Abstract— A novel algorithm for designing low power and hardware efficient linear phase 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 (BAB) 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 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 less 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 sub-expression 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, max. stop-band attenuation, etc.); 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 has been attempts to combine the two approaches, where a set of coefficients with reduced number of ones is found and then sub-expression 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 is 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 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 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 rest 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 based search. By 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 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 rest of the paper is organized as follows: Section 2 describes the proposed algorithm, FIRGAM. Section 3 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 4 where a comparison to other existing methods is also given. Section 5 concludes the discussion.

II. THE FIRGAM ALGORITHM

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

$$|H(\omega) - H_d(\omega)| \leq \delta(\omega).$$  (1)

Obviously, $H(\omega) = \sum_{m=0}^{M-1} h[m]T(\omega)$ 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 a FIR filter and symmetry halves the number of multiplications, 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

$$M = \left\lfloor \frac{N + 1}{2} \right\rfloor. $$  (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 $h[i]$ ($i=0,1,..,M-1$) can be determined by solving the following set of linear optimization problems independently:

$$h_{\text{min}}[i] = \min h[i]$$

such that

$$\sum_{m=0}^{M-1} h[m]T(\omega) - H_d(\omega) \leq \delta(\omega), \omega \in [0, \pi)$$

$h[m]:$ free, $0 \leq m < M$  (3)

$$h_{\text{max}}[i] = \max h[i]$$

such that

$$\sum_{m=0}^{M-1} h[m]T(\omega) - H_d(\omega) \leq \delta(\omega), \omega \in [0, \pi)$$

$h[m]:$ free, $0 \leq m < M$  (4)

Coefficient $h[i]$ is a number that appears in this range, i.e. $h[i] \in [h_{\text{min}}[i], h_{\text{max}}[i]]$. 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 $P \ll B/2$ 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 $\bigcup_{m=0}^{M-1} V_m$ must
satisfy (1).

Experimentally, it has been observed that $V_0$ is the smallest subset of $\bigcup_{m=0}^{M-i} V_m$ 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 branch-and-bound algorithm that runs on $\bigcup_{m=0}^{M-i} V_m$ will obviously find the optimal solution provided that enough memory and time is given. It is easy to observe that there exist $\prod_{m=0}^{M-i} |V_m|$ 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 pseudo-code is given in Fig. 1, is a modified branch-and-bound 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 $V^*_i$ 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:

$$h_{\min}[i] = \min h[i]$$

such that

$$\sum_{m=0}^{M-i} h[m] T(\omega) - H_\delta(\omega) \leq \delta(\omega), \omega \in [0, \pi]$$

$$h[m]:\begin{cases}
\text{fixed, } & 0 \leq m < i \\
\text{free, } & i \leq m < M
\end{cases}$$

$$h_{\max}[i] = \max h[i]$$

such that

$$\sum_{m=0}^{M-i} h[m] T(\omega) - H_\delta(\omega) \leq \delta(\omega), \omega \in [0, \pi],$$

$$h[m]:\begin{cases}
\text{fixed, } & 0 \leq m < i \\
\text{free, } & i \leq m < M
\end{cases}$$

It should be noted that these equations are different from equations (3) and (4) so as to refine the search space for the $i$'th tap after fixing $i - 1$ taps. The superscript $s$ in these equations and in the pseudo-code stands for the refined variables. Obviously fixing the values of $i - 1$ taps will move the boundary values of the $i$'th tap towards each other and the number of possible values for this tap will be reduced. If the refined value set ($V^*_i$) of $h[i]$ is not empty, then the iteration goes on as usual, but mid-value $h_{\min}[i]$ 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 $V^*_i$. 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. Yet, 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 branch and bound based algorithms on enumeration based algorithms is that they employ a cut-off 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 cut-off 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 $F_i + p + U_i \geq O$ 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.

