Bit-Level Multplierless FIR Filter Optimization Incorporating Sparse Filter Technique

Wen Bin Ye, *Student Member, IEEE* and Ya Jun Yu, *Senior Member, IEEE*

***Abstract*— Multiplierless FIR filter optimization has been extensively studied in the past decades to minimize the number of adders. A more accurate measurement of the implementation complexity is the number of full adders counted at bit-level. However, the high computational complexity of the optimization at bit-level hinders the technique from practical applications. In this paper, the sparse filter technique is exploited and makes the search space at bit-level significantly reduced. Thus, the bit-level optimization of multiplierless FIR filters for the first time becomes possible. When the sparse filter technique is employed for the multiplierless filter design, the sparsity of the filter is properly controlled so that the feasibility of the bit-level optimization in discrete space is maintained. Thereafter, in the reduced search space, a tree search algorithm is formulated at bit-level, and techniques to estimate the bit level hardware cost and to accelerate the search are presented. Design examples show that the proposed bit-level optimization method generates designs with lower hardware cost and power consumption than that of the best word-level optimization methods, while the design time is still at an acceptable level. The average power savings to 3 recent published techniques are 13.6% , 8.0% and 26.1%, respectively.**

***Index Terms*— finite impulse response (FIR), sparsity, bit-level optimization, multiplierless.**

I. Introduction

Finite impulse response (FIR) filters have many applications in the digital processing systems for their guaranteed stability and linear phase. However, compared with their counterpart infinite impulse response filters, the hardware cost of FIR filters is much higher due to the large number of multiplications. Basically, there are two categories of techniques to design low hardware cost and low power consumption FIR filters. One is sparse FIR filters [1]–[10], and the other is multiplierless FIR filters. Sparse FIR filters, according to its name, have fewer number, or in other words, sparser non-zero coefficients than conventional direct form minimum order designs. Fewer non-zero coefficients lead to fewer multiplications, lower hardware cost and lower power consumption.

On the other hand, multiplierless FIR filters replace the coefficient multipliers with a network of adders and shifts, resulting in reduced hardware cost and power consumption as well. Many algorithms [11]–[33] have been proposed to minimize the number of adders used to synthesize the adder-and-shift network. While algorithms in [11]–[24], named as MCM algorithms, are applied to a set of discrete coefficients which already meets a given filter specification, more algorithms recently proposed, named as MCM and filter coefficient joint optimization algorihtms, incorporate the synthesis of coefficients into the optimization search of discrete coefficients for a given filter specification [25]–[33].

The algorithms [25]–[33] in the later group generally produce designs with lower hardware cost and power consumption. However, these algorithms are all aiming to optimize the filter at word-level, i.e, to minimize the number of adders, regardless of the wordlength of adders. However, fewer number of word-level adders is not necessary leading to fewer number of full adders (FAs) — a more accurate indication of hardware cost and power consumption — since the width of each adder may differ significantly. The idea to optimize the filter at bit-level is always attractive and many MCM algorithms [12], [22]–[24] for bit-level design have already been proposed. However, the development of MCM and filter coefficient joint optimization algorithms in bit-level is hindered by the practical high computational complexity to obtain a (sub)optimal design. If the word-level joint optimization techniques are directly mapped to bit-level, it is highly possible that in the design of a filter with order around 60, within the same acceptable time, the bit-level optimization results may be worse than the word-level design in terms of number of FAs due to the significantly reduced coverage of the entire search space.

The critical issue to successfully optimize filters in bit-level is to reduce the search space but without significant sacrifice in the optimality of the final solution. The sparse filter technique reveals that the filter specification may be satisfied when a few coefficients are constrained to zero for a given filter order. This technique can be used in the bit-level optimization of multiplierless FIR filters to reduce significantly the search space. Setting several coefficients values to zero reduces not only the number of coefficients to be optimized, but also the feasible range of the rest non-zero coefficients.

While the sparse filter design pursues the most sparse solution, when it is used for multiplierless FIR filter design, the filter sparsity must be constrained such that the order of the filter is not increased, and the rest non-zero coefficients still have sufficient room to be optimized in discrete spaces.

In this paper, first, a proper control of sparsity tailored for the bit-level optimization of FIR filters is proposed. Thereafter, a ripple-carry-adder (RCA) based bit-level optimization problem is formulated as a tree search within the significantly reduced search space. The cost estimation of a tree node and cut-off scheme, which play vital roles for an efficient search, are presented. Due to the novel approaches to estimate the cost and cutting off the tree, the structure adders(SAs) are

Manuscript received on Dec. 15, 2013 and revised on Apr. 13, 2014. W. B. Ye and Y. J. Yu are with the school of Electrical and Electronic Engineering, Nanyang Technological University, Singapore (e-mail:yewe0003@e.ntu.edu.sg; eleyuyj@pmail.ntu.edu.sg)

also optimized, whereas previous algorithms only optimize multiplier block adders(MBAs). The definitions of SAs and MBAs are given in Section II-A.

The rest of the paper is organized as follows. In Section II, the sparsity control technique is discussed. Section III presents the algorithm for the optimization of FIR filters at bit-level. Design examples are given in Section IV to show the advantages of the proposed algorithm. Some discussions related to the design speed and power consumption of the FIR filters are presented in Section V. Finally, the paper is concluded in Section VI.

II. Sparsity Control For the Bit-level Optimization of FIR Filters

In this section, first, the motivation and concerns in using sparse filter technique for the multiplierless filter optimization are presented. A sparsity control technique is then proposed. With the controlled sparsity, a sparse filter can thus be optimized.

A. Motivation and Concerns

As mention in the Introduction, by setting several coefficients to zero, not only the number of coefficients to be optimized is reduced, but also the feasible range for the remaining non-zero coefficients are shrunk. Furthermore, in multiplierless filter design, researches have shown that SAs, the adders summing up the delayed weighted signal in the transposed direct form structure, account for much more areas than MBAs [34]. It is highly motivated to fix more coefficients values to zero to remove more SAs.

Traditional sparse filter design tried to fix as many coefficients to zeros as possible, such that the filter ripple requirements are marginally satisfied. For the zero-phase frequency response of a liner phase FIR filter of order N, expressed as

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

the optimization of the sparse FIR filter is to minimize the number of non-zero coefficients $h(n)$ for $n = 0, 1, ..., N$, subject to the filter specification. In (1), $\mathrm{Trig}(\omega, n)$ is an appropriate trigonometric function depending on the parity of N and symmetry of the filter [30]. Let $\mathbf{h}$ represent the unknown half of the symmetric coefficient vector $[h(0), h(1), ..., h(\lfloor \frac{N}{2} \rfloor)]^T$. Ideally, traditional sparse filter design problem is formulated as an l_0-norm problem, expressed as:

$$\begin{array}{lll} \text{min:} & ||\mathbf{h}||_0 & \\ \text{s. t.:} & 1-\delta_p \le H(\omega) \le 1+\delta_p, & \text{for } \omega \in [0, \omega_p] \\ & -\delta_s \le H(\omega) \le \delta_s, & \text{for } \omega \in [\omega_s, \pi], \end{array} \tag{2}$$

where δ_p, δ_s, ω_p and ω_s are the given passband ripple, stopband ripple, passband edge and stopband edge, respectively.

However, in the multiplierless FIR filter design, the sparsity of the filter must be controlled, such that the ripple deterioration due to the fixing of zero coefficients still leave enough space for the optimization of the remaining non-zero coefficients. Therefore, the design of sparse filter for the multiplierless filter is a trade-off of the sparsity and the filter ripple margins. In an earlier work [35] in the optimization of multiplierless variable fractional delay filters, the idea to fix some coefficients to be zero has been implemented. In this section, a quantitative criteria is derived to trade-off the sparsity and the filter ripple margins.

B. The Sparsity Control

Due to the consideration in Section II-A, the sparsity of the filters is controlled by setting a more stringent passband ripple and stopband ripple than δ_p and δ_s given in (2). Denoting the new passband ripple as δ'_p, the stopband ripple is correspondingly adjusted to $\delta'_s = \frac{\delta_s \delta'_p}{\delta_p}$. In other words, through setting the new ripple specification for the sparse filter design, the ripple deterioration due to fixing coefficients to zeros will be restricted to a reasonable level. Thus, the sparsity control problem is transformed to find a proper δ'_p.

