An Algorithm for the Design of Low-Power Hardware-Efficient FIR Filters

Mustafa Aktan, Arda Yurdakul, and Günhan Dündar, *Member, IEEE*

***Abstract*—A novel algorithm for designing low-power and hardware-efficient linear-phase finite-impulse response (FIR) filters is presented. The algorithm finds filter coefficients with reduced number of signed-power-of-two (SPT) terms given the filter frequency response characteristics. The algorithm is a branch-and-bound-based algorithm that fixes a coefficient to a certain value. The value is determined by finding the boundary values of the coefficient using linear programming. Although the worst case run time of the algorithm is exponential, its capability to find appreciably good solutions in a reasonable amount of time makes it a desirable CAD tool for designing low-power and hardware-efficient filters. The superiority of the algorithm on existing methods in terms of SPT term count, design time, hardware complexity, and power performance is shown with several design examples. Up to 30% reduction in the number of SPT terms is achieved over unoptimized Remez coefficients, which is 20% better than compared optimization methods. The average power saving is 20% over unoptimized coefficients, which is up to 14% better than optimized coefficients obtained with existing methods.**

***Index Terms*—Discrete coefficient finite-impulse-response (FIR) filter design, FIR digital filters, linear programming, multiplierless design, power-of-two coefficients.**

I. Introduction

CONSTANT coefficient finite-impulse-response (FIR) digital filters can be realized in parallel using as many multipliers as the number of coefficients in the filter. However, since multipliers are power- and area-consuming circuits, it is a common practice to first represent coefficients as sums of signed-power-of-two (SPT) terms and then replace the multipliers by shift and add circuits.

From the power perspective, the fewer the number of adders, the less power the filter will consume. The number of adders depends on the number of nonzero bits (SPT terms) of the quantized coefficients. A practical way to provide reduced SPT terms is to use canonic signed digit (CSD) representation since it offers fewer SPT terms in the representation than two's complement representation.

Methods proposed in the literature in general consider the reduction of hardware cost; however, they serve as a basis for the search for low power consumption. They can be grouped into two categories. The first group makes use of the redundant or repeated operations in multiplications of coefficients by data and will be referred to as subexpression sharing/elimination-based methods [1]–[6]. The second group of methods makes use of the idea that given the filter frequency characteristics (such as max. pass-band ripple or max. stop-band attenuation); the set of coefficients that satisfy the characteristics is not unique. Therefore, one can search for a coefficient set that has reduced number of SPT terms and hence better area/power performance. Optimal [7], [8] and suboptimal methods [9]–[13] have been proposed. There also have been attempts to combine the two approaches, where a set of coefficients with reduced number of ones is found and then subexpression elimination is applied on these coefficients [7], [13] or the search for reduced number of SPT terms is changed to a search for reduced number of adders [14].

Coefficient scaling is extensively used in methods that search for reduced number of SPT term coefficients [9]–[13]. Scaling all coefficients by the same factor changes the distribution of SPT terms among coefficients without altering the shape of the frequency response. This can greatly improve the search for reduced SPT terms by providing a good starting point for a local search. Based on this, a method is proposed that first searches for a scaling factor that minimizes the quantization error in the mean square sense [9]. In the second step of the algorithm, coefficients are optimized to minimize the normalized maximum ripple in the frequency response of the filter. An improved version of this method is proposed in [10], where the selection of the scaling factor is carried out considering the normalized pass-band ripple (NPR) of the frequency response of the filter. Then, a local search is performed starting from the quantized Remez coefficients scaled by the scaling factor that resulted in the minimum NPR. A common property of both algorithms is that they try to improve the frequency response of the filter by restricting the maximum number of SPT terms in each coefficient.

Another method that makes use of scaling is proposed in [12]. The algorithm differs from the previously mentioned methods in that the frequency-response characteristics of the filter are taken as a constraint rather than as an objective and, instead of trying to get the best frequency response (minimum NPR), tries to find a coefficient set under the restriction of maximum number of SPT terms per coefficient with fewer SPT terms.

An optimization method extensively used in the design of discrete coefficient filters is mixed integer linear programming (MILP). The main reason for using MILP is that it can find optimum discrete coefficients for which NPR is much better than coefficients obtained by simply rounding infinite precision coefficients [15]. Given the filter length and coefficient

Manuscript received November 27, 2006; revised August 3, 2007. First published February 7, 2008; last published July 10, 2008 (projected). This work was supported in part by The Scientific and Technological Research Council of Turkey (TUBITAK) under Project 104E139. This paper was recommended by Associate Editor T. Hinamoto.

M. Aktan and G. Dündar are with the Department of Electrical and Electronics Engineering, Bogazici University, Istanbul 34342, Turkey (e-mail: aktanmus@boun.edu.tr; dundar@boun.edu.tr).

A. Yurdakul is with the Computer Engineering Department, Bogazici University, Istanbul 34342, Turkey (e-mail: yurdakul@boun.edu.tr).

Digital Object Identifier 10.1109/TCSI.2008.917997

word-length, MILP can find the optimum frequency response [15]–[17]. However, the optimality criterion is NPR, not hardware or power cost. An MILP formulation where the optimization goal is to reduce the number of SPT terms, and hence hardware/power cost, is given in [8].

A drawback of MILP-based methods is that solution time increases exponentially with the increase of number of taps of a filter. Therefore, it is not practical to use MILP for filters having a large number of taps. In [13], a method that partially overcomes this problem by limiting the optimization variables to be the last D digits of the coefficients is introduced. D is taken to be no more than 3. The remainder of the digits are initialized to the values obtained by quantizing/rounding the Remez solution of the filter.

The desired frequency response of an FIR filter imposes boundaries on the values coefficients can have, thereby limiting the coefficient search space. A linear programming (LP) formulation is given in [7], by which the minimum and maximum values are found for each coefficient. Within these boundary values, possible coefficient values exceeding the maximum number of SPT terms allowed per coefficient are eliminated. Then, the feasibility of each possible combination of coefficient values is checked. This is done using a branch-and-bound (BAB)-based search. First checking the combination of coefficient values having the minimum number of SPT terms will eliminate the need for searching after a solution is found.

This paper presents the algorithm FIRGAM for designing low-power/area linear-phase FIR digital filters by reducing the number of SPT terms in the coefficients while keeping the quantization wordlength as small as possible. Unlike most algorithms present in the literature, it does not use passband scaling, but still produces better results. A desired property for an optimization tool is its ability to solve large problems as well as small problems. For the case of linear-phase FIR filters, the tool should be able to handle filters with a large number of taps. FIRGAM is able to find optimized discrete coefficient filters in a reasonable amount of time even for very large filters.

The remainder of this paper is organized as follows. Section II describes the proposed algorithm, FIRGAM. Section III discusses hardware implementation issues, such as parameters affecting the critical path of an FIR filter. The performance of FIRGAM is tested with several filters. The results are presented in Section IV, where a comparison with other existing methods is also given. Section V concludes the discussion.