- **$U_{\min}$:** $V_i$ 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 $P_{i,m}^{\min}$ 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
FIRGAM($M, L, H_d(\omega), \bigcup_{m=1}^{M-1} V_m$) 
\begin{align*}
H' &= \emptyset; \\
O &= \infty; \\
\text{Obtain } h_{\text{min}}'[0] \text{ and } h_{\text{max}}'[0] \text{ by solving equations (5) and (6);} \\
V'_0 &= \text{VALUE_SELECT}(L, h_{\text{min}}'[0], h_{\text{max}}'[0], V_0); \\
i &= 0; \\
\text{while } (i \geq 0) \{ \\
\text{if } (V' \neq \emptyset) \{ \\
& h[i] = v' \text{ such that } v' \text{ is the first element of ordered set } V'; \\
& H' = H' \cup \{h[i]\}; \\
& V'_i = V'_i - \{v'\}; \\
& \text{if}(\text{there can be a solution having SPT terms less than } O) \{ \\
& \text{if } (i < M - 1) \{ \\
& \text{Obtain } h_{\text{min}}'[i] \text{ and } h_{\text{max}}'[i] \text{ by solving equations (5) and (6);} \\
& V'_i = \text{VALUE_SELECT}(L, h_{\text{min}}'[i], h_{\text{max}}'[i], V_i); \\
& \text{else if } (\text{problem is feasible}) \{ \\
& H' = \text{H' is a solution to the problem;} \\
& O = \text{number of SPT terms of } H'; \\
& \text{else} \{ \\
& H' = H' - \{h[i]\}; \\
i &= i - 1; \\
& \} \}
\}} \}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
TABLE 1
FILTER CHARACTERISTICS OF THE FIRGAM EXAMPLE
<table>
<thead>
<tr>
<th></th>
<th>( \delta_p )</th>
<th>( \delta_s )</th>
<th>( f_p )</th>
<th>( f_s )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum passband ripple</td>
<td>0.05</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Normalized passband edge frequency</td>
<td>0.10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maximum stopband ripple</td>
<td>0.05</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Normalized stopband edge frequency</td>
<td>0.25</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TABLE 2
VALUE SETS OF THE COEFFICIENTS FOR THE FIRGAM EXAMPLE

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

\[V_i\] of coefficient \( h[i] \), \( L \) values in the vicinity of \( h'_{\text{med}}[i] \) are selected. If \( L \) is set to infinity all values in \( V_i \) 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, \( h'_{\text{med}}[i] \). 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 \( V_i \) to fix coefficient \( h[i] \), one can use different strategies. The strategy adopted in FIRGAM is selecting the value closest to \( h'_{\text{med}}[i] \) 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 \( V_i \) by calling the function VALUE_SELECT in Fig. 1, and then reordering the values in \( V_i \) 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.

B. Run-time of FIRGAM

The exponential run-time \( (O(L^M)) \) of the algorithm is due to the branch and bound 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 10 taps is to be designed, i.e. \( N = 10, M = 5 \). The coefficients are quantized to 7 bits \( (B = 7) \). The desired filter characteristics are given in Table 1. The boundary values of the coefficients are found using (3) and (4). The value sets \( (\bigcup_{m=0}^{M-1} V_m) \) formed using the boundary values are listed in Table 2.

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 \( h'_{\text{med}}[0] = -3.62 \) and \( h'_{\text{med}}[0] = -0.83 \) from which the middle value is found as \( h'_{\text{med}}[0] = -2.24 \). 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 \( h'_{\text{med}}[0] \) are selected, which makes \( V_0' = \{-2,-3\} \). The value closest to \( h'_{\text{med}}[0] = -2.24 \) in \( V_0' = \{-2,-3\} \) is \(-2\), so \( h[0] = -2, H' = \{-2\} \) and \(-2\) is removed from the refined value set making \( V_0'' = \{-3\} \). The algorithm branches to the next coefficient, \( h[1] \). The refined boundary values are found to be \( h'_{\text{min}}[1] = -4.38 \) and \( h'_{\text{max}}[1] = -1.26 \) making \( h'_{\text{med}}[1] = -2.82 \). The refined value set is \( V_1' = \{-3,-2\} \). \( h[1] \) is set to \(-3\) making \( H' = \{-2,-3\} \) and \( V_1'' = \{-2\} \). The algorithm branches to the next coefficient, \( h[2] \). The search goes on until all values in \( V_i \) are tried out.

The values of the variables at each iteration during the whole search process are given in Table 3. 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.

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 which the corresponding coefficients are fixed to.