Let δ_{opt} be the minimum passband ripple that can be achieved by the filter of order N with continuous coefficients. The value of δ_{opt} can be obtained by

$$\begin{array}{lll} \text{min: } \delta_{opt} & & \\ \text{s. t: } 1-\delta_{opt} \le H(\omega) \le 1+\delta_{opt}, & & \text{for } \omega \in [0, \omega_p] \\ \quad -\delta_s\delta_{opt}/\delta_p \le H(\omega) \le \delta_s\delta_{opt}/\delta_p, & & \text{for } \omega \in [\omega_s, \pi]. \end{array} \tag{3}$$

Obviously, δ'_p should be set to a value between δ_p and δ_{opt}. Here, a weighting factor α with a value between 0 and 1 is introduced to express δ'_p as $\alpha\delta_p + (1-\alpha)\delta_{opt}$, and α has to be determined. In an extreme case that the coefficient effective wordlength (EWL) is very large, the coefficient values are nearly continuous and α can be simply set to 1. In this paper, the EWL of a coefficient refers to the wordlength excluding the sign bit and the leading zero bits of the coefficient value; and the EWL of a filter refers to the EWL of the coefficient with the maximum magnitude in the filter. Generally, in the multiplierless FIR filter design, the EWL should be set to the minimum value because every increment of the coefficient wordlength may increase the bit-level hardware cost significantly. The minimum EWL refers to the wordlength below which no discrete solution can be found. Under such wordlength constraints, a larger α will result in more zero coefficients, and this may further lead to nonexistence of feasible discrete solution when the remaining non-zero coefficients are optimized, although it may reduce the search space significantly. The value of α is affected by many factors such as the filter order, EWL, ripple requirement, etc., and is difficult to be determined theoretically.

According to our design experience, the range of α may vary from 0.0124 to 0.927, i.e., for different filters, the value of α may differ significantly. In order to dynamically adapt the value of α to the change of filter specifications, the following procedure is used.

- Step 1. Set l to 0 and α_l to a conservative value which is 0.1 in our case.

- Step 2. For the value α_l, determine the number of coefficients that can be set to zero. Denote the number as N^l_{zero}. The methods on the determination of the value N^l_{zero} and the determination of the indexes of the N^l_{zero} zero coefficients are discussed in the next sub-section. If $N^l_{zero} = 0$, stop, i.e, no coefficient can be set to 0 for this filter specification and α is set to α_l, i.e., 0.1.
- Step 3. If $l > 1$ and $N^l_{zero} = N^{l-1}_{zero}$, stop and α is set to α_{l-1}; otherwise, increase l by 1, set $\alpha_l = \alpha_{l-1} + \alpha_{l-1}/N^{l-1}_{zero}$, and go to step 2.

In step 3, the stepsize $\alpha_{l-1}/N^{l-1}_{zero}$ reflects the average weight ripple deterioration due to a fixed 0. Through the above procedure, when α_l is increased by such a way, but the number of zero-coefficients is not increased further, α will stop increasing, i.e., the cost of ripple deterioration is too high to increase α further.

C. Determine the Indexes of Zero Coefficients

With the adjusted ripple requirements presented in Section II-B, the sparse filter optimization in (2) is revised by replacing δ_p and δ_s with δ'_p and δ'_s. The l_0-norm problem in (2) is non-convex. Many algorithms have been proposed for the optimization of the problem, including the global optimal solutions [5], [6] and trade-offs between the optimality and computation time [7]–[10].

When the sparsity is used for multiplierless filter designs, the filter order is usually close to the minimum order of the continuous coefficient filter. Thus the number of coefficients which can be fixed to 0 is relatively small. Therefore, all the above design techniques tend to converge to the global optimal solution within acceptable computation time. Taking the filter Y1 from [30] as an example, all the sparse techniques [5]–[10] produces the design with the same 3 zero coefficients. Among these algorithms, the algorithm [8] using the smallest-coefficient rule is the simplest one. It optimizes the filter coefficients to minimize the filter ripples using linear programming. The coefficient with the smallest magnitude is fixed to zero. The remaining coefficients are reoptimized and the smallest one is again fixed to zero until the ripple requirement is violated. This algorithm is used in this paper to locate the zero coefficients for the obtained δ'_p and δ'_s.

III. Optimization of FIR Filters at Bit-level by Exploiting Sparsity

By applying the sparse filter technique in section II, some coefficients for a given filter specification have been fixed to 0. In this section, first, the RCA based bit-level implementation of multiplierless FIR filters is briefly reviewed. After that, a mixed integer linear programming (MILP) tree search algorithm is introduced to optimize the rest non-zero coefficients at bit-level. In the optimization, the non-zero coefficients are selected to be quantized to discrete values and further synthesized to an adder-and-shift network. New cut-off scheme and bit-level hardware cost estimation are developed for the tree search.

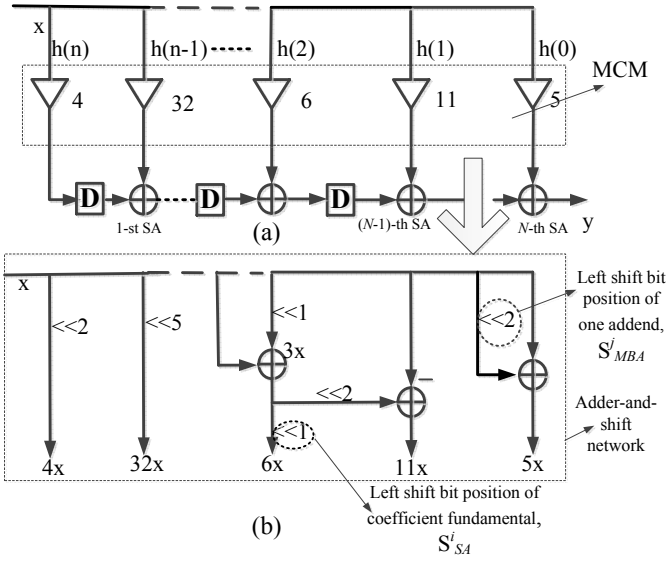

Fig. 1. Diagram of FIR filters with coefficients synthesized as a network of adders and shifts.

A. Review of RCA based bit-level implementation of mutliplierless FIR Filters

Transposed direct forms can be used to synthesize multiplierless FIR filters efficiently as shown in Fig. 1 (a). In this structure, the input signal is concurrently multiplied by filter coefficient values, and the resultant signals are then summed through a delay chain. The part consisting of all discrete coefficient multipliers is named as multiple constant multiplication (MCM) block and can be replaced with a network of adders and shifts as shown in Fig. 1 (b) [36]. Such multiplierless realization maximally increases the filter operation speed and reduces the circuit complexity and power consumption. In the transposed direct form, the two types of adders that are in the MCM block and in the delay chain, are named as MBA and SA, respectively, which have been mentioned in Section II-A.

In the adder-and-shift network in Fig. 1 (b), the intermediate or final odd scale factors of the signal x generated by shifts and additions, such as 3, 11 and 5 are referred to as fundamentals or bases. In this paper, the fundamentals are denoted as b_j. There are two types of fundamentals. One is coefficient fundamental, i.e. coefficient values can be obtained by the left-shift operation of such fundamentals; the other is the intermediate fundamentals which are used to synthesize the coefficient fundamentals. For example, in Fig. 1 (b), 11 is the coefficient fundamental to realize the coefficient value of $h(1)$ while 3 is the intermediate fundamental to synthesize 11. From Fig. 1 (b), it can be seen that some coefficient fundamentals may also serve as intermediate fundamental of other coefficients, for example, 3 is the coefficient fundamental of $h(2)$ but also serves as an intermediate fundamental of $h(1)$.

In the following, the bit-level implementation of adders are reviewed. Only are RCA based bit-level adders considered in this paper because of its regular structures and massive available studies of synthesis techniques [11]–[17].