II. FIRGAM Algorithm

Given the wordlength B and the desired filter characteristics $H_{\mathrm{d}}(\omega)$, the FIRGAM algorithm iteratively finds the coefficients of the resulting symmetric filter $H(\omega)$ within an acceptable error margin $\delta(\omega)$

$$|H(\omega) - H_{\mathrm{d}}(\omega)| \leq \delta(\omega). \tag{1}$$

Obviously, $H(\omega) = \sum_{m=o}^{M-1} h[m]T(\omega)$, where M is half the number of coefficients and $T(\omega)$ is the trigonometric function to obtain the frequency response of the filter generated by FIRGAM. In hardware implementation of FIR filters, symmetric filters are preferred for their linear-phase property to avoid waveform distortion and reduced number of multiplications. It should be noted that multiplications are the most power-consuming operations in an FIR filter, and symmetry halves the number of multiplications and, hence, the power consumption.

Let N denote the number of coefficients in $H(\omega)$. Since the filter is symmetric, then the FIRGAM algorithm should determine only M coefficients where

$$M = \left\lfloor \frac{N+1}{2} \right\rfloor. \tag{2}$$

There may be a lot of coefficients satisfying (1). However, once N is fixed, then the range of all possible values for each coefficient $h[i] (i = 0, 1, \ldots, M-1)$ can be determined by solving the following set of linear optimization problems independently:

$$\begin{aligned} h_{\min}[i] &= \min h[i] \\ \text{such that } & \left| \sum_{m=0}^{M-1} h[m]T(\omega) - H_{\mathrm{d}}(\omega) \right| \leq \delta(\omega), \quad \omega \in [0, \pi] \\ & h[m] : \text{free}, \qquad\qquad 0 \leq m < M \end{aligned} \tag{3}$$

$$\begin{aligned} h_{\max}[i] &= \max h[i] \\ \text{such that } & \left| \sum_{m=0}^{M-1} h[m]T(\omega) - H_{\mathrm{d}}(\omega) \right| \leq \delta(\omega), \quad \omega \in [0, \pi]. \\ & h[m] : \text{free}, \qquad\qquad 0 \leq m < M \end{aligned} \tag{4}$$

Coefficient $h[i]$ is a number that appears in this range, i.e., $h[i] \in ([h_{min}[i], h_{\max}[i]]$. Without the finite wordlength constraint, it can take over infinitely many numbers. However, since each coefficient is restricted to be represented with a finite wordlength B, all possible values of $h[i]$ form a finite set. Hence, let V_i be such a digital value set of $h[i]$. This set can be reduced further by forcing each value to contain at most $P < \lceil B/2 \rceil$ nonzero bits in the representation. This also helps low-power implementation because the power consumption is directly proportional to the number of SPT terms in coefficient representations. The selected coefficients from $\cup_{m=0}^{M-1} V_m$ must satisfy (1).

Experimentally, it has been observed that V_0 is the smallest subset of $\cup_{m=0}^{M-1} V_m$ and $|V_0|$ should not be lower than a certain value for a reasonably good search space to exist. To satisfy this, increasing the wordlength B or tap number N should be considered.

A simple BAB algorithm that runs on $\cup_{m=0}^{M-1} V_m$ will obviously find the optimal solution provided that enough memory and time are given. It is easy to observe that there exist $\Pi_{m=0}^{M-1} |V_m|$ combinations to be searched. However, a design automation tool should find a considerably good solution in a reasonable time using a reasonable amount of memory. The FIRGAM algorithm, whose pseudocode is given in Fig. 1, is a modified BAB algorithm; hence, its worst case performance is exponential. Yet, it reaches a reasonably good result in a sufficiently short time since it refines the search space during its execution. Obviously, a program that uses FIRGAM as a subroutine can also be developed to find the optimal solution.

The operation of the FIRGAM algorithm can be summarized as follows. Starting from the initial tap (i.e., $i = 0$), the algorithm iteratively selects a value v^* from the refined value set V_i^s

```
FIRGAM($M$, $L$, $H_d(\omega)$, $\bigcup_{m=0}^{M-1} V_m$ ) {
  $H^* = \varnothing$;
  $O = \infty$;
  Obtain $h^s_{\min}[0]$ and $h^s_{\max}[0]$ by solving equations (5) and (6);
  $V_0^s$ = VALUE_SELECT($L$, $h^s_{\min}[0]$, $h^s_{\max}[0]$, $V_0$);
  $i = 0$;
  while ($i \geq 0$) {
    if ($V_i^s \neq \varnothing$) {
      $h[i] = v^*$ such that $v^*$ is the first element of ordered set $V_i^s$;
      $H^* = H^* \cup \{h[i]\}$;
      $V_i^s = V_i^s - \{v^*\}$;
      if(there can be a solution having SPT terms less than $O$) {
        if ($i < M - 1$){
          $i = i + 1$;
          Obtain $h^s_{\min}[i]$ and $h^s_{\max}[i]$ by solving equations (5) and (6);
          $V_i^s$ = VALUE_SELECT($L$, $h^s_{\min}[i]$, $h^s_{\max}[i]$, $V_i$);
        }
        else if (problem is feasible){
          $H^*$ is a solution to the problem;
          $O$ = number of SPT terms of $H^*$;
        }
      }
    }
    else{
      $H^* = H^* - \{h[i]\}$;
      $i = i - 1$;
    }
  }
}

VALUE_SELECT($L$, $h^s_{\min}[i]$, $h^s_{\max}[i]$, $V_i$){
  $h^s_{\mathrm{mid}}[i] = (h^s_{\min}[i] + h^s_{\max}[i])/2$;
  $V_i^s = \{\forall\, v \in V_i \mid h^s_{\min}[i] \leq v \leq h^s_{\max}[i]\}$;
  Order $V_i^s=(v_1,v_2,\cdots,v_{|V_i^s|})$ such that $|v_1-h^s_{\mathrm{mid}}[i]| < |v_2-h^s_{\mathrm{mid}}[i]| < \cdots < |v_{|V_i^s|}-h^s_{\mathrm{mid}}[i]|$;
  Limit $V_i^s=(v_1,v_2,\cdots,v_{|V_i^s|})$ such that $|V_i^s| \leq L$;
  Return $V_i^s$;
}
```

Fig. 1. FIRGAM algorithm.

of tap coefficient $h[i]$ such that v^* is closest to the average of maximum and minimum values for $h[i]$. After this value is assigned to $h[i]$, v^* is removed from the value set and moved to the solution set H^*, which is an ordered set. In other words, the first element in H^* corresponds to $h[0]$ and the last element in H^* corresponds to $h[M-1]$. At each iteration, only one tap is fixed. After fixing each tap, the following pair of optimization equations is solved for the next tap:

$$h_{\min}[i] = \min h[i] \quad \text{such that} \left|\sum_{m=0}^{M-1} h[m]T(\omega) - H_{\mathrm{d}}(\omega)\right| \leq \delta(\omega),\ \omega \in [0,\pi] \quad h[m] : \begin{cases} \text{fixed}, & 0 \leq m < i \\ \text{free}, & i \leq m < M \end{cases} \tag{5}$$

$$h_{\max}[i] = \max h[i] \quad \text{such that} \left|\sum_{m=0}^{M-1} h[m]T(\omega) - H_{\mathrm{d}}(\omega)\right| \leq \delta(\omega),\ \omega \in [0,\pi] \quad h[m] : \begin{cases} \text{fixed}, & 0 \leq m < i \\ \text{free}, & i \leq m < M. \end{cases} \tag{6}$$

It should be noted that these equations are different from (3) and (4) so as to refine the search space for the ith tap after fixing $i-1$ taps. The superscript s in these equations and in the pseudocode stands for the refined variables. Obviously, fixing the values of $i-1$ taps will move the boundary values of the ith tap towards each other and the number of possible values for this tap will be reduced. If the refined value set (V_i^s) of $h[i]$ is not empty, then the iteration goes on as usual, but midvalue $h_{\mathrm{mid}}[i]$ might change due to the possibility that the moving amount of each boundary might not be equal. However, if the refined value set of $h[i]$ is empty, then the selected value for $h[i-1]$ is removed from H^* and the next value in the value set of $h[i-1]$ is selected for the next iteration, and this is repeated until a nonempty value set of $h[i]$ is obtained.

