FILTER DESIGN IN SUBEXPRESSION SPACE

For long filters, MILP may requires unacceptably long computer time [2]. An approach to overcome this difficulty is to optimize the coefficients in multi-steps. The coefficients are split into two or more groups, say Group A and Group B. First, the coefficients in Group A are optimized in discrete subexpression space, while the others are optimized in continuous space. Next, the coefficients in Group A are fixed at the optimized discrete values and the coefficients in Group B are then optimized in subexpression space.

Such multi-step optimization tremendously accelerates the MILP search; the price to be paid for is the degradation of the resulting filter's performance. Nevertheless, the multi-step optimization introduces an additional benefit, which may compensate for its disadvantage.

It is noted from the examples in Section V that the optimized discrete coefficient which is allocated with more than 1 subexpression terms may be a value which is not in the basis set. However, such new value cannot serve as a basis to produce other coefficients.

In multi-step optimization, the discrete values produced in earlier steps can serve as bases in later optimization steps. Thus, the supported discrete space is enlarged. For a given basis set, the supported discrete space is sparser in larger magnitude region than in smaller magnitude region. Therefore, intuitively, the coefficients with smaller magnitude in a filter should be optimized in the discrete space earlier than those with larger magnitude. Thus, the new values produced together with the original bases comprise a higher order basis set; this higher order basis set supports denser discrete space in larger magnitude region. Therefore, the optimization of coefficients with larger magnitude will have high chances to obtain better results.

An example taken from example 1 of [2] is used to illustrate the efficiency of this multi-step approach in optimizing long filters in subexpression space.

The filter specification is a 121-tap highpass filter with stopband and passband edges at 0.74π and 0.8π; the stopband and passband ripples are not larger than 0.0001 and 0.0057, respectively; the effective wordlength of filter coefficients is 14 bits excluding the sign bit.

From our experience, using a 3 GHz CPU desktop PC, the computer time required by MILP becomes unacceptable for filter length longer than 70. In this example, the filter coefficients are split into three groups: group A includes coefficients from $h(0)$ to $h(29)$; group B includes coefficients from $h(30)$ to $h(34)$, and group C includes coefficient from $h(34)$ to $h(60)$. The initial basis set is a seventh-order contiguous set, i.e.,

$$\mathrm{SS}_1 = \{\pm 1, \pm 3, \pm 5, \pm 7, \pm 9, \pm 11, \pm 13, \pm 15\}.$$

The reason to have group B with only 5 coefficients is that the magnitude of most of the coefficients in group A are small. Many of those only need one subexpression term to represent when they are optimized in the subexpression space based on the initial basis set. There are only a small number of new discrete values being generated to form new bases. Group B is introduced for the purpose of producing more bases.

The optimization procedure runs as follows. First, the coefficients of group A are optimized in discrete subexpression space constructed based on basis set SS_1, while the coefficients of group B and C are optimized in continuous space. The 11 co-