If the "removing half adder technique" proposed in [23] is adopted, the number of full adders to implement the j-th

fundamental b_j, denoted as NFA^j_{FUN}, is expressed as [22], [23]

$$NFA^j_{FUN} = \lceil \log_2(b_j) \rceil + B_{in} - S^j_{MBA}, \tag{4}$$

where B_{in} is the wordlength of the input signal, S^j_{MBA} is the left shift bit position of an addend of the j-th MBA adder as shown in Fig. 1 (b). Note that in the two addends of any MBA adders, only one addend is left shifted. It should be noted further that when the synthesis of fundamental b_j is not certain, i.e., if extra intermediate fundamental is required , the term S^j_{MBA} is omitted, because S^j_{MBA} refers to the shift position of the extra fundamental.

On the other hand, denoting NFA^i_{SA} as the number of FAs used for the i-th SA (for $0 < i \leq N$) which adds $h(N-i)x$ into the tap delay line, NFA^i_{SA} is expressed as

$$NFA^i_{SA} = \left\lceil \log_2 \sum_{k=0}^{i} h(N-k) \right\rceil + B_{in} - S^i_{SA}; \tag{5}$$

where S^i_{SA} is the left shift bit position of the coefficient fundamental to realize the coefficient $h(N-i)$ as shown in Fig. 1 (b). It should be noted that there is no SA for $i = 0$, or in other words, $NFA^0_{SA} = 0$. An example to compute NFA^i_{SA} for $i = 1$ in Fig. 1 is given as $\lceil \log_2 (4+32) \rceil + B_{in} - 1$. In the case of linear phase FIR filters, the coefficients are symmetrical and $h(N-i) = h(i)$. Hence, NFA^i_{SA} can also be expressed as

$$NFA^i_{SA} = \left\lceil \log_2 \sum_{k=0}^{i} h(k) \right\rceil + B_{in} - S^i_{SA}. \tag{6}$$

In this paper, linear phase FIR filters are considered, and (6) is used for the computation of NFA^i_{SA}. From (6), it can be seen that the number of FAs for different SAs increases with increasing i due to the term of $\log_2 \sum_{k=0}^{i} h(k)$. In the word-level optimization method, the hardware costs of the SAs are all assumed to be the same, such that the implementation cost of SAs are not optimized. In the proposed bit-level optimization, the hardware cost of the i-th SA is counted into the cost of the synthesis of the i-th coefficient. By considering the hardware cost of SAs, the bit-level optimization may results in significant hardware saving compared with the word-level optimization.

B. MILP Formulation for the FIR Filter Design

To find the feasible coefficient set $h(n)$ for the design of multiplierless FIR filter with floating passband gain, a linear programming is formulated to [30]

$$\begin{aligned} &\text{min: } f = \delta - \delta_p b \\ &\text{s. t.: } b - \delta \leq H(\omega) \leq b + \delta, && \text{for } w \in [0, \omega_p] \\ &\quad -(\delta_s \delta)/\delta_p \leq H(\omega) \leq (\delta_s \delta)/\delta_p, && \text{for } w \in [\omega_s, \pi] \\ &\quad h(n) = 0, && \text{for } n \in Z \\ &\quad b_l \leq b \leq b_u \end{aligned} \tag{7}$$

where b_l and b_u are two constants defining the lower bound and upper bound of the passband gain; in this paper, they are chosen to be 0.7 and 1.4, respectively. Z is the zero-index set obtained by the sparse technique proposed in Section II. In the formulation (7), the variables to be optimized are the unfixed coefficient set $h(n)$ for $n \notin Z$, ripple δ and the passband gain b. Frequency ω is discretized to $8N$ samples to form the constraints. The formulation can be used to obtain both continuous solution (i.e., all coefficients are continuous values) and partial continuous solution (i.e, some coefficients have been fixed to integer values).

Note that the objective function of the linear programming is not the normalized peak ripple (NPR) defined as δ/b [37]; instead, $\delta - \delta_p b$ is minimized. With such formulation, the algorithm may not find the optimum solution in the sense of minimum NPR, but it always manages to find a feasible solution making $\delta - \delta_p b < 0$, if such solution exists [30].

C. Tree Search Procedure

For a given filter EWL, the discrete value of each coefficient is maximally in the range $[-2^{EWL} \quad 2^{EWL}]$. Potential solutions are combinations of these discrete values. In the proposed tree search algorithm, the root of the tree is the optimum continuous solution of the FIR filter with $h(n) = 0$ for $n \in Z$. The root produces its child nodes by quantizing a selected coefficient to a discrete value. All child nodes will further produce their own child nodes by quantizing another coefficient to a discrete value. The process continues until it reaches a leaf which contains only discrete coefficients. The nodes having the same parent and generated by quantizing the same coefficient to different discrete values are called siblings. If a node cannot further produce a child node to forward the search for whatever reason, for example for the reason that all coefficients of the node have been quantized, the search will switch to its sibling; and if no more sibling exists, the search will track back to its parent node to continue the forward search. In such a manner, the program traverses all the nodes which have a possibility of yielding a feasible solution in a depth-first width-recursive manner, to find a solution with minimum number of FAs. The step by step search procedure is described as follows:

1) The root is generated by solving the linear programming formulated in (7). The feasible range of each coefficient is obtained as proposed in [30]. The root is the current node and the search depth d_s is set to 0. Let I be the coefficient index set excluding the indexes of zero coefficients. For example, if $h(2)$ and $h(4)$ are fixed to 0 in the sparse filter design, I is $\{0, 1, 3, 5, ..., \lfloor \frac{N}{2} \rfloor\}$.
2) A child of the current node is generated by quantizing the coefficient $h(I(d_s))$ to a new discrete value in its range found in step 1.
   - If the child is successfully generated, the generated child node becomes the current node, $d_s = d_s + 1$, and go to step 3.
   - If such child node does not exist, i.e., $h(I(d_s))$ has been fixed to its all possible discrete values, there are two cases: a) if $d_s > 0$, the search backtracks to the parent of the current node, $d_s = d_s - 1$ and the parent of the current node becomes the current node

and repeat step 2; b) if $d_s = 0$, the traverse of the tree is completed and the program is stopped.

3) The quantized discrete value is synthesized to an adder-and-shift network based on the subexpression basis set. The subexpression basis set of the current node is updated if new fundamentals are generated. A brief description of the subexpression basis set see Section III-D.
4) The first pruning checking: compute the FA cost of the realization of the current node and check whether this node or its descendent can grow to be the optimum solution (i.e. if the overall number of FAs up to the current node is smaller than that of the current best solution). If yes, go to step 5; otherwise, the current node is cut off and the search backtrack to its parent node which becomes the new current node, $d_s = d_s - 1$, and go to step 2. The details of computing the number of FAs for the realization of different type discrete coefficients are described in Section III-E.
5) Reoptimize the unquantized coefficients by solving (7) and obtain the objective value f meanwhile.
6) The second pruning checking: it is to check the feasibility of the node. If $f \leq 0$, the current node meets the ripple specification and go to step 7. Otherwise, the current node is cut off; the search backtracks to its parent node which becomes the new current node and $d_s = d_s - 1$, go to step 2.
7) If $d_s = \lfloor \frac{N}{2} \rfloor - N_{zero}$, where N_{zero} is the number of zero coefficients, all coefficients of the current node are discrete values, i.e., a leaf is reached. Go to step 8; otherwise, search forward by going to step 2.
8) If the leaf is better than the current best solution in terms of overall number of FAs, the best solution is updated.
9) The search backtracks to its parent and the parent node becomes the current node. Set $d_s = d_s - 1$ and go to step 2.