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 \(1 \text{ multiplier} + 1 \text{ adder}\) 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 \(h[i]\) to be \(P\), the depth of the adder tree used to replace the multiplier will be at most \(P–1\) adders. If the maximum number of SPT terms per coefficient is \(P_{\text{max}}\), then the contribution of a replaced multiplier to the critical path will be at most \(P_{\text{max}}–1\) adders.

Then for the transposed form realization of an FIR filter the

\[
\begin{array}{c|c|c|c|c|c|c|c|c|c}
\text{i} & h_{\text{min}}[i] & h_{\text{max}}[i] & h_{\text{mid}}[i] & V_{\text{r}}[i] & h[i] & H[i] & O & \text{Status} \\
\hline
0 & -3.62 & -0.83 & -2.24 & (-2,3) & 2 & -2 & \infty & \\
1 & -4.38 & -1.26 & -2.82 & (-3,2) & 2 & -2 & & \\
2 & 1.01 & 3.57 & 2.29 & (2,3) & 3 & 2 & & \\
4 & 21.18 & 22.26 & 21.72 & (22) & 22 & [-2,-3,2,13,22] & 10 & S \\
3 & & & & (13) & 2 & [-2,3] & & \\
2 & & & & (3) & 3 & [-2,3,3] & & \\
1 & & & & (13) & 2 & [-2,3,3,12] & & \\
2 & 1.75 & 3.99 & 2.87 & (3,2) & 2 & [-2,2,3] & & \\
3 & & & & (13) & 2 & [-2,2,3,21] & & \\
2 & & & & (2) & 2 & [-2,2,2,21] & & \\
1 & & & & (13) & 2 & [-2,2,2,21] & & \\
0 & & & & (21) & 21 & [-2,-2,2,21,21] & & \\
1 & -3.62 & -1.28 & -2.45 & (-2,3) & 3 & [-2,-3] & & \\
2 & 2.65 & 4.01 & 3.33 & (3,4) & 3 & [-2,3,2,3] & C & \\
2 & & & & (13) & 3 & [-2,2,3,2] & & \\
1 & & & & (3) & 3 & [-3,2,3,2] & C & \\
0 & & & & (1) & 3 & [-2,3,3] & C & \\
\end{array}
\]

The iterative solution of FIRGAM example.

The table above shows the iterative solution of the FIRGAM example. The values in the table represent the coefficients and their associated SPT terms. The Status column indicates whether the solution is S, C, or I, where S indicates a stable solution, C indicates a critical path, and I indicates an iterative solution.
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-bit ripple carry adder will consist of 8 full adders, and hence will have an 8 full adder long critical path for the carry signal to travel to the most significant output sum. For a FIR filter, the maximum size of an adder, which is at the same time the output wordlength, can be calculated by the following formula:

\[
W_{\max} = W_{in} + \log_2 \left( \sum_{n=0}^{N-1} |h[n]| 2^{N-1} \right) + B - 1
\]

(11)

where \( W_{in} \) is the input data wordlength (including sign bit), \( h[n] \) filter coefficients \((-1 \leq h[n] < 1)\), \( N \) number of coefficients in the filter, and \( B \) wordlength of the coefficients (including sign bit). \( W_{\max} \) is the minimum adder size that ensures 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 a FIR filter having at most \( P_{\max} \) SPT terms in a coefficient will be

\[
CP = W_{\max} + P_{\max} - 1
\]

(12)

full adders long. This is also shown in Figure 4. From (12) it is clear that the critical path does not depend only on the maximum number of SPT terms \( P_{\max} \) 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 minimum, i.e. the depth is \( \left\lceil \log_2 P_{\max} \right\rceil \) adders, reducing the wordlength is even a more effective way (linear vs. logarithmic dependence).

**IV. EXPERIMENTAL RESULTS**

In this section several example filters are designed using FIRGAM and compared to other methods from the literature. The compared methods are the Remez algorithm, trellis algorithm [12], Li’s algorithm [11], Lim’s algorithm [18], Samueli’s algorithm [10], Yao’s algorithm [13], 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 PMILP are taken directly from the papers they appeared. The algorithms TRE, LI, and FIRGAM are implemented in the C programming language. FIRGAM uses the linear programming (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 2.8GHz processor and 1 GB RAM. The hardware realizations of the filters are done using the transposed form

![Fig. 3. Transposed form realization of an FIR filter.](image)

![Fig. 4. Critical path in a FIR filter.](image)
structure. For all filters the subexpression sharing method in [1] is used. The input data width of the filters is taken as 8 bits. 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 computed so that no overflow occurs. The reduction in power consumption.