Experimentally, it has been observed that the algorithm finds an initial solution in less than $2M$ iterations depending on the tightness of the search space: If one can follow the aforementioned experimental rule (i.e., B is set to its lowest possible value), then the algorithm finds a solution in at most L^M iterations where L is a parameter used to limit the maximum size of the refined value sets V_i^s. If B is a bigger value, then a solution is obtained in approximately $2M$ iterations but the authors cannot state which one will always yield a better solution because this completely depends on the value sets of the coefficients. However, experimental results show that the solutions produced by this algorithm are much better than the ones found in the literature.

A. Performance Tuning

There are three main items that determine the performance of FIRGAM:

- prediction of the number of SPT terms for the cut-off mechanism;
- refined value set size (L);
- value selection order.

1) Prediction of the Number of SPT Terms: An advantage of BAB-based algorithms on enumeration-based algorithms is that they employ a cutoff mechanism that compares the best solution and best obtainable solution at any point of the search tree and decide not to go deeper in the search tree if a better solution cannot exist. This in turn avoids searching all possible combinations. The cutoff mechanism in FIRGAM is as follows. The number of SPT terms of the best solution is denoted as O (which is infinity when there is no solution). Let the number of SPT terms of the value v^* (used to fix coefficient $h[i]$) be denoted by p. Let the number of SPT terms in the coefficients that are already fixed be denoted by F_i. Let the minimum possible number of SPT terms of the unfixed coefficients be U_i. If $F_i + p + U_i = O$, there is no need to further branch to the next coefficient since there does not exist a better solution than the current solution. In this case, the algorithm proceeds by setting the current coefficient $h[i]$ to its next feasible value v^*. The main point here is to forecast the number of SPT terms of the unfixed coefficients ($h[i+1]$ to $h[M-1]$), namely U_i.

Three SPT term prediction strategies will be mentioned here.

a) U_MIN: V_i was defined to be the feasible value set of coefficient $h[i]$. In V_i, a value v which has the minimum number of SPT terms is found. Then, let $P_{i,\min}$ be the number of SPT terms of v. Then, the predicted minimum number of SPT terms for the unfixed coefficients while fixing coefficient $h[i]$ is

$$U_i = \sum_{m=i+1}^{M-1} P_{m,\min}. \tag{7}$$

b) U_AVG: Unfortunately, (7) gives an underestimated value to the actual number of SPT terms causing unnecessary search effort devoted to the end of the tree, i.e., the search is stuck at the last coefficients. Instead, another prediction value is needed that allows the whole search space to be searched in a reasonable amount of time. One can use the average number of SPT terms a coefficient can have by averaging the SPT terms of the values in the feasible value set (V_i). Then, let $P_{i,\text{avg}}$ be this value, which is calculated as

$$P_{i,\text{avg}} = \sum_{k=1}^{|V_i|} p_k / |V_i| \tag{8}$$

where p_k is the number of SPT terms of value $v_k \in V_i$. Then

$$U_i = \sum_{m=i+1}^{M-1} P_{m,\text{avg}}. \tag{9}$$

c) U_SUB_AVG: It is observed that the predicted number of SPT terms U_i in (9) is an overestimate. This causes the search to conclude in a very short time with a highly suboptimal result. Choosing the lower integer bound of the average instead of the average itself is the best choice in terms of search time–optimality tradeoff. Now the predicted number of SPT terms for $M - i$ unfixed coefficients is

$$U_i = \sum_{m=i+1}^{M-1} \lfloor P_{m,\text{avg}} \rfloor. \tag{10}$$

2) Refined Value Set Size (L): As mentioned previously, the size of the refined value set (V_i^s) is limited to a certain value L. This is when selecting the values from the feasible value set V_i^s of coefficient $h[i]$, L values in the vicinity of $h_{\text{mid}}^s[i]$ are selected. If L is set to infinity, all values in V_i^s are selected. The purpose of limiting the size of the refined feasible value set is to reduce search time by avoiding unnecessary search devoted to possibly infeasible combinations. How can one predict these infeasible combinations? Experimentally it is observed that feasible solutions are the combinations of coefficient values which are selected in the neighborhood of the middle values, $h_{\text{mid}}^s[i]$. The values close to the boundary values of $h[i]$ either do not or rarely produce feasible solutions. Practically setting $L = 2$ is enough.

3) Value Selection Order: When selecting values from the refined value set (V_i^s) to fix coefficient $h[i]$, one can use different strategies. The strategy adopted in FIRGAM is selecting the value closest to $h_{\text{mid}}^s[i]$ first, which is phrased in the algorithm in Fig. 1 as the function VALUE_SELECT. This strategy is called MID_VAL_FIRST. The selection order of the values could also be done by first selecting the values having the least number of SPT terms. This is done by first constructing the set V_i^s by calling the function VALUE_SELECT in Fig. 1 and then reordering the values in V_i^s according to their number of SPT terms from low to high. This strategy is called MIN_SPT_FIRST. To our observations, MIN_SPT_FIRST should not be used unless the coefficient wordlength is greater than the minimum possible wordlength.

TABLE I
FILTER CHARACTERISTICS OF THE FIRGAM EXAMPLE

Maximum passband ripple	(δ_p)	=	0.05
Normalized passband edge frequency	(f_p)	=	0.10
Maximum stopband ripple	(δ_s)	=	0.05
Normalized stopband edge frequency	(f_s)	=	0.25

TABLE II
VALUE SETS OF THE COEFFICIENTS FOR THE FIRGAM EXAMPLE

V_0	=	{-3,-2,-1}
V_1	=	{-4,-3,-2,-1}
V_2	=	{1,2,3,4}
V_3	=	{12,13}
V_4	=	{20,21,22}

B. Run-Time of FIRGAM

The exponential run-time $(O(L^M))$ of the algorithm is due to the BAB nature of the algorithm. However, this is the total search time of the algorithm. As it would be clear from the experimental results in the following section, an initial solution or an acceptably good solution can be obtained in a reasonable amount of time. The optimum solution could be found 100 times faster than other optimum methods with exponential run-time complexity.

There is always a possibility for FIRGAM to find an initial solution in exponential time. To our observations, this might occur when the coefficient wordlength is the minimum possible value and the filter length is the minimum filter length that satisfies this wordlength. Since there is no clear formula that gives the minimum filter length for which we can find the minimum wordlength filter, one has to try a range of filter lengths. In general, FIRGAM finds a solution after $2M$ iterations where M is half the filter length. One can allow FIRGAM to run for a predetermined number of iterations and in case FIRGAM cannot find a solution increase the filter length. Another way is to limit the run-time of the algorithm, that is, FIRGAM is terminated when it cannot find any solution after a predetermined amount of time. Again, then the filter length is increased until a solution is found.