TABLE VI
IMPULSE RESPONSE OF SUBEXPRESSION COEFFICIENT FILTERS OF $L1, h(n) = h(120-n)$ FOR $61 \le n \le 120$

Passband gain 59327.7019												
Impulse Response $\times 65536$												
$h(60) =$	13×2^{10}	$-$	1×2^{1}	$h(44) =-$	15×2^{6}	$-$	1×2^{2}	$h(28) =-$	1×2^{7}	$h(12) =$	1×2^{6}	
$h(59) =-$	3×2^{12}	$+$	29×2^{1}	$h(43) =$	129×2^{2}	$+$	5×2^{1}	$h(27) =$	15×2^{4}	$h(11) =-$	13×2^{1}	
$h(58) =$	89×2^{0}	$+$	9×2^{10}	$h(42) =$	9×2^{3}	$+$	1×2^{2}	$h(26) =-$	57×2^{2}	$h(10) =-$	7×2^{1}	
$h(57) =-$	183×2^{2}	$-$	145×2^{5}	$h(41) =-$	271×2^{1}	$-$	11×2^{0}	$h(25) =$	29×2^{2}	$h(9) =$	5×2^{3}	
$h(56) =$	183×2^{3}	$+$	9×2^{2}	$h(40) =$	89×2^{3}	$+$	1×2^{0}	$h(24) =$	1×2^{5}	$h(8) =-$	11×2^{2}	
$h(55) =$	89×2^{0}	$+$	5×2^{8}	$h(39) =-$	271×2^{1}	$+$	5×2^{1}	$h(23) =-$	145×2^{0}	$h(7) =$	15×2^{1}	
$h(54) =-$	11×2^{8}	$+$	13×2^{3}	$h(38) =$	9×2^{4}	$-$	5×2^{0}	$h(22) =$	89×2^{1}	$h(6) =-$	7×2^{0}	
$h(53) =$	5×2^{9}	$-$	5×2^{1}	$h(37) =$	129×2^{1}	$+$	1×2^{2}	$h(21) =-$	129×2^{0}	$h(5) =-$	7×2^{1}	
$h(52) =-$	89×2^{4}	$+$	53×2^{0}	$h(36) =-$	31×2^{4}	$+$	3×2^{0}	$h(20) =$	1×2^{5}	$h(4) =$	13×2^{1}	
$h(51) =-$	53×2^{1}			$h(35) =$	59×2^{3}	$+$	3×2^{0}	$h(19) =$	1×2^{6}	$h(3) =-$	7×2^{2}	
$h(50) =$	271×2^{0}	$+$	59×2^{4}	$h(34) =-$	125×2^{1}			$h(18) =-$	59×2^{1}	$h(2) =$	11×2^{1}	
$h(49) =$	59×2^{1}	$-$	53×2^{5}	$h(33) =-$	53×2^{0}			$h(17) =$	113×2^{0}	$h(1) =-$	13×2^{0}	
$h(48) =$	145×2^{3}	$+$	15×2^{1}	$h(32) =$	145×2^{1}			$h(16) =-$	31×2^{1}	$h(0) =$	3×2^{1}	
$h(47) =-$	89×2^{2}	$+$	1×2^{0}	$h(31) =-$	183×2^{1}			$h(15) =-$	3×2^{1}			
$h(46) =-$	31×2^{4}	$+$	5×2^{0}	$h(30) =$	271×2^{0}			$h(14) =$	15×2^{2}			
$h(45) =$	31×2^{5}	$-$	5×2^{1}	$h(29) =-$	9×2^{3}			$h(13) =-$	5×2^{4}			

efficients with the largest magnitude in the continuous design in group A are allocated with a maximum of two subexpression terms each, whereas the others are allocated with a maximum of one term each. MILP produces a set of coefficients using only 38 subexpression terms with 8 coefficients requiring 2 subexpression terms. Therefore, in addition to the adders used for constructing the basis set, the number of adders used to produce the coefficients of group A is 8. Meanwhile, the 8 coefficient values using 2 subexpression terms are repeatedly divided by 2 until becoming odd numbers. The 8 odd numbers are 29, 31, 57, 59, 89, 113, 129, 145, and they will serve as new basis values in the next step of the optimization. Thus, the basis set for the second step optimization is a noncontiguous set

$$\mathrm{SS}_2 = \{\pm1, \pm3, \pm5, \pm7, \pm9, \pm11, \pm13, \pm15, \pm29, \pm31, \pm57, \pm59, \pm89, \pm113, \pm129, \pm145\}$$

with order 15.

In the second step, the coefficients in group B are optimized in discrete subexpression space constructed based on basis set SS_2, while the coefficients of group A are fixed to the optimized discrete value and the coefficients of group C are optimized in continuous space. Each coefficient in group B is allocated with a maximum of 2 subexpression terms. The resulted coefficients in group B uses 9 subexpression terms, which corresponds to 4 adders. Four new values are produced which can be used in the next step of the optimization. They are 53, 125, 183, and 271 obtained by dividing the values of the two-term coefficients by 2 repeatedly until the quotients are odd number. Thus, the basis set for the last step of the optimization becomes a 19th-order set

$$\mathrm{SS}_3 = \{\pm1, \pm3, \pm5, \pm7, \pm9, \pm11, \pm13, \pm15, \pm29, \pm31, \pm53, \pm57, \pm59, \pm89, \pm113, \pm125, \pm129, \pm145, \pm183, \pm271\}.$$

In the last step of the optimization, the coefficient in group C are optimized in discrete subexpression space constructed based on basis set SS_3, while the coefficients of group A and B are fixed to the optimized discrete values obtained in the earlier optimization steps. Each coefficient in group C is allocated with a maximum of 2 subexpression terms. The resulted coefficients in group C uses 51 subexpression terms, which corresponds to 25 adders.