D. Dynamically Expanding Subexpression Space

The algorithm proposed in Section III-C is a tree search method, and each node of the tree represents a set of coefficient values which is a mixture of discrete and continuous coefficient values. The discrete coefficient values of the node are synthesized, and the synthesis is based on the subexpression space constructed from a subexpression basis set [27], [28], [30]. The elements of the subexpression basis set are the fundamentals that have been synthesized as an adder-and-shift network. Each node of the tree is associated with a subexpression basis set. The root of the tree has an initial zeroth order set including an element 1. All other nodes inherit a basis set from their parents, and the set is expanded by including the coefficient fundamental if the fundamental is not earlier included. If all coefficient fundamentals in the basis set can be synthesized without inserting extra intermediate fundamentals, the set is marked as "certain"; otherwise, the set is marked as "uncertain"; For example, if a node P_1 currently associates with a basis set {1,3}. Its child node P_2 inherits this basis set and use it to synthesize a new coefficient, say 43. The basis set of P_2 is expanded by adding 43. Since at least one intermediate fundamental is needed to synthesize 43 based on {1,3}, so the basis set P_2 is marked as "uncertain". After that, the child node of P_2, denoted as P_3, inherits the basis set {1,3,43} and uses it to synthesize a new coefficient, say 5. Obviously, no extra intermediate fundamental is needed to synthesize 5. Thus, the basis set of P_3 becomes {1,3,5,43}. With this new basis 5, the program tries to re-synthesize the coefficient 43, which makes the set "uncertain", and it turns out that 43 can be successfully synthesized as $43 = 5 \times 2^3 + 3$. Thus the "uncertain" of the set is removed. If the final basis set is still "uncertain", at that point, the MCM algorithm C1 [38] will be used to insert the necessary intermediate fundamentals. For more details of the subexpression space and its dynamic expansion, refer to [27], [28], [30].

E. Bit-level Cost Estimation for Coefficient synthesis

As introduced in the beginning of this section, the FAs used to construct the corresponding SA of a coefficient has to be counted into the cost of the coefficient when it is synthesized. It should be noted that for the linear phase FIR filter design, due to the symmetry of the coefficients, the coefficient $h(i)$ is not only connect to the i-th SA but also $(N-i)$-th SA, if i is not equal to $N-i$. When $h(i)$ is fixed, the number of FAs used in the i-th SA can be exactly computed, whereas that in the $(N-i)$-th SA cannot, since the values of unfixed coefficients $h(i+1)$ to $h(\lfloor \frac{N}{2} \rfloor)$ are required for the computation as shown in (6). For this reason, in the proposed algorithm, only the SAs whose index is less than $\lfloor \frac{N}{2} \rfloor$ are taken into consideration, i.e., the optimization objective function is the number of FAs of MBAs and the first half SAs, given by $\sum_j NFA^j_{MBA} + \sum_{k=0}^{\lfloor \frac{N}{2} \rfloor} NFA^k_{SA}$ for all fundamentals b_j. Though our design experiences show that this optimization strategy may lead to low hardware cost, the objective function is a suboptimal of the problem under consideration.

With these considerations, the cost to synthesize a fixed coefficient $h(i)$ (of which the corresponding coefficient fundamental is express as b_j) is computed as:

$$\begin{aligned} NFA^i_{fix} =& NFA^i_{MBA} + NFA^i_{SA} \\ =& NFA^j_{FUN} + NFA^i_{FUN,extra} + NFA^i_{SA} \end{aligned} \tag{8}$$

where NFA^i_{MBA} and NFA^i_{SA} are the number of FAs of MBAs used to synthesize the non-zero coefficient $h(i)$ and its corresponding i-th SA. The NFA_{MBA} can be further classified into two terms: one is the MBA used to realize the coefficient fundamental b_j, NFA^j_{FUN} given in (4), and the other is used to realize the extra intermediate fundamentals, denoted as $NFA^i_{FUN,extra}$ if any. Depending on the values of the discrete coefficients and how they are synthesized, NFA^i_{MBA} and NFA^i_{SA} are computed as follows:

Case 1: If the coefficient value $h(i)$ can be expressed as an existing fundamental scaled by a power-of-two term, then $NFA^i_{MBA} = 0$ and NFA^i_{SA} is computed according to (6).

Case 2: If the coefficient value $h(i)$ (of which the corresponding coefficient fundamental is express as b_j) is synthesized

TABLE I
AN EXAMPLE TO ILLUSTRATE THE PROPOSED COEFFICIENT SYNTHESIS PROCEDURE, THE UPDATE OF BASIS SETS, AND THE FAS COST OF EACH COEFFICIENT, WHERE NFA IS THE NUMBER OF FAS.

Coefficients	Main basis	NFAs	Total NFAs	Certainty
h(0)=11	{1,11}	$2B_{in}+4$	$2B_{in}+4$	N
h(1)=3	{1,3,11}	$2B_{in}+4$	$3B_{in}+6$	Y
h(2)=6	{1,3,11}	$B_{in}+4$	$4B_{in}+7$	Y

based on the basis set using one adder, i.e., no extra intermediate fundamentals are needed to synthesize this coefficient, then NFA^i_{FUN} is computed according to (4), $NFA^i_{FUN,extra}$ is 0 and NFA^i_{SA} is computed according to (6). The current basis set is updated to include the new coefficient fundamental. It should be noted that $NFA^i_{FUN,extra}$=0 does not necessarily mean that no intermediate fundamentals are required to synthesize this coefficient; it is possible that the current basis set before the update contains the required intermediate fundamentals for this coefficient.

Case 3: If the coefficient value $h(i)$ cannot be successfully synthesized based on the current basis set using not more than an adder, the basis set is marked as "uncertain" and other intermediate fundamentals are expected to be added into the set during the subsequent coefficient synthesis to restore the certainty of the set. In this case, at least an extra intermediate fundamental is required to synthesize the coefficient, and the minimum FA cost to realize the extra fundamental is B_{in}, i.e., $NFA^i_{FUN,extra}$=B_{in}. Thus, the FAs cost for the MBA is computed as $NFA^i_{MBA}=B_{in}+\lceil\log_2(b_j)\rceil+B_{in}$, where S^j_{MBA} in 4 is omitted because extra intermediate fundamental is required in this case. And NFA^i_{SA} is computed in the same way as that in Cases 1 and 2. Note that the basis set is updated to include the coefficient fundamental only.

An FIR filter with filter length 8 (i.e., 4 distinct coefficients) is given to illustrate the synthesis mechanism in details. As shown in Table I, first, $h(0)$ is fixed to 11 and it cannot be synthesized based on the initial zeroth order basis set using not more than an adder. The basis set is updated to include 11 and marked as "uncertain". The synthesis of 11 corresponds to Case 3. For $h(0)$, there is no SA to be implemented. Therefore, the total FA cost is $\lceil\log_2(11)\rceil+B_{in}+B_{in}$=$2B_{in}+4$. Thereafter, $h(1)$ is fixed to 3. Clearly, 3=$1\times 2^2-1$ can be realized based on the current basis set. After the synthesis, the basis set is updated to {1,3,11}. The FA cost for the coefficient $h(1)$ is of Case 2, and thus is equal to $\lceil\log_2(3)\rceil+B_{in}-2$ for the MBA, and $\lceil\log_2(11+3)\rceil+B_{in}-0$ for the SA; the total FA cost is $2B_{in}+4$. Furthermore, after adding 3 into the set, the program re-synthesizes 11 with the newly updated basis set. It turns out that 11 can be realized as $11=3\times 2^2-1$. Therefore, the uncertainty is removed. The FA cost for $h(0)$ is revised to $\lceil\log_2(11)\rceil+B_{in}-2$=$B_{in}+2$, and the total FA cost is $B_{in}+2$ plus $2B_{in}+4$, i.e., $3B_{in}+6$.

The last coefficient $h(2)$ can be realized as Case 1, i.e, $NFA^2_{MBA}=0$ and $NFA^2_{SA}=\lceil\log_2(11+3+6)\rceil+B_{in}-1$=$B_{in}+4$.

Overall, the total FA cost to synthesize the 8 tap filter is $4B_{in}+7$.

F. Cut-off Scheme for Bit-level Optimization

Good cut-off schemes play a key role in the MILP tree search. In the above tree search, two cut-off schemes are adopted to accelerate the search procedure. The second cut off mechanism in step 6 in Section III-C is the same as that proposed in [30], where the objective function, f, is obtained by solving MILP in (7) to determine if the node and its descendent remain feasible or not.