C. FIRGAM Example

A linear-phase low-pass FIR filter with ten taps is to be designed, i.e., $N = 10$, $M = 5$. The coefficients are quantized to 7 b $(B = 7)$. The desired filter characteristics are given in Table I. The boundary values of the coefficients are found using (3) and (4). The value sets $(\cup_{m=0}^{M-1} V_m)$ formed using the boundary values are listed in Table II.

The feasible coefficient values are given as integers for ease of demonstration. The actual values can be found by dividing the values with 2^6. The refined values set sizes are limited to 2, i.e., $L = 2$. The SPT term prediction strategy is U_MIN. The value selection strategy is MID_VAL_FIRST.

The algorithm starts by finding the boundary values of the first coefficient $h[0]$. They are found to be $h_{\min}^s[0] = -3.62$ and $h_{\max}^s[0] = -0.83$, from which the middle value is found

TABLE III
ITERATIVE SOLUTION OF FIRGAM EXAMPLE

i	$h^s_{\min}[i]$	$h^s_{\max}[i]$	$h^s_{\text{mid}}[i]$	V_i^s	$h[i]$	H^*	O	Status
0	-3.62	-0.83	-2.24	{-2,-3}				
				{-3}	-2	{-2}	∞	
1	-4.38	-1.26	-2.82	{-3,-2}				
				{-2}	-3	{-2,-3}		
2	1.01	3.57	2.29	{2,3}				
				{3}	2	{-2,-3,2}		
3	12.07	13.45	12.76	{13}				
				{}	13	{-2,-3,2,13}		
4	21.18	22.26	21.72	{22}				
				{}	22	{-2,-3,2,13,22}	10	S
3				{}		{-2,-3,2}		
2				{3}				
				{}	3	{-2,-3,3}		
3	11.97	13.15	12.56	{12,13}				
				{13}	12	{-2,-3,3,12}		
4	20.69	20.78	20.74	{}		{-2,-3,3,12}		I
3				{13}				
				{}	13	{-2,-3,3,13}		C
2				{}		{-2,-3}		
1				{-2}				
				{}	-2	{-2,-2}		
2	1.75	3.99	2.87	{3,2}				
				{2}	3	{-2,-2,3}		
3	12.03	13.66	12.85	{13}				
				{}	13	{-2,-2,3,13}		
4	20.69	21.37	21.03	{21}				
				{}	21	{-2,-2,3,13,21}		C
3				{}		{-2,-2,3}		
2				{2}				
				{}	2	{-2,-2,2}		
3	12.65	13.48	13.06	{13}				
				{}	13	{-2,-2,2,13}		
4	20.69	21.26	20.98	{21}				
				{}	21	{-2,-2,2,13,21}	9	S
3				{}		{-2,-2,2}		
2				{}		{-2,-2}		
1				{}		{-2}		
0				{-3}				
				{}	-3	{-3}		
1	-3.62	-1.28	-2.45	{-2,-3}				
				{-3}	-2	{-3,-2}		
2	2.65	4.01	3.33	{3,4}				
				{4}	3	{-3,-2,3}		C
				{}	4	{-3,-2,4}		
3	13.13	13.16	13.15	{}				I
2				{}				
1				{-3}				
				{}	-3	{-3,-3}		C
0				{}		{}	9	

as $h^s_{\text{mid}}[0] = -2.24$. The values in V_0 within the boundary values are $\{-3,-2,-1\}$. Since the refined value set size is limited to 2, the two values that are closest to $h^s_{\text{mid}}[0]$ are selected, which makes $V_0^s = \{-2,-3\}$. The value closest to $h^s_{\text{mid}}[0] = -2.24$ in $V_0^s = \{-2,-3\}$ is -2, so $h[0] = -2$ and $H^* = \{-2\}$, and -2 is removed from the refined value set making $V_0^s = \{-3\}$. The algorithm branches to the next coefficient $h[1]$. The refined boundary values are found to be $h^s_{\min}[1] = -4.38$ and $h^s_{\max}[1] = -1.26$, making $h^s_{\text{mid}}[1] = -2.82$. The refined value set is $V_1^s = \{-3,-2\}$. $h[1]$ is set to -3, making $H^* = \{-2,-3\}$ and $V_1^s = \{-2\}$. The algorithm branches to the next coefficient $h[2]$. The search goes on until all values in V_0^s are tried.

The values of the variables at each iteration during the whole search process are given in Table III. The status column in the table tells the current status of the problem. "S" indicates that a solution is found. "C" resembles a cutoff situation, meaning that a better solution cannot be found going deeper in the search tree. "I" denotes an infeasible situation where an empty refined value set is encountered for the next coefficient after fixing the current coefficient.

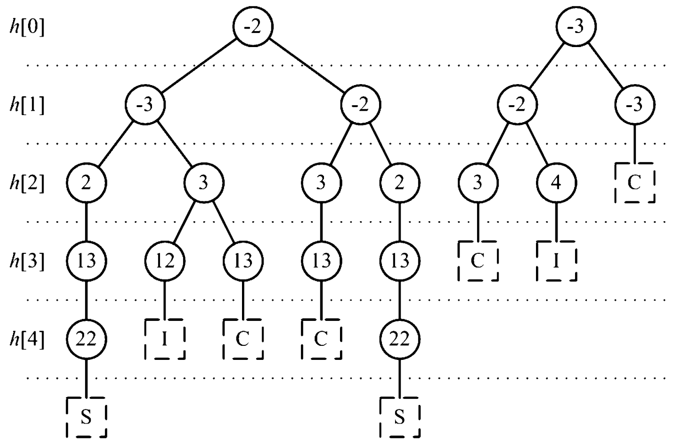

Fig. 2. Search tree for FIRGAM example (S: solution; I: infeasible; C: cutoff).

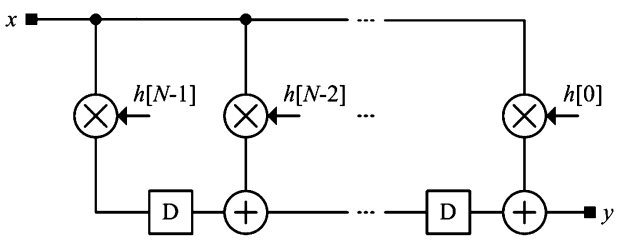

Fig. 3. Transposed form realization of an FIR filter.

The search is also demonstrated as a search tree in Fig. 2, where the leftmost branches are the first branched values. The values in the circles are the values to which the corresponding coefficients are fixed.

III. Hardware Implementation Issues

In the hardware realization of constant coefficient FIR filters, adders/subtractors replacing the multipliers are called multiplier adders. The inter-tap adders are called structural adders [4]. One may choose among different adder topologies for the multiplier and structural adders. In terms of power and area, the ripple carry adder seems to be a good choice. The delay elements can be realized with D-type flip-flops.

The transposed form realization of a FIR filter, which is shown in Fig. 3, is preferred because of its short critical path offering high-speed operation. The worst case critical path is one multiplier + one adder long, which is independent of the length of the filter, i.e., number of coefficients.