After this 3-step optimization, all coefficients are discrete values in the subexpression space constructed based on basis set SS_3. The total number of adders used is 44, which can be broken down into 7 adders for constructing the initial basis set SS_1, 8, 4, and 25 adders for combining the bases into coefficient values for coefficients in group A, B, and C, respectively. The resulting discrete coefficient filter has a stopband ripple of 0.0009704 and a passband ripple of 0.0056, which meet the given specifications. The total number of adders required to realize the coefficients is 18% less than that of the best results published; the number of adders required for the proposed algorithm as well as that for the best published are listed in $L1$ of Table IV for comparison. The filter coefficients obtained using the proposed algorithm are listed in Table VI.

VI. COEFFICIENT CRITICAL PATH LENGTH

In transposed direct form implementation, the "adder step" of a coefficient is the number of adders along the critical path in the realization of the coefficient. The maximum adder step of all coefficients determines the throughput rate of the filter. Therefore, it is desirable to limit the maximum adder step. Using our proposed technique, it is possible to limit the maximum adder step by choosing suitable basis set and limiting the maximum subexpression terms allowed to each coefficient.

For example, the maximum adder step of the coefficients of filter $S2$ given in Table V is 3, occurring in coefficients

TABLE VII
IMPULSE RESPONSE OF SUBEXPRESSION COEFFICIENT FILTERS OF $S2$, WHERE THE MAXIMUM ADDER STEP IS LIMITED TO 2. $h(n) = h(59-n)$ FOR $30 \le n \le 59$

Passband gain 10738.8435								
Impulse Response $\times 8192$:								
$h(29) =$	7×2^7	$+$	1×2^0	$h(14) =$	$-$	3×2^5		
$h(28) =$	7×2^7	$-$	3×2^3	$h(13) =$	$-$	3×2^5	$-$	1×2^1
$h(27) =$	3×2^8	$+$	7×2^3	$h(12) =$	$-$	3×2^5	$+$	5×2^0
$h(26) =$	3×2^8	$-$	3×2^2	$h(11) =$	$-$	5×2^4	$+$	1×2^1
$h(25) =$	5×2^7	$+$	1×2^5	$h(10) =$	$-$	1×2^6		
$h(24) =$	9×2^6	$-$	1×2^0	$h(\ 9) =$	$-$	3×2^4		
$h(23) =$	15×2^5	$-$	7×2^0	$h(\ 8) =$	$-$	17×2^1		
$h(22) =$	3×2^7	$-$	15×2^0	$h(\ 7) =$	$-$	5×2^2		
$h(21) =$	17×2^4	$-$	3×2^0	$h(\ 6) =$	$-$	5×2^1		
$h(20) =$	5×2^5	$+$	1×2^4	$h(\ 5) =$	$-$	1×2^2		
$h(19) =$	3×2^5			$h(\ 4) =$		1×2^1		
$h(18) =$	7×2^2			$h(\ 3) =$		1×2^2		
$h(17) =$	$-\ 3\times2^3$			$h(\ 2) =$		5×2^0		
$h(16) =$	$-\ 31\times2^1$			$h(\ 1) =$		5×2^0		
$h(15) =$	$-\ 5\times2^4$	$-$	3×2^1	$h(\ 0) =$		5×2^0		

$h(27), h(25), h(24), h(20)$ and $h(12)$. To constrain the maximum adder step to be 2, the basis set is comprised with only first-order basis, such as 101, $10\bar{1}, 10001, 1000\bar{1}$, besides 0, 1 and $\bar{1}$. Meanwhile, each coefficient is allocated with a maximum of not more than two subexpression terms. Thus, the maximum adder step of the resulting filter will not be larger than 2.