The first cut-off mechanism in step 4 is catered for the bit-level optimization, i.e., whether the total number of FAs of the node, denoted as NFA_{all}, has been larger than that of the best solution NFA_{best}. When all coefficients of the node have been quantized and synthesized, NFA_{all} is simply the sum of FAs of all coefficients. Denoting the number of NFAs used in all fix coefficients as NFA_{fix}, we have $NFA_{all}=NFA_{fix}$. However, in the search procedure, most nodes have partial fixed and partial unfixed coefficients. If NFA_{fix} is used to compare with NFA_{best} as the cut-off criteria, the criteria is too conservative to search efficiently. In this paper, a more efficient cut-off criteria is proposed that NFA_{all} is determined as

$$NFA_{all}=NFA_{fix}+NFA_{unfix} \tag{9}$$

where NFA_{unfix} is an estimation of the number of FAs contributed by the unfixed coefficients.

Assuming that there are N_{unfix} unfixed coefficients $h(i_k)$ for $1\le k\le N_{unfix}$, a very simple but conservative way to estimate NFA_{unfix} is given by

$$NFA_{unfix}=N_{unfix}B_{in}, \tag{10}$$

since even if all the unfix coefficients can be realized by using 0 MBA, the number of SAs required are N_{unfix}, and the minimum number of FAs for each SA is equal to the wordlength of the input signal. In order to improve the search efficiency, a more accurate estimation of the number of FAs to realize the N_{unfix} SAs is given by

$$\begin{aligned} NFA_{unfix} &= \sum_{k=1}^{N_{unfix}}\left\lceil\log_2\left(\sum_{j=1}^{k}|h(i_j)|+S_{fix}\right)\right\rceil \\ &+ N_{unfix}B_{in}-3N_{unfix} \end{aligned} \tag{11}$$

where S_{fix} is the sum of the absolute value of the fixed coefficients, and $h(i_j)$ is the i_j-th optimal continuous value of the unfix coefficients that are obtained in the tree search step 5. The estimation on the term NFA_{unfix} is based on the fact that the unfixed coefficients are quantized to the discrete values not significantly deviated from their optimal continuous value. When the continuous values are quantized, some are larger than their optimal continuous values while the others are smaller; they may compensate for each other, leading to the final estimation of the sum of the discrete values of the unfixed coefficients not significantly deviated from its actual value.

It should be noted that the last term $3N_{unfix}$ is used to estimate the total number of shifts for the basis to realize the coefficients, which is empirically determined, i.e, every

TABLE II
THE FILTER SPECIFICATIONS FOR G1,Y1,Y2,Y3,L2,S2,A AND B

Filters	Type	Filter order	ω_p	ω_s	δ_p	δ_s
G1	Low pass	15	0.2π	0.5π	0.01	0.01
Y1	Low pass	29	0.3π	0.5π	0.00316	0.00316
Y2	Low pass	37	0.3π	0.5π	0.001	0.001
Y3	Low pass	49	0.4π	0.51π	0.01	0.001
L2	Low pass	62	0.2π	0.28π	0.028	0.001
S2	Low pass	59	0.042π	0.14π	0.012	0.001
A	Low pass	58	0.125π	0.225π	0.01	0.001
B	Low pass	104	0.2π	0.24π	0.01	0.01

TABLE III
THE FILTER SPECIFICATIONS FOR L3

Bands (π)	δ_p^+ (dB)	δ_p^- (dB)	δ_s (dB)
$0 \sim 0.15$	0.2	-0.2	-
$0.15 \sim 0.1875$	0.2	-0.5	-
$0.1875 \sim 0.2125$	0.2	-0.9	-
$0.2875 \sim 1$	-	-	-30

unfixed coefficient is obtained by shift the coefficient fundamentals left by 3 bit positions in average.

IV. DESIGN EXAMPLES

In this section, 9 benchmark filters are given to illustrate the superiority of the proposed algorithm in terms of bit-level hardware cost, i.e., the number of FAs, and power consumption compared to the latest word-level optimization algorithms proposed in [30], [32], and [33]. These 3 algorithms are denoted as SHI, KONG and YAO, respectively. All the benchmark filters are taken from literatures [26]–[28], [30], [39]–[41] and their specifications can be found in Tables II and III. The actual power consumptions of different filter architectures are simulated. The synthesis and simulation results are listed in Table IV. The IC technology used is UMC 0.18μm standard cell, and Primepower, a gate level tool, is used to measure the power consumption of each architecture. The filter is operating at clock frequency of 50 MHz and the input signal of all FIR filter are 8 bit random samples with uniform amplitude distribution. The proposed designs and that of both SHI and YAO are designed using a 2.4-GHz desktop PC. Due to the fact that the computational complexity of MILP based algorithm is exponentially increased with filter order and coefficient wordlength, the runtime of the programs is restricted to 24 hours.

From Table IV, it can be seen that except filter B with order more than 100, the proposed bit-level optimization method generates all designs using the least number of FAs. The average savings for those filters with order less 100 compared to SHI's, YAO's and KONG's are 2.12%, 1.64% and 8.31%, respectively. It can be seen that the average of the area improvement compared to the best state-of-the-art word-level algorithm is only about 2% and this improvement seems not so significant. The reason is because the dominate part of the area for a digital FIR filter is the SAs and delays instead of the MBAs. For example, the area of SAs and delays of benchmark filters is at least 80% of that of the overall filters. The area of SAs and delays is related to the coefficients values and independent of how to synthesize the coefficients, because the area of SAs and delays are determined by the width of the SAs and delays, which are further determined by the EWL of the coefficients for a given input wordlength. Since the proposed algorithm and those word-level algorithms [30], [33] adopt the same filter minimum EWL, the variations of the EWL of each coefficient of the proposed algorithm from those word-level algorithms are relatively small. Therefore, the room left for the optimization of the dominant area of SAs and delays is relative small.

Although the saving in the number of FAs is small, the power saving is more significant as seen in Table IV. The average savings are 13.6% , 8.0% and 26.1%, respectively, compared with the 3 algorithms. The lower number of FAs is just one of the factors for the lower power consumption. The more important factor is the reduced average adder depth (AAD) shown in Table IV. The AAD accounts both MBAs' and SAs' contributions and is defined as

$$AAD = \frac{\sum_i AD^i_{MBA} + \sum_j AD^j_{SA}}{NA}, \tag{12}$$

where AD^i_{MBA} is the adder depth of the i-th MBA, NA is the number of adders and AD^j_{SA} is the adder depth of the j-th SA. Here, the adder depth means the number of cascaded adders along the longest path from the primary input to the generated output. For example if a filter has 3 coefficients {3,5,26}, it needs 3 MBAs to synthesize the coefficient fundamentals {3,5,13}. The adder depth of the 3 MBAs are 1, 1, 2, respectively. Besides that, there are two SAs in the delay chains corresponding to the coefficients 5 and 26. The adder depthes of these 2 SAs are 2 and 3, respectively. Thus, the overall AAD for this filter is computed as $(1+1+2+2+3)/(3+2)$, i.e., 1.8.

It can be seen from Table IV that in the case that the proposed algorithm generates the design with the same number of adders as the word-level algorithms, the AAD of the proposed algorithm is usually lower. This is credited to the bit-level optimization although the AAD is not explicitly enforced by the algorithm. It can be seen from (4) and (6) that to minimize the number of FAs, the fundamentals b_j and coefficient values $h(k)$ are all the smaller the better. In addition, S^i_{MBA} and S^i_{SA}, the left shift bit positions of the fundamentals (including the intermediate and coefficient fundamentals) are the larger the better. Both these conditions constrain the fundamentals to be small, whereas in the word-level optimization, there is no such constraints. In other words, if there are two fundamental sets which use the same number of adders, the bit-level optimization algorithm chooses the one with smaller fundamental values. The larger the fundamental values, the higher possibility that the fundamental is synthesized on a higher adder depth. For example, if a fundamental value is less than 200, the possibilities for it to be synthesized on adder depth one, two or three, are 0.12, 0.83 and 0.05, respectively; if the range of the fundamental is from 200 to 400, the possibilities for it to be synthesized on the 3 adders become 0.02, 0.76 and 0.26, respectively. Therefore, with smaller fundamental values, there is a higher probability to achieve a lower AAD for the overall filter.

The above results confirm the massive previous works [12], [42]–[46] which suggest that low AAD is important

TABLE IV

RESULTS AND COMPARISON OF 9 BENCHMARK FILTERS. AAD IS THE AVERAGE ADDER DEPTH, NA IS THE NUMBER OF ADDERS, NFA IS THE NUMBER OF FAS, AND POWER IS THE SIMULATED POWER CONSUMPTION OF THE FILTER.