When a coefficient multiplier is replaced with an adder tree, the number of adders and the depth of the adder tree depend on the number of SPT terms in the coefficient. Then, letting the number of SPT terms in coefficient $h[i]$ be P_i, the depth of the adder tree used to replace the multiplier will be at most $P_i - 1$ adders. If the maximum number of SPT terms per coefficient is $P_{\max}$, then the contribution of a replaced multiplier to the critical path will be at most $P_{\max} - 1$ adders. Then for the transposed form realization of an FIR filter the critical path including the structural adder consists of $P_{\max}$ adders in the worst case. Thus, by limiting the number of SPT terms per coefficient, one can limit the critical path length, and hence increase the performance of the filter. This formulation, however, neglects the effect of the structure of the adders. The length of the critical path is affected by the structure of the adders.

Here, an analysis for ripple carry adders is given. Ripple carry adders are composed of full adders. The critical path is the path for the carry signal generated by the least significant input bits to propagate to the most significant output sum. The contribution of an adder to the critical path can be approximated to be equal to the adder length of full adders. For example, an 8-b ripple carry adder will consist of eight full adders and hence will have an eight-full-adder-long critical path for the carry signal to travel to the most significant output sum. For an FIR filter, the maximum size of an adder, which is at the same time the output wordlength, can be calculated by the following formula:

$$\begin{aligned} W_{\max} &= W_{\text{in}} + \left\lceil \log_2 \sum_{n=0}^{N-1} |h[n]|\, 2^{B-1} \right\rceil \\ &= W_{\text{in}} + B - 1 + \left\lceil \log_2 \sum_{n=0}^{N-1} |h[n]| \right\rceil \end{aligned} \tag{11}$$

where W_{in} is the input data wordlength (including sign bit), $h[n]$ is the filter coefficients $(-1 \le h[n] < 1)$, N is the number of coefficients in the filter, and B is the wordlength of the coefficients (including sign bit). $W_{\max}$ is the minimum adder size that ensures that no overflow will occur. This size is a limiting size for structural adders. The size of the multiplier adders will always be less than $W_{\max}$.

The critical path of an FIR filter having at most $P_{\max}$ SPT terms in a coefficient will be

$$\text{CP} = W_{\max} + P_{\max} - 1 \tag{12}$$

full adders long. This is also shown in Fig. 4. From (12), it is clear that the critical path does not depend only on the maximum number of SPT terms $(P_{\max})$ that a coefficient can have, but also on the wordlength (B) of the coefficients. Therefore, keeping the wordlength small is an alternative method for critical path-length minimization. Furthermore, when the multiplier adder tree depth can be kept to a minimum, i.e., the depth is $\lceil \log_2 P_{\max} \rceil$ adders, reducing the wordlength is even a more effective way (linear versus logarithmic dependence).

IV. Experimental Results

Here, several example filters are designed using FIRGAM and compared with other methods from the literature. The compared methods are the Remez algorithm, the trellis algorithm [12], Li's algorithm [11], Lim's algorithm [18], Samueli's algorithm [10], Yao's algorithm [13], and the MILP-based optimum method of [8] and are denoted by the abbreviations RMZ, TRE, LI, LIM, SAM, PMILP, and MILP, respectively. For the Remez algorithm, MATLAB's Remez function is used, and the filter coefficients are quantized to the minimum wordlength satisfying the filter characteristics. The results for LIM, SAM, and

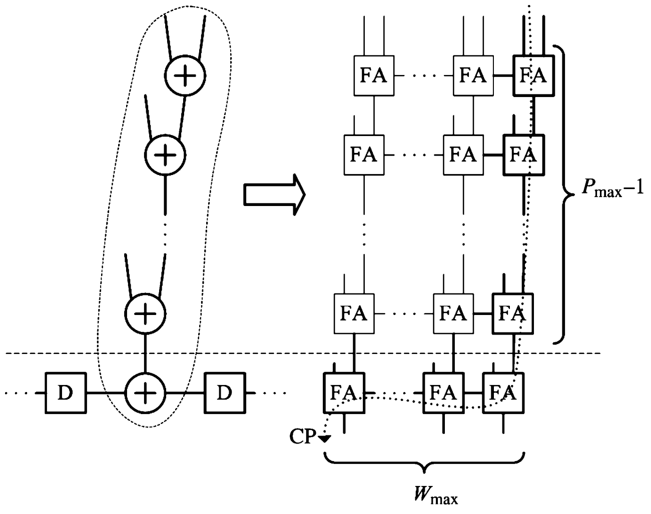

Fig. 4. Critical path in an FIR filter.

TABLE IV
Frequency-Response Characteristics of the Filters of Example 1

Filter	Type	δ_p (dB)	f_p	δ_s (dB)	f_s
L1	High-pass	0.10	0.400	-80	0.37
L2	Low-pass	0.50	0.100	-60	0.14
S2	Low-pass	0.20	0.021	-60	0.07

PMILP are taken directly from the papers in which they appeared. The algorithms TRE, LI, and FIRGAM are implemented in the C programming language. FIRGAM uses the LP library of QSOPT [18]. For the optimum MILP method, ILOG CPLEX's [20] optimization package is used. The frequency grid on which the frequency response is evaluated consists of 1000 frequency samples for all filters. This makes 2000 constraints for the LP problem of FIRGAM. The implemented algorithms are run on a PC with a Pentium D 2.8-GHz processor and 1-GB RAM. The hardware realizations of the filters are done using the transposed form structure. For all filters, the subexpression sharing method in [1] is used. The input data width of the filters is taken as 8 b. Adders are realized with ripple carry adders. Delay elements are realized with D-type flip-flops. The hardware cost of the adders and delay elements are both measured in terms of the number of adders and bit-wise components, namely full adders for adders and D-type flip-flops for the delay elements. All of the adder and delay element sizes are computed so that no overflow occurs. The reduction of the number of full adders used in an adder due to shifted inputs is also considered [4], [6]. The filters are mapped to the AMS 0.35-μ technology cell library for power simulations. The simulations are done with an event-driven gate-level simulator written in C++. The operating voltage is 3.3 V. The input data applied to the filters is a random sequence of 16 384 8-b samples.

Example 1: The desired frequency-response characteristics of the filters are given in Table IV. δ_p is the passband peak-to-peak ripple, δ_s is the stopband attenuation, f_p is the normalized passband edge frequency, and f_s is the normalized stopband edge frequency. L1 and L2 are the example filters 1 and 2 given in [18]. S2 is the second example filter of [10]. The methods

TABLE V
PROPERTIES OF DESIGNED FILTERS OF EXAMPLE 1

Filter	Method	N	B	B_e	P	SPT Term		# of adders			Hardware complexity			Power	
						#	Gain (%)	MA	SA	Total	# of FA	# of DFF	Total	μW/MHz	Gain (%)
L1	RMZ	121	19	16	8	483	-	78	120	198	3647	2907	6554	1090	-
	LIM	121	17	14	7	416	13.9	61	120	181	3146	2664	5810	938	13.9
	PMILP	121	-	15	6	442	8.5	59	120	179	3248	2748	5996	988	9.4
	FIRGAM	121	17	14	6	362	25.1	47	120	167	2886	2654	5540	864	20.7
L2	RMZ	63	15	12	5	197	-	30	62	92	1510	1281	2791	422	-
	LIM	63	13	10	4	159	19.3	23	62	85	1311	1154	2465	349	17.3
	PMILP	63	-	11	4	163	17.3	19	62	81	1285	1182	2467	344	18.4
	FIRGAM	63	13	10	4	140	28.9	18	62	80	1213	1154	2367	328	22.3
S2	RMZ	60	16	12	6	204	-	32	59	91	1494	1251	2745	421	-
	SAM	60	-	13	4	174	14.7	32	59	91	1455	1298	2753	402	4.5
	PMILP	60	-	12	4	174	14.7	25	59	84	1342	1255	2597	368	12.6
	FIRGAM	60	15	11	5	160	21.6	27	59	86	1337	1180	2517	362	14.0