A design optimized from a seventh-order basis set $\{0, \pm1, \pm3, \pm5, \pm7, \pm9, \pm15, \pm17, \pm31\}$ uses 21 adders in total meeting the given specification. Since each basis has order less than 2, the coefficients constructed from the summation of not more than two subexpression terms have a maximum adder step of not larger than 2. The coefficients are listed in Table VII. Therefore, by constraining the basis set and the maximum number of subexpression terms for each coefficient, the maximum adder step of the coefficients can be controlled. The price paid for the shorter critical path in this example is that 2 more adders are required in the implementation, when compared with the design shown in Table V. When compared with the previous publications, where the total number of adder is 26 for a maximum adder step of 3 [10], our proposed approach generates filter with fewer adders, as well as shorter critical path.

VII. CONCLUSION

Common subexpression sharing in filter coefficients tremendously reduces the number of adders in FIR filter implementations. In this paper, subexpression spaces constructed based on subexpression basis set is presented. A greedy algorithm is proposed to efficiently find the best approximation of a number in a given subexpression space. MILP is used to optimize filter coefficients in subexpression spaces. Examples show that the number of adders used to synthesize the filter coefficients are significantly reduced when comparing with those obtained using the traditional two-stage optimization approach. Conventionally, MILP is notorious for its high computation load for the design of high-order filters. In this paper, a multi-step optimization approach is tailored to circumvent the difficulty of optimizing long filter in subexpression space. The algorithm proposed in this paper also showed its flexibility for designing filters with other constraints such as the maximum adder steps of coefficients when filters are implemented in transposed direct form structure.

REFERENCES

[1] Y. C. Lim and S. R. Parker, "FIR filter design over a discrete power-of-two coefficient space," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-31, no. 6, pp. 583–591, Jun. 1983.

[2] Y. C. Lim and S. R. Parker, "Discrete coefficient FIR digital filter design based upon an LMS criteria," *IEEE Trans. Circuits Syst.*, vol. CAS-30, no. 10, pp. 723–739, Oct. 1983.

[3] Y. C. Lim *et al.*, "Signed power-of-two term allocation scheme for the design of digital filters," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 46, no. 5, pp. 577–584, May 1999.

[4] D. R. Bull and D. H. Horrocks, "Primitive operator digital filters," *IEE Proc. G*, vol. 138, pp. 401–412, Jun. 1991.

[5] R. I. Hartley, "Subexpression sharing in filters using canonic signed digit multipliers," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 43, no. 10, pp. 677–688, Oct. 1996.

[6] M. D. Macleod and A. G. Dempster, "Mulitplierless FIR filter design algorithms," *IEEE Signal Process. Lett.*, vol. 12, no. 3, pp. 186–189, Mar. 2005.

[7] O. Gustafsson and L. Wanhammar, "ILP modeling of the common subexpression sharing problem," in *Proc. IEEE ICECS'02*, Dubrovnic, Croatia, 2002, pp. 1171–1174.

[8] A. G. Dempster and M. D. Macleod, "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.

[9] R. Paško *et al.*, "A new algorithm for elimination of common subexpressions," *IEEE Trans. Comput.-Aided Design*, vol. 18, no. 1, pp. 58–68, Jan. 1999.

[10] C.-Y. Yao *et al.*, "A novel commom-subexpression-elimination method for synthesizing fixed-point FIR filters," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 51, pp. 2215–2221, Nov. 2004.

[11] Y. Wang and K. Roy, "CSDC: A new complexity reduction technique for multiplierless implementation of digital FIR filters," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 52, no. 9, pp. 1845–1853, Sep. 2005.

[12] 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, Exp. Briefs*, vol. 52, pp. 695–700, Oct. 2005.

[13] F. Xu, C.-H. Chang, and C.-C. Jong, "Design of low-complexity FIR filters based on signed-powers-of-two coefficients with reusable common subexpressions," *IEEE Trans. Comput.-Aided Design*, to be published.

[14] I.-C. Park and H.-J. Kang, "Digital filter synthesis based on an algorithm to generate all minimal signed digit representation," *IEEE Trans. Comput.-Aided Design*, vol. 21, no. 12, pp. 1525–1529, Dec. 2002.