Filters	method	EWL	AAD	NA	NFA	Power(mw)	Area (sq. um)	runtime	Power Saving of the proposed†(%)	Area Saving of the proposed†(%)
G1	Proposed†	6	1.29	17	209	1.31	30794	0.47s	-	-
	SHI [30]	6	1.58	17	222	1.65	31903	0.39s	20.6	3.48
	YAO [33]	The same as SHI [30]						1s	20.6	3.48
	Proposed††	The same as Proposed†						0.47s	0	0
Y1	Proposed†	10	2.14	29	416	4.42	65627	1m21s	-	-
	SHI [30]	10	2.75	29	454	6.06	69851	5m09s	27.1	6.05
	YAO [33]	10	2.17	29	428	4.73	66783	42s	6.5	1.73
	Proposed††	The same as Proposed†						12m37s	0	0
Y2	Proposed†	11	2.73	37	721	8.01	85641	11s	-	-
	SHI [30]	The same as Proposed†						40m26s	0	0
	YAO [33]	The same as Proposed†						24s	0	0
	Proposed††	The same as Proposed†						7m21s	0	0
Y3	Proposed†	11	2.14	57	901	16.26	131191	17h54m	-	-
	SHI [30]	11	2.59	59	972	21.39	138651	24*	24.0	5.38
	YAO [33]	11	2.14	58	919	16.57	132287	7m26s	2.0	0.83
	Proposed††	11	2.61	60	983	22.57	139832	24h*	27.9	6.18
S2	Proposed†	10	2.41	76	1225	24.05	171143	8h36m	-	-
	SHI [30]	10	2.56	76	1255	27.30	173732	16h28m	11.9	1.49
	YAO [33]	11	2.51	76	1268	27.11	174849	4h9m	11.3	2.12
	KONG [32]	13	N/A	81	N/A	N/A	N/A	N/A	N/A	N/A
	Proposed††	10	2.47	79	1286	25.87	175477	24h*	7.0	2.47
L2	Proposed†	10	2.20	72	1127	21.97	164586	1h41m	-	-
	SHI [30]	10	2.31	73	1151	24.51	165696	16h28m	10.4	0.67
	YAO [33]	10	N/A°	73	N/A°	N/A°	N/A°	12h24m	N/A°	N/A°
	Proposed††	The same as Proposed†						24h*	0	0
L3	Proposed†	7	1.83	36	513	5.87	76579	57s	-	-
	SHI [30]	7	1.83	36	513	5.87	76579	12m35s	0	0
	YAO [33]	7	1.80	38	536	5.91	78857	1m35s	0.7	2.89
	Proposed††	The same as Proposed†						37m41s	0	0
A	Proposed†	10	2.34	67	1088	20.37	157243	11h21m	-	-
	SHI [30]	10	2.53	68	1091	23.88	157911	24h*	14.7	0.42
	YAO [33]	The same as SHI [30]						2h7m	14.7	0.42
	KONG [32]	13	2.56	72	1199	27.56	171494	25m41s	26.1	8.31
	Proposed††	10	2.21	72	1143	20.15	162123	24h*	−1.1	3.01
	Proposed††	10	2.35	69	1123	21.04	158687	120h*	3.2	0.91
B	Proposed†	338	1.72	109	1688	32.09	237387	24h*	-	-
	SHI [30]	No feasible solution						24h*	N/A	N/A
	YAO [33]	8	1.72	105	1615	30.12	230339	24h*	−6.5	−3.06
	KONG [32]	11	NA	114	N/A	N/A	N/A	21m7s	N/A	N/A
	Proposed††	No feasible solution						24h*	N/A	N/A

* Manually stopped at the specific hours.
† The proposed algorithm using sparsity.
†† The proposed algorithm not using sparsity.
N/A Not available.
N/A° Not available due to a coefficient is missing in the published data.

in the design of low power consumption FIR filters. Based on the power models proposed in [42], [43], the MCM algorithms in [44]–[46] explicitly constrain the adder depth when the number of adders or FAs is minimized. Low power consumption is achieved in such algorithms.

With the savings achieved in most examples, for the long filter B, the proposed algorithm generates a design requiring more FAs and consuming more power than YAO's algorithm. The proposed algorithm is not efficient when the filter length is longer than 100 because the search only can cover a small portion of the overall search space in 24 hours in such cases.

As to design time, obviously, KONG's algorithm is the best, since it is a greedy algorithm with compromised hardware cost. The rest 3 algorithms(including the proposed one) are all MILP based tree search. Among these tree search algorithms, YAO's algorithm constructes the tree as a ternary tree and generally is faster than the other two which are the complete tree search algorithms. However, in some cases (of Filters Y2, L2 and L3), the proposed one is faster than that of YAO's algorithm. The improvement in speed is credited to the use of filter sparsity. To show the contrast, the design time and the total number of FAs obtained by the proposed bit-level optimization without using the sparsity are also listed in Table IV. These designs are also constrained to 24 hours. It can be seen that the obtained FA costs may be even higher than that of the word-level designs [30], [33] for some cases, and the design time is significantly increased. In particular, a design of filter A without using the sparsity is run for 5 days and the result is also listed in Table IV. In terms of FAs, the design obtained in 5 days is worse than the bit-level design using the sparsity obtained in 21 and half hours, and it is even worse than the word-level designs.

Overall speaking, the proposed bit-level optimization algorithm incorporating the sparse filter design technique outper-

TABLE V
IMPULSE RESPONSE AND SUBEXPRESSION BASES OF FILTERS G1, Y1, Y3 AND L3

Filter G1 $h(n) = h(15-n)$ for $0 \le n \le 7$
passband gain:193.5701 EWL: 6
subexpression set {5,39}
$h(0), ..., h(7)$: 1,2,-1,-8,-8,8,39,64
Filter Y1 $h(n) = h(29-n)$ for $0 \le n \le 14$
passband gain:2531.99113 EWL: 10
subexpression set { 15, 5, 11, 49, 501, 945 }
$h(0), ..., h(14)$: -2,-8,0,16,15,-20,-44,0,80,64,-88,-196,0,501,945
Filter Y3 $h(n) = h(49-n)$ for $0 \le n \le 24$
passband gain:4948.6720 EWL: 11
subexpression set { 3 ,11, 5, 19, 7, 91, 31, 65, 213, 127 }
$h(0), ..., h(24)$: 8,12,0,-16,-6,22,22,-20,-38,12,56,3,-80,-40,91,91,-88 -160, 62,260,12,-426,-224,895,2032
Filter L3 $h(n) = h(35-n)$ for $0 \le n \le 18$
passband gain: 517.7814 EWL: 7
subexpression set { 5, 7, 3, 25, 31}
$h(0), ..., h(17)$: 5,0,-4,-7,-8,3,12,12,4,-10,-20,-25,-12,20,62,100,124

TABLE VI
IMPULSE RESPONSE AND SUBEXPRESSION BASES OF FILTERS S2, L2 AND A

Filter S2 $h(n) = h(59-n)$ for $0 \le n \le 29$
passband gain:12107.8854 EWL: 10
subexpression set{5, 3, 25, 19, 55, 9, 45, 51, 27, 69, 13, 33, 533,649 189, 851,29, 491, 63}
$h(0), ..., h(29)$:5,6,5,5,5,0,-4,-12,-25,-38,-55,-72,-90,-102,-110,-108, -96,-69,-26,33,108,200,304,416,522,649,756,851, 928,982,1008
Filter L2 $h(n) = h(62-n)$ for $0 \le n \le 31$
passband gain:4349.079940 EWL: 10
subexpression set{3, 5, 7, 19, 9, 15, 51, 169, 201, 85, 379, 43, 461, 1009}
$h(0), ..., h(31)$:3,5,7,6,0,-10,20,-24,-19,-3,19,36,38,20,-14,-48,-64, -48,0,60,102,96,32,-72,-169,-201,-120,85,379, 688,922,1009
Filter A $h(n) = h(58-n)$ for $0 \le n \le 29$
passband gain:6085.029204 EWL: 10
subexpression set{3, 5, 7, 29, 77, 79, 177, 23, 131, 87, 199, 209, 975}
$h(0), ..., h(29)$:3,5,7,8,0,-10,20,-28,-29,-20,0,28,56,77,79,56,8,-58, -128,-177,-184,-131,-10,174,398,632,836,975,1024,

forms the existing best word-level optimization methods in terms of hardware cost and power consumption, and the design time is still maintained at an acceptable level if the filter order is not higher than 100. The coefficient sets obtained using the proposed algorithm for the benchmark filters G1, Y1,Y3, L3, S2, L2 and A are listed in Tables V and VI for references.