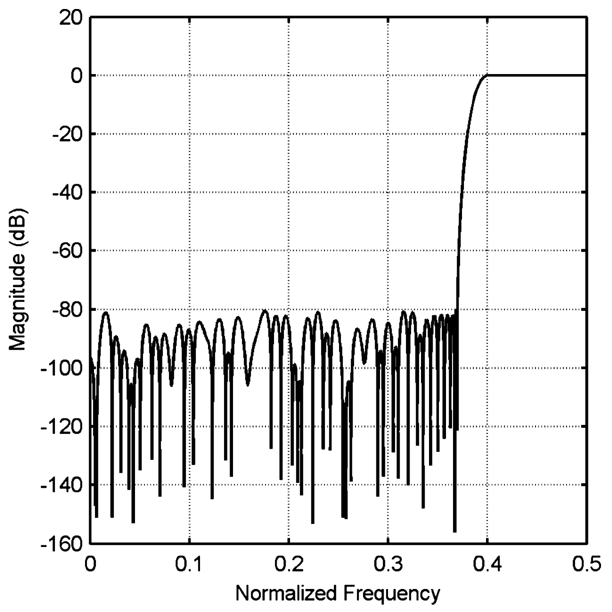

Fig. 5. Frequency response of FIRGAM filter L1 with $N = 121, B = 17$.

TABLE VI
SOLUTION TIMES FOR FIRGAM FILTERS OF EXAMPLE 1

Filter	First Solution		Best Solution		Run Time
	# of SPT	Time	# of SPT	Time	
L1	419	2m	362	32m	56m
L2	157	15s	140	26m	54m
S2	182	12s	160	23m	27m

TABLE VII
COEFFICIENTS OF FIRGAM FILTER L1 WITH $N = 121, B = 17$

$h[0] = 2^{-14}$	$h[30] = 2^{-8} + 2^{-11} + 2^{-13}$
$h[1] = -2^{-13} - 2^{-15}$	$h[31] = -2^{-7} + 2^{-9} - 2^{-12}$
$h[2] = 2^{-12} + 2^{-16}$	$h[32] = 2^{-8} + 2^{-10} - 2^{-14}$
$h[3] = -2^{-12} - 2^{-14} - 2^{-16}$	$h[33] = -2^{-10} + 2^{-13}$
$h[4] = 2^{-12} + 2^{-15}$	$h[34] = -2^{-8} - 2^{-12} - 2^{-14}$
$h[5] = -2^{-14} - 2^{-16}$	$h[35] = 2^{-7} + 2^{-13}$
$h[6] = -2^{-12}$	$h[36] = -2^{-7} - 2^{-11} + 2^{-13} - 2^{-16}$
$h[7] = 2^{-11} + 2^{-14} + 2^{-16}$	$h[37] = 2^{-8} + 2^{-11} - 2^{-13} + 2^{-16}$
$h[8] = -2^{-10} + 2^{-12}$	$h[38] = 2^{-9} + 2^{-11} + 2^{-16}$
$h[9] = 2^{-11} + 2^{-13}$	$h[39] = -2^{-7} - 2^{-10} - 2^{-12}$
$h[10] = -2^{-13} - 2^{-15}$	$h[40] = 2^{-6} - 2^{-8} + 2^{-12} + 2^{-14}$
$h[11] = -2^{-11} - 2^{-15}$	$h[41] = -2^{-7} - 2^{-9} + 2^{-11}$
$h[12] = 2^{-10} + 2^{-13} + 2^{-15}$	$h[42] = 2^{-10} + 2^{-12}$
$h[13] = -2^{-9} + 2^{-11} + 2^{-13} - 2^{-16}$	$h[43] = 2^{-7} + 2^{-10} + 2^{-13}$
$h[14] = 2^{-10} + 2^{-15}$	$h[44] = -2^{-6} - 2^{-11} - 2^{-13} - 2^{-15}$
$h[15] = -2^{-13} + 2^{-16}$	$h[45] = 2^{-6} + 2^{-10} - 2^{-14}$
$h[16] = -2^{-10} - 2^{-15}$	$h[46] = -2^{-7} - 2^{-11} + 2^{-14} - 2^{-16}$
$h[17] = 2^{-9} - 2^{-13} - 2^{-16}$	$h[47] = -2^{-7} + 2^{-9} - 2^{-13}$
$h[18] = -2^{-9} + 2^{-13} - 2^{-15}$	$h[48] = 2^{-6} + 2^{-8} + 2^{-11}$
$h[19] = 2^{-10} - 2^{-15}$	$h[49] = -2^{-5} + 2^{-8} + 2^{-10} - 2^{-13} - 2^{-15}$
$h[20] = 2^{-11} + 2^{-13} + 2^{-15}$	$h[50] = 2^{-6} + 2^{-8} + 2^{-10} - 2^{-13}$
$h[21] = -2^{-9} - 2^{-12}$	$h[51] = -2^{-9} + 2^{-12}$
$h[22] = 2^{-8} - 2^{-10}$	$h[52] = -2^{-5} + 2^{-7} + 2^{-12} + 2^{-14}$
$h[23] = -2^{-9} - 2^{-11} + 2^{-13} + 2^{-16}$	$h[53] = 2^{-4} - 2^{-6} - 2^{-8}$
$h[24] = 2^{-11} - 2^{-13} - 2^{-16}$	$h[54] = -2^{-4} + 2^{-6} + 2^{-10} + 2^{-12} - 2^{-16}$
$h[25] = 2^{-9} + 2^{-12} - 2^{-14}$	$h[55] = 2^{-5} - 2^{-7} - 2^{-11} + 2^{-13}$
$h[26] = -2^{-8} - 2^{-13} + 2^{-15}$	$h[56] = 2^{-5} - 2^{-7} + 2^{-9} - 2^{-12} + 2^{-14}$
$h[27] = 2^{-8} + 2^{-12}$	$h[57] = -2^{-3} + 2^{-5} + 2^{-8} - 2^{-11} - 2^{-16}$
$h[28] = -2^{-9} - 2^{-12} - 2^{-15}$	$h[58] = 2^{-3} + 2^{-5} + 2^{-12}$
$h[29] = -2^{-10} - 2^{-12} + 2^{-14}$	$h[59] = -2^{-2} + 2^{-4} - 2^{-6} - 2^{-9} - 2^{-11} - 2^{-13}$
	$h[60] = 2^{-2} - 2^{-5} + 2^{-8} + 2^{-10} + 2^{-12} - 2^{-16}$
$h[n] = h[120-n]$ for n=61, 62, 63, … ,120	

TABLE VIII
FREQUENCY-RESPONSE CHARACTERISTICS OF THE FILTERS OF EXAMPLE 2

Filter	Type	δ_p	f_p	δ_s	f_s	N_{min}
A	Low-pass	0.010	0.0625	0.001	0.1125	56
B	Low-pass	0.010	0.1000	0.010	0.120	101
C	Low-pass	0.005	0.0625	0.005	0.070	311

used for comparison are the Remez algorithm (RMZ), Lim's algorithm (LIM) [18], Samueli's algorithm (SAM) [10], and the PMILP algorithm [5], [13].