[15] M. Martinez-Peiró, E. Boemo, and L. Wanhammar, "Design of highspeed multiplierless filters using a nonrecursive signed common subexpression algorithm," *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.*, vol. 49, no. 3, pp. 196–203, Mar. 2002.

[16] Y. Voronenko and M. Püschel, "Multiplierless multiple constant multiplication," *ACM Trans. Algorith.*, vol. 3, pp. 1–38, May 2007.

[17] J. Yli-Kaakinen and T. Saramäki, "A systematic algorithm for the design of multiplierless FIR filters," in *Proc. IEEE ISCAS'01*, Sydney, Australia, 2001, pp. 185–188.

[18] O. Gustafsson and L. Wanhammar, "Design of linear-phase FIR filters combining subexpression sharing with MILP," in *Proc. MWSCAS'02*, 2002, pp. 9–12.

[19] Y. J. Yu and Y. C. Lim, "Roundoff noise analysis of signals represented using signed power-of-two terms," *IEEE Trans. Signal Process.*, vol. 55, no. 5, pp. 1409–1416, May 2007.

[20] T. Saramäki, , S. K. Mitra and J. F. Kaiser, Eds., "Finite impluse response filter design," in *Handbook for Digital Signal Processing*. New York: Wiley, 1993, ch. ch. 4.

[21] R. S. Garfinkel and G. L. Nemhauser, *Integer Programming*. New York: Wiley, 1972.

[22] Y. C. Lim, "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.

[23] Y. C. Lim, Y. Sun, and Y. J. Yu, "Design of discrete-coefficient fir filters on loosely connected parallel machines," *IEEE Trans. Signal Process.*, vol. 50, no. 6, pp. 1409–1416, Jun. 2002.
[24] 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, Jul. 1989.

Ya Jun Yu (S'99–M'05) received both the B.Sc. and M.Eng. degrees in biomedical engineering from Zhejiang University, Hangzhou, China, in 1994 and 1997, respectively, and the Ph.D. degree in electrical and computer engineering from the National University of Singapore, Singapore, in 2004.

Since 2005, she has been with the School of Electrical and Electronic Engineering, Nanyang Technological University, where she is currently an Assistant Professor. From 1997 to 1998, she was a Teaching Assistant with Zhejiang University. She joined the Department of Electrical and Computer Engineering, National University of Singapore as a Post Master Fellow in 1998 and remained in the same department as a Research Engineer until 2004. In 2002, she was a Visiting Researcher at the Tampere University of Technology, Tampere, Finland and the Hong Kong Polytechnic University, Hong Kong, China. She joined the Temasek Laboratories at Nanyang Technological University as a Research Fellow in 2004. Her research interests include digital signal processing and VLSI circuits and systems design.

Yong Ching Lim (S'79–M'82–SM'92–F'00) received the A.C.G.I. and B.Sc. degrees in 1977 and the D.I.C. and Ph.D. degrees in 1980, all in electrical engineering, from Imperial College, University of London, U.K.

Since 2003, he has been with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, where he is currently a Professor. From 1980 to 1982, he was a National Research Council Research Associate in the Naval Postgraduate School, Monterey, CA. From 1982 to 2003, he was with the Department of Electrical Engineering, National University of Singapore. His research interests include digital signal processing (DSP) and VLSI circuits and systems design.

Dr. Lim was a recipient of the 1996 IEEE Circuits and Systems Society's Guillemin–Cauer Award, the 1990 IREE (Australia) Norman Hayes Memorial Award, 1977 IEE (U.K.) Prize and the 1974–77 Siemens Memorial (Imperial College) Award. He served as a lecturer for the IEEE Circuits and Systems Society under the distinguished lecturer program from 2001 to 2002 and as an associate editor for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS from 1991 to 1993 and from 1999 to 2001. He has also served as an Associate Editor for *Circuits, Systems and Signal Processing* from 1993 to 2000. He served as the Chairman of the DSP Technical Committee of the IEEE Circuits and Systems Society from 1998 to 2000. He served in the Technical Program Committee's DSP Track as the Chairman in IEEE ISCAS'97 and IEEE ISCAS'00 and as a Co-chairman in IEEE ISCAS'99. He is the General Chairman for IEEE APCCAS'06. He is a member of Eta Kappa Nu.