Example 1: The desired frequency response characteristics of the filters are given in Table 4. $\delta_p$ is the passband peak-to-peak ripple, $\delta_s$ 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 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 5. 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], $P$ 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 $\mu$W/MHz. The SPT term and power gains are calculated taking the Remez (RMZ) filters as reference. As an example, the frequency response and coefficients of FIRGAM filter L1 are given in Fig. 5 and Table 7 respectively.

Looking at the results in Table 5, 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 to the other methods. The reduced SPT term count and wordlength has reduced the hardware cost and power consumption. Again looking at Table 5, 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 6. 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

\begin{table}[h]
\centering
\caption{Frequency response characteristics of the filters of Example 1}
\begin{tabular}{|c|c|c|c|c|c|c|}
\hline
Filter & Type & $\delta_p$ (dB) & $f_s$ & $\delta_s$ (dB) & $f_p$ \\
\hline
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 \\
\hline
\end{tabular}
\end{table}

\begin{table}[h]
\centering
\caption{Properties of designed filters of Example 1}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|}
\hline
Filter & Method & $N$ & $B$ & $B_e$ & $P$ & SPT Term & # of SPT & Gain & # of adders & Hardware complexity & Power & Gain \\
& & & & & & # & Time & (%) & MA & SA & Total & # of FA & # of DFF & Total & $\mu$W/MHz & Gain \\
\hline
L1 & RMZ & 121 & 19 & 16 & 8 & 483 & 78 & 120 & 198 & 3647 & 2907 & 6554 & 1090 & - \\
& LIM & 121 & 17 & 14 & 7 & 416 & 61 & 120 & 181 & 3146 & 2664 & 5810 & 938 & 13.9 \\
& PMILP & 121 & - & 15 & 6 & 442 & 59 & 120 & 179 & 3248 & 2748 & 5996 & 988 & 9.4 \\
& FIRGAM & 121 & 17 & 14 & 6 & 362 & 47 & 120 & 167 & 2886 & 2654 & 5540 & 864 & 20.7 \\
\hline
L2 & RMZ & 63 & 15 & 12 & 5 & 197 & 30 & 62 & 92 & 1510 & 1281 & 2791 & 422 & - \\
& LIM & 63 & 13 & 10 & 4 & 159 & 23 & 62 & 85 & 1311 & 1154 & 2465 & 349 & 17.3 \\
& PMILP & 63 & - & 11 & 4 & 163 & 19 & 62 & 81 & 1285 & 1182 & 2467 & 344 & 18.4 \\
& FIRGAM & 63 & 13 & 10 & 4 & 140 & 18 & 62 & 80 & 1213 & 1154 & 2367 & 328 & 22.3 \\
\hline
S2 & RMZ & 60 & 16 & 12 & 6 & 204 & 32 & 59 & 91 & 1494 & 1251 & 2745 & 421 & - \\
& SAM & 60 & - & 13 & 4 & 174 & 32 & 59 & 91 & 1455 & 1298 & 2753 & 402 & 4.5 \\
& PMILP & 60 & - & 12 & 4 & 174 & 25 & 59 & 84 & 1342 & 1255 & 2597 & 368 & 12.6 \\
& FIRGAM & 60 & 15 & 11 & 5 & 160 & 27 & 59 & 86 & 1337 & 1180 & 2517 & 362 & 14.0 \\
\hline
\end{tabular}
\end{table}
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 8. $N_{\text{min}}$ is the minimum number of taps required to realize each filter.

The properties of the designed filters are given in Table 9. The run times of the algorithms are listed in Table 10. 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. 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 10 is the result of multiple runs of the algorithm. The algorithms are not allowed to run more than 24 hours. 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 obtained by FIRGAM compensates 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 hours 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 1 day.

**Example 3:** For the same frequency response characteristics of the filters of example 2, filters are re-designed with different number 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 11. 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 by the reduction of the wordlength of the coefficients, which is clearly shown in Table 11.

**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 to existing methods with several examples. The filters found by the proposed algorithm have less 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