The properties of the designed filters are given in Table V. Here, N is the number of taps, B is the wordlength (including sign bit) of the coefficients, B_e is the effective wordlength excluding the sign bit and most significant zero bits [18], and P is the maximum number of SPT terms found in a coefficient. The number of adders is given in terms of multiplier adders (MA) and structural adders (SA). The number of multiplier adders is the number after subexpression elimination is applied. The total hardware complexity after realizing the adders with ripple carry adders and the delay elements with D-type flip-flops is the sum of the number of full adders (FA) and number of D-type flip-flops (DFF). The power performance of the filters is given as $\mu\mathrm{W/MHz}$. The SPT term and power gains are calculated taking

TABLE IX
PROPERTIES OF DESIGNED FILTERS OF EXAMPLE 2

Filter	Method	N	B	B_e	P	SPT Term		# of adders			Hardware Complexity			Power	
						#	Gain (%)	MA	SA	Total	# of FA	# of DFF	Total	μW/MHz	Gain (%)
A	RMZ	59	15	12	6	200	-	29	58	87	1449	1183	2632	412	-
	TRE	59	13	10	5	160	20.0	23	58	81	1263	1102	2365	340	17.4
	LI	59	13	10	4	151	24.5	21	58	79	1218	1097	2315	325	21.1
	MILP	59	13	10	5	145	27.5	18	58	76	1173	1063	2236	317	23.1
	FIRGAM	59	13	10	5	145	27.5	18	58	76	1173	1063	2236	317	23.1
B	RMZ	105	13	10	5	232	-	24	102	126	2007	1958	3965	549	-
	TRE	105	12	9	3	199	14.2	15	102	117	1814	1836	3650	464	15.0
	LI	105	12	9	4	212	8.6	20	100	120	1853	1843	3696	487	11.3
	MILP	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	FIRGAM	105	11	8	4	169	27.2	11	100	111	1672	1739	3411	426	22.4
C	RMZ	325	15	12	5	820	-	57	322	379	7045	6818	13863	1884	-
	TRE	325	14	11	5	743	9.4	43	322	365	6650	6665	13315	1721	8.7
	LI	325	14	11	6	740	9.8	42	320	362	6608	6618	13226	1712	9.1
	MILP	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	FIRGAM	325	13	10	4	549	33.0	22	306	328	5687	6148	11835	1450	23.0

the Remez (RMZ) filters as reference. As an example, the frequency response and coefficients of the FIRGAM filter L1 are given in Fig. 5 and Table VII, respectively.

Looking at the results in Table V, it is clear that FIRGAM outperforms all methods in terms of the number of SPT terms. In terms of wordlength, FIRGAM coefficients have either shorter or equal wordlength when compared with the other methods. The reduced SPT term count and wordlength has reduced the hardware cost and power consumption. Again, looking at Table V, FIRGAM filters have the least hardware cost in terms of full adders and D-type flip-flops. Note that the reduction in the number of SPT terms (taking the RMZ coefficients as reference) for the FIRGAM filters is in parallel with the reduction in power consumption.

The time spent to find the filters for FIRGAM is given in Table VI. The feasible value set size L for FIRGAM was set to 2. The number of SPT terms per coefficient was not limited. The SPT term prediction and value selection strategies used were U_SUB_AVG and MID_VAL_FIRST, respectively. The first solution is the initial solution found by FIRGAM. The best solution is best in terms of both the total number of SPT terms of the coefficients and maximum number of SPT terms in a coefficient. Since the compared algorithms were not implemented, a direct comparison to their search time was not possible.

Example 2: In this example, FIRGAM is compared to the Trellis algorithm (TRE) [12], Li's algorithm (LI) [11], and MILP formulation of [8]. The algorithms are implemented in the C programming language. The comparison is made with the filters listed in Table VIII. $N_{\min}$ is the minimum number of taps required to realize each filter.

The properties of the designed filters are given in Table IX. The run-times of the algorithms are listed in Table X. For the MILP method, passband scaling could be used, however, to make a direct comparison with the results of FIRGAM, the passband gain is taken to be unity, i.e., $s = 1$ in [8]. LI and TRE directly make use of passband scaling in their optimization.

TABLE X
TIME SPENT BY EACH ALGORITHM TO DESIGN THE FILTERS OF EXAMPLE 2

Filter	Method	First solution		Best solution		Run time
		# of SPT	Time	# of SPT	Time	
A	RMZ	200	-	200	-	-
	TRE	160	38m	160	38m	2h 57m
	LI	151	1s	151	1s	4s
	MILP	147	13d	145	14d	15d
	FIRGAM	153	17m	145	3h 2m	4h 14m
B	RMZ	232	-	232	-	-
	TRE	215	48m	199	3h 19m	4h 32m
	LI	221	1s	212	1s	3s
	MILP	-	-	-	-	24h*
	FIRGAM	179	1m	169	9m	24h*
C	RMZ	820	-	820	-	-
	TRE	767	55m	743	20h 46m	24h*
	LI	818	1s	740	18s	41s
	MILP	-	-	-	-	24h*
	FIRGAM	583	18m	549	13h 47m	24h*

*manually stopped after 24 hours

LI searches for a range of scaling factors to find the optimum coefficients in terms of the number of SPT terms. Although TRE uses passband scaling, no method is given in [12] to determine a proper scaling factor. Since TRE is a polynomial time algorithm, multiple runs of the algorithm on different scaling factors is possible. Therefore, TRE is run for the same range of scaling factors used by LI. The run time of the algorithm TRE given in Table X is the result of multiple runs of the algorithm. The algorithms are not allowed to run more than 24 h. The only exception is made for MILP when run on filter A. The reason is that filter A is the smallest filter and is the one that is most likely to produce a result in a reasonable amount of time.

In terms of search time, LI seems to be the best algorithm. However, for large filters, the gain in the number of SPT terms

TABLE XI
PROPERTIES OF THE FIRGAM FILTERS OF EXAMPLE 3

Filter	N	B	B_e	P	SPT Term		# of adders			Hardware complexity			Power	
					#	Gain (%)	MA	SA	Total	# of FA	# of DFF	Total	μW/MHz	Gain (%)
A	56	19	16	6	244	-	42	55	97	1708	1330	3038	518	-
	57	15	12	5	159	34.8	24	56	80	1264	1138	2402	352	32.0
	58	14	11	5	144	40.9	21	57	78	1190	1097	2287	315	39.2
	59	13	10	5	145	40.6	18	58	76	1173	1063	2236	317	38.8
B	101	14	11	4	235	-	24	100	124	2017	1985	4002	547	-
	102	13	10	4	206	12.3	18	97	115	1793	1893	3686	471	13.9
	103	12	9	4	189	19.6	16	102	118	1806	1813	3619	467	14.6
	105	11	8	4	169	28.1	11	100	111	1672	1739	3411	426	22.1
C	311	16	13	5	771	-	58	306	364	6771	6840	13611	1815	-
	312	15	12	5	674	12.6	43	305	348	6259	6547	12806	1657	8.7
	317	14	11	4	660	14.4	31	312	343	6219	6333	12552	1600	11.8
	325	13	10	4	549	28.8	22	304	326	5687	6148	11835	1450	20.1

obtained by FIRGAM compensates for the search time. Note also that even the initial solutions obtained by FIRGAM are much better than the best solutions obtained by LI. It is interesting to note that the best solution found for filter A by FIRGAM is the optimum solution. The complete search took only 4 h for FIRGAM, whereas it took 15 days for the optimal MILP method. Hence, FIRGAM is almost 100 times faster than the optimal MILP method. For filters B and C, the MILP method could not find any solutions in one day.