V. DISCUSSIONS

In this section, the computational efficiency improvement is justified. The enlightenment from the design examples regarding to the design criteria (i.e., the number of FAs) and the power consumption is briefly discussed.

A. Justification for the Computational Efficiency Improvement

In the MILP based tree search method, the size of search space is approximated to

$$Space = R^{N_{opt}} \tag{13}$$

where N_{opt} is the number of coefficients to be optimized, and R is assumed to be the average range of each coefficient. Form (13), it can be seen that by reducing either R or N_{opt}, the overall search space can be significantly reduced. As mentioned in Introduction, with exploiting the sparsity of filters, both R and N_{opt} are reduced, such that the search efficiency is improved. Our design experience shows that by fixing one coefficient to 0, the value of R can be reduced at least by 5%. Thus the overall search space becomes $0.95^{N_{opt}-1}R^{N_{opt}-1}$. Taking N_{opt} and R to be 30 and 20 as an example, the search space becomes only 1.1% of the original one, i.e, the search efficiency is improved more than 93 times. In other words, fixing one coefficient to 0 may already lead to significant increase in the search speed.

The results in Table IV verify that the significant speed improvement with maintained feasibility of the discrete coefficient filters for the cases that the filter order is not higher than 100. As to filter B, although the search efficiency increases exponentially with every fixed zero, the original search space is too large to be shrunk to an acceptable level. Within 24 hours, the obtained result is still worse than that obtained by [33]. Nevertheless, the proposed ideas about the bit-level optimization incorporated with the sparse techniques may also be applied to [33] to improve its performance further.

B. Power Consumption Indicator

Fig. 2 shows the normalized power, defined as power/(power of the proposed††), and the normalized AAD-NFA product, defined as (AAD-NFA product)/(AAD-NFA product of the proposed††), for the 5 designs of filter A (given in Table IV) evaluated at bit-level. The normalized data are listed in Table VII. When the 5 designs are ordered with increased normalized power, the normalized AAD-NFA product also increases monotonically. For comparison, the normalized NFA, NA and AAD, defined in the similar way, are also listed and plotted. Obviously, the values of NFA and NA oscillate randomly with respect to the monotonically increased power in Fig. 2. Therefore, NA or NFA standalone is not a good indicator for the power consumption. On the other hand, in Fig. 2, AAD also increases monotonically, although the increasing trend of AAD is less consistent than that of AAD-NFA product, with respect to the increasing trend of power. However, it should be noted that this set of AADs is obtained when NFA or NA is minimized. If AAD alone is minimized, an unfavorable but inevitable scenario is that

TABLE VII
THE NORMALIZED POWER CONSUMPTION, AAD-NFA PRODUCT, AAD, NFA AND NA OF THE 5 DESIGNS OF FILTER A.

Designs	Proposed††	Proposed†	Proposed††(5d)	(SHI,YAO)	KONG
Normalized Power	1.000	1.011	1.044	1.185	1.367
Normalized AAD-NFA	1.000	1.026	1.045	1.093	1.215
Normalized AAD	1.000	1.060	1.063	1.145	1.158
Normalized NFA	1.000	0.969	0.983	0.955	1.049
Normalized NA	1.000	0.931	0.958	0.944	1.000

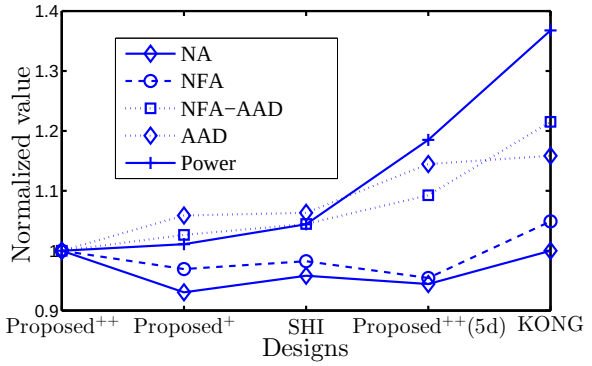

Fig. 2. The normalized value of NA, NFA , AAD-NA and AAD-NFA to power consumption for 5 designs of the filter A.

unnecessary low adder depth intermediate fundamentals are employed to synthesize the coefficient fundamentals. For example, to synthesize a coefficient set {3,11,13,21}, the solution of 7 adders (including 3 SAs) with AAD equal to 2.3 is the is the commonly accepted best results when power is concerned, because no additional intermediate fundamentals are required and each fundamental is synthesized on its minimum adder depth. However, if minimum AAD alone is considered, obviously any intermediate fundamentals with adder depth smaller than 2.3 is favored to be included in the synthesis of the coefficients. For example, 5 and 7 may be inserted to synthesized coefficient 11 and 13, and the resultant AAD becomes 2. This will increase the power consumption inevitably.

According to the above analysis, AAD-NFA product among the 5 criteria considered in this section is the best power indicator in the evaluation of the power of FIR filters. The AAD-NFA product potentially may be used as an objective function in the tree search optimizations. Alternatively, instead of minimizing NFA with implicit AAD constraint as in this paper, minimization of the adder depth of each coefficient as in [12], [44]–[46] with proper constraints on the use of intermediate fundamentals may also achieve the lower power consumption. However, in both of the above suggestions, the search problem including the estimation of node cost and cut-off scheme needs to be redefined, and similar low power designs are expected to be obtained. This may be further verified in the future researches.

VI. CONCLUSION

In this paper, a bit-level multiplierless FIR filter optimization technique is proposed. By incorporating the sparse filter with proper sparsity control scheme, for the first time, the bit-level efficient optimization of multiplierless FIR filters becomes possible. The formulation of the problem optimizes not only MBAs, but also SAs. Design examples show that FIR filters with low hardware cost and low power consumption can be obtained in a reasonable time. Results also show that power consumption is more related to AAD-NFA product than NFA or NA. This result confirms the previous findings of MCM algorithms.

REFERENCES

[1] Y. Neuvo, D. Cheng-Yu, and S. K. Mitra, "Interpolated finite impulse response filters," *IEEE Trans. Acoust., Speech, Signal Process*, vol. 32, pp. 563–570, Jun. 1984.

[2] T. Saramaki, Y. Neuvo, and S. K. Mitra, "Design of computationally efficient interpolated FIR filters," *IEEE Trans. Acoust., Speech, Signal Process*, vol. 35, pp. 70–88, Jan. 1984.

[3] Y. C. Lim, "Frequency-response masking approach for the synthesis of sharp linear phase digital filters," *IEEE Trans. Circuits Syst.*, vol. 33, pp. 357–364, Apr. 1986.

[4] Y. C. Lim and Y. Lian, "The optimum design of one- and two-dimensional FIR filters using the frequency-response masking technique," *IEEE Trans. Circuits Syst. II*, vol. 40, pp. 88–95, Feb. 1993.

[5] W. J. O. J. T. Kim and Y. H. Lee, "Design of nonuniformly spaced linear-phase FIR filters using mixed integer linear programming," *IEEE Trans. Signal Process*, vol. 44, pp. 123–126, Jan. 1996.

[6] Y. S. Song and Y. H. Lee, "Design of sparse FIR filters based on branch-and-bound algorithm," in *Proc. IEEE 40th MidwestSymp. Circuits Syst.*, Sacramento, CA,, Aug. 1997, pp. 1445–1448.

[7] W. S. Lu and T. Hinamoto, "Digital filters with sparse coefficients," in *Proc. IEEE Int. Symp. Circuits Syst.*, Paris, France, May 2010, pp. 169–172.