Example 3: For the same frequency-response characteristics of the filters of example 2, filters are redesigned with different numbers of taps using FIRGAM. The number of taps is increased starting from the minimum possible to the number beyond which no reduction in the coefficient wordlength could be achieved. The properties of the filters are given in Table XI. Power gain is calculated taking the filter having the minimum number of taps as reference.

An additional tap requires an additional structural adder and delay element. Therefore, increasing the number of taps, at first glance, might seem to increase the hardware cost. However, this increase is compensated for by the reduction of the wordlength of the coefficients, which is clearly shown in Table XI.

V. CONCLUSION

An algorithm for designing low-power/hardware-cost linear-phase FIR filters was presented. The algorithm optimizes SPT terms in the coefficients given the filter frequency-response characteristics. Although the worst case run-time of the algorithm is exponential, its capability to find appreciably good solutions in a reasonable amount of time (at least 100 times faster than traditional optimum MILP based formulation) makes it a desirable CAD tool for designing low-power/hardware-cost linear-phase FIR filters. The algorithm is compared with existing methods with several examples. The filters found by the proposed algorithm have fewer SPT terms and are shorter in wordlength than the filters found by the other methods. FIRGAM filters consume 20% less power on average than unoptimized Remez coefficients. The superiority over existing methods has been shown to be more apparent on high-order filters.

FIRGAM is a generic method for designing finite wordlength filters. Besides its SPT term optimization capability, the FIRGAM algorithm can be used as a standalone discrete coefficient filter design algorithm.

REFERENCES

[1] A. Yurdakul and G. Dündar, "Fast and efficient algorithm for the multiplierless realisation of linear DSP transforms," in *IEE Proc. Circuits, Devices Syst.*, 2002, vol. 149, pp. 205–211.
[2] R. Hartley, "Subexpression sharing in filters using canonic signed digit multipliers," *IEEE Trans. Circuits Systs. II, Analog Digit. Signal Process.*, vol. 43, no. 10, pp. 677–688, Oct. 1996.
[3] R. Pasko, P. Schaumont, V. Derudder, S. Vernalde, and D. Iuraekova, "A new algorithm for elimination of common subexpressions," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 18, no. 1, pp. 58–68, Jan. 1999.
[4] 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.-Aided Design Integr. Circuits Syst.*, vol. 24, no. 2, pp. 295–304, Feb. 2005.
[5] C.-Y. Yao, H. H. Chen, C.-Y. Chien, and C.-T. Hsu, "A high-level synthesis procedure for linear-phase fixed-point FIR filters with SPT coefficients," *Int. J. Elect. Eng.*, vol. 12, no. 1, pp. 75–84, 2005.
[6] K. Johansson, O. Gustafsson, and L. Wanhammar, "A detailed complexity model for multiple constant multiplication and an algorithm to minimize the complexity," in *Proc. Eur. Conf. Circuit Theory Design*, Cork, Ireland, 2005, vol. 3, p. 465-468.
[7] J. Yli-Kaakinen and T. Saramäki, "A systematic algorithm for the design of multiplierless FIR filters," in *Proc. IEEE Int. Symp. Circuits Syst.*, Sydney, Australia, May 2001, vol. 2, pp. 185–188.
[8] O. Gustafsson, H. Johansson, and L. Wanhammar, "An MILP approach for the design of linear-phase FIR filters with minimum number of signed-power-of-two terms," in *Proc. Eur. Conf. Circuit Theory Design*, Espoo, Finland, Aug. 2001, vol. 2, p. 217–220.
[9] Q. F. Zhao and Y. Tadokoro, "A simple design of FIR filters with power-of-two coefficients," *IEEE Trans. Circuits Syst.*, vol. 35, no. 5, pp. 566–570, May 1988.
[10] 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.
[11] D. Li, J. Song, and Y. C. Lim, "A polynomial-time algorithm for designing digital filters with power-of-two coefficients," in *Proc. IEEE Int. Symp. Circuits Syst.*, Chicago, IL, May 1993, vol. 1, pp. 84–87.
[12] C. Chen and A. N. Willson, Jr., "A trellis search algorithm for the design of FIR filters with signed-powers-of-two coefficients," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 46, no. 1, pp. 29–39, Jan. 1999.
[13] C.-Y. Yao and C.-J. Chien, "A partial MILP algorithm for the design of linear phase FIR filters with SPT coefficients," *IEICE Trans. Fundamentals*, vol. E85-A, pp. 2302–2310, Oct. 2002.

[14] O. Gustafsson and L. Wanhammar, “Design of linear-phase FIR filters combining subexpression sharing with MILP,” in *Proc. IEEE 45th Midwest Symp. Circuits Syst.*, Tulsa, OK, Aug. 2002, vol. 3, pp. 9–12.

[15] D. M. Kodek, “Design of optimal finite wordlength FIR digital filters using integer programming techniques,” *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-28, no. 3, pp. 304–308, Jun. 1980.

[16] Y. C. Lim and S. R. Parker, “FIR filter design over a discrete powers-of-two coefficient space,” *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-31, no. 3, pp. 583–591, Jun. 1983.

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

[18] 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.

[19] QSOPT Linear Programming Solver. Aug. 2007 [Online]. Available: http://www2.isye.gatech.edu/~wcook/qsopt/

[20] ILOG CPLEX. Aug. 2007 [Online]. Available: http://www.ilog.com/products/cplex/

Mustafa Aktan received the B.S., M.S., and Ph.D. degrees in electrical and electronics engineering from Bogaziçi University, Istanbul, Turkey, in 1999, 2001, and 2007, respectively.

His research interests include VLSI design automation and low-power digital circuit design.

Arda Yurdakul received the B.S., M.S., and Ph.D. degrees in electrical and electronics engineering (with honors) from Bogaziçi University, Istanbul, Turkey.

Between 1999 and 2001, she was an Assistant Professor with Kadir Has University, Turkey. Since 2001, she has been with the Computer Engineering Department, Bogaziçi University, where she is currently an Associate Professor. She is the founder and director of the Computer Architecture and Systems Laboratory (CASLAB) in the same department. Her research interests include VLSI design automation, embedded systems and reconfigurable computing.

Dr. Yurdakul was the recipient of the Career Award from the Scientific and Technological Research Council of Turkey in 2005.

Gunhan Dundar (M’95) was born in Istanbul, Turkey, in 1969. He received the B.S. and M.S. degrees from Bogazici University, Istanbul, Turkey, in 1989 and 1991, respectively, and the Ph.D. degree from Rensselaer Polytechnic Institute, Troy, NY, in 1993, all in electrical engineering.

Since 1994, he has been with the Electrical and Electronic Engineering Department, Bogazici University, where he is currently a full Professor and the Department Chairman. Between 1994 and 1995, he was with the Turkish Naval Academy on military service and, between 2000 and 2001, he was with EPFL, Switzerland, on sabbatical leave. His research interests include analog integrated circuit design, design automation of analog systems, and power optimization.