[8] T. Baran, D. Wei, and A. V. Oppenheim, "Linear programming algorithms for sparse filter design," *IEEE Trans. Signal Process*, vol. 58, pp. 1605–1617, Mar. 2010.

[9] A. Jiang, H. K. Kwan, and Y. Zhu, "Peak-error-constrained sparse FIR filter design using iterative SOCP," *IEEE Trans. Signal Process*, vol. 60, pp. 4035–4044, Mar. 2012.

[10] H. Zhao, W. B. Ye, and Y. J. YU, "Sparse FIR filter design based on genetic algorithm," in *Proc. IEEE Int. Symp. Circuits Syst.*, Beijing, China, May 2013, pp. 97–100.

[11] L. Aksoy, E. Gunes, and P. Flores, "An exact breadth-first search algorithm for the multiple constant multiplications problem," in *Proc. IEEE NORCHIP*, Tallinn, Nov. 2008, pp. 41–46.

[12] M. Faust and C. H. Chang, "Minimal logic depth adder tree optimization for multiple constant multiplication," in *Proc. IEEE Int. Symp. Circuits Syst.*, Paris, May 2010, pp. 457–460.

[13] R. I. Hartley, "Subexpression sharing in filters using canonic signed digit multipliers," *IEEE Circuits and Syst. II, Analog Digit. Signal Process*, vol. 43, pp. 677–688, Oct. 1996.

[14] D. R. Bull and D. H. Horrocks, "Primitive operator digital filters," *IEE Proceedings, Part G: Circuits, Devices and Systems*, vol. 138, pp. 401–412, Jun. 1991.

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

[16] R. Pasko, P. Schaumont, V. Derudder, S. Vernalde, and D. Durackova, "A new algorithm for elimination of common subexpressions," *IEEE Trans. Computer-Aided Design*, vol. 18, pp. 56–68, Jan. 1999.

[17] M. Potkonjak, M. B. Srivastava, and A. P. Chandrakasan, "Multiple constant multiplications: efficient and versatile framework and algorithms for exploring common subexpression elimination," *IEEE Trans. Circuits and Systems*, vol. 15, pp. 151–165, Feb. 1996.

[18] A. G. Dempster, S. S. Demirsoy, and I. Kale, "Designing multiplier blocks with low logic depth," in *Proc. IEEE International Symposium on Circuits and Systems*, Scottsdale, Arizona, May 2002, pp. 773–776.

[19] O. Gustafsson and L. Wanhammar, "A novel approach to multiple constant multiplication using minimum spanning trees," in *Proc. IEEE Midwest Symp. Circuits Syst*, Tulsa, OK, Aug. 2002, pp. 652–655.

[20] O. Gustafsson, H. Ohlsson, and L. Wanhammar, "Improved multiple constant multiplication using minimum spanning trees," in *Proc. IEEE Asilomar Conf. Signals Syst. Comp.*, Pacific Grove, CA, Nov. 2004, pp. 63–66.

[21] O. Gustafsson, "A difference based adder graph heuristic for multiple constant multiplication problems," in *Proc. IEEEInt. Symp. Circuits Syst.*, New Orleans, LA, May 2006, pp. 1097–1100.

[22] P. F. L. Aksoy, E. Costa and J. Monteiro, "Optimization of area in digital FIR filters using gate-level metrics," in *Design Automation Conference (DAC)*, San Diego, CA, 2007, pp. 420–423.

[23] K. Johansson, O. Gustafsson, and L. Wanhammar, "Bit-level optimization of shift-and-add based FIR filters," in *Proc. IEEE International Conference on Electronics, Circuits and Systems*, Marrakech, Dec. 2007, pp. 713–716.

[24] Y. Pan and P. K. Meher, "Bit-level optimization of adder-trees for multiple constant multiplications for efficient FIR filter implementation," *IEEE Trans. Circuits and Systems I*, vol. 61, pp. 455–462, Feb. 2014.

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

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

[27] Y. J. Yu and Y. C. Lim, "Design of linear phase FIR filters in subexpression space using mixed integer linear programming," *IEEE Trans. Circuits Syst. I*, vol. 54, pp. 2330–2338, Oct. 2007.

[28] Y. J. Yu and Y. C. Lim, "Optimization of linear phase FIR filters in dynamically expanding subexpression space," *Circuits Systems and Signal Processing*, vol. 29, pp. 65–80, Feb. 2010.

[29] M. Aktan, A. Yurdakul, and G. Diznda, "An algorithm for the design of low-power hardware-efficient FIR filters,," *IEEE Trans. Circuits and Systems*, vol. 55, pp. 1536–1545, Jul. 2008.

[30] D. Shi and Y. J. Yu, "Design of linear phase FIR filters with high probability of achieving minimum number of adders," *IEEE Trans. Circuits Syst. I*, vol. 58, pp. 126–136, Jan. 2011.

[31] A. Shahein, Q. Zhang, and Y. Manoli, "A novel hybrid monotonic local search algorithm for FIR filter coefficients optimization," *IEEE Trans. Circuits and Systems*, vol. 59, no. 3, pp. 616–627, 2012.

[32] B. Y. Kong and I.-C. Park, "FIR filter synthesis based on interleaved processing of coefficient generation and multiplier-block synthesis," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, no. 8, pp. 1169–1179, 2012.

[33] C. Y. Yao, W. C. Hsia, and Y. H. Ho, "Designing hardware-efficient fixed-point FIR filters in an expanding subexpression space," *IEEE Trans. Circuits and Systems I.*

[34] D. Shi and Y. J. Yu, "Design of discrete-valued linear phase FIR filters in cascade form," *IEEE Trans. Circuits and Systems*, vol. 58, no. 7, pp. 1627–1636, July 2011.

[35] J. Yli-Kaakinen and T. Saramaki, "Multiplication-free polynomial-based FIR filters with an adjustable fractional delay," *Circuits, Systems and Signal Processing*, vol. 25, pp. 265–294, Apr. 2006.

[36] Y. C. Lim and S. R. Parker, "FIR filter design over a discrete power-of-two coefficient space," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-31, pp. 583–591, Jun. 1983.

[37] Y. C. Lim, "Design of discrete-coefficient-value linear phase FIR filters with optimum normalized peak ripple magnitude," *IEEE Trans. Circuits Syst.*, vol. 37, pp. 1480–1486, Dec. 1990.

[38] A. G. Dempster, S. S. Demirsoy, and I. Kale, "Designing multiplier blocks with low logic depth," in *Proc. IEEE Int. Symp. Circuits Syst.*, Scottsdale, AZ, May 2002, pp. 773–776.

[39] Y. C. Lim and S. R. Parker, "Discrete coefficient FIR digital filter design based upon an LMS criteria," *IEEE Trans. Circuits Syst.*, vol. 30, pp. 723–739, Oct. 1983.

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

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

[42] S. S. Demirsoy, A. Dempster, and I.Kale, "Transition analysis on FPGA for multiplier-block based FIR filter structures," in *Proc. The 7th IEEE International Conference on Electronics, Circuits and Systems*, Lebanon, Dec. 2000, pp. 862–865.

[43] S. S. Demirsoy, A. G. Dempster, and I. Kale, "Power analysis of multiplier blocks," in *Proc. IEEE International Symposium on Circuits and Systems*, Phoenix, Arizona, May 2002, pp. 297–300.

[44] K. Johansson, L. S. DeBrunner, O. Gustafsson, and L. Wanhammar, "Design of multiplierless FIR filters with an adder depth versus filter order trade-off," in *Proceedings of the 43rd Asilomar conference on Signals, systems and computers*, Pacific Grove, California, USA, 2009, pp. 744 – 748.

[45] K. Johansson, O. Gustafsson, L. DeBrunner, and L. Wanhammar, "Minimum adder depth multiple constant multiplication algorithm for low power FIR filters," in *Proc. IEEE Int. Symp. Circuits Syst.*, Rio de Janeiro, Brazil, May 2011, pp. 15–18.

[46] P. F. L. Aksoy, E. Costa and J. Monteiro, "Design of low-power multiple constant multiplications using low-complexity minimum depth operations," in *Proceedings of the Great Lakes Symposium on VLSI*, New York, 2011, pp. 79–84.