Design of Optimal Multiplierless FIR Filters with Minimal Number of Adders

Martin Kumm, Anastasia Volkova, Silviu-Ioan Filip

Design of Optimal Multiplierless FIR Filters with Minimal Number of Adders

Martin Kumm, *Member, IEEE*, Anastasia Volkova, and Silviu-Ioan Filip, *Member, IEEE*

***Abstract*—This work presents two novel methods that simultaneously optimize both the design of a finite impulse response (FIR) filter and its multiplierless hardware implementation. We use integer linear programming (ILP) to minimize the number of adders used to implement a direct/transposed FIR filter adhering to a given frequency specification. The proposed algorithms work by either fixing the number of adders used to implement the products (multiplier block adders) or by bounding the adder depth (AD) used for these products. The latter can be used to design filters with minimal AD for low power applications. In contrast to previous multiplierless FIR filter approaches, the methods introduced here ensure adder count optimality. We perform extensive numerical experiments which demonstrate that our simultaneous filter design approach yields results superior to those in the literature.**

***Index Terms*—FIR filters, multiplierless implementation, ILP optimization, MCM problem**

I. Introduction

FIR filters are fundamental building blocks in digital signal processing (DSP). They provide strict stability and phase linearity, enabling many applications. However, their flexibility typically comes at the expense of a large number of multiplications, making them compute-intensive. Hence, many attempts have been made in the last four decades to avoid costly multiplications and to implement FIR filters in a multiplierless way [1]–[23].

One of the most widespread ways to do so is to replace constant multiplications by additions, subtractions and bit shifts. Consider, for instance, the multiplication by a constant coefficient 23. It can be computed without dedicated multipliers as

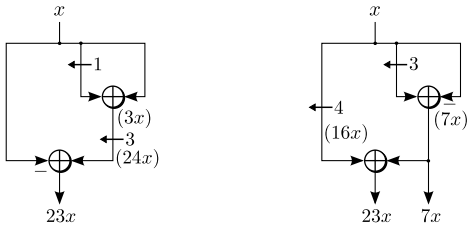
$$23x = 8 \cdot (2x + x) - x = ((x << 1) + x) << 3) - x, \quad (1)$$

where $(x << b)$ denotes the arithmetic left shift of x by b bits. This computation uses one addition and one subtraction. Since in this context the add and subtract operations both have similar hardware cost, the total number of add/subtract units is usually referred to as *adder cost*. Bit shifts can be hard-wired in hardware implementations and do not contribute any cost.

M. Kumm is with the Fulda University of Applied Sciences, 36037 Fulda, Germany (e-mail: martin.kumm@cs.hs-fulda.de).

A. Volkova is with University of Nantes, Nantes, France (e-mail: anastasia.volkova@univ-nantes.fr).

S.I. Filip is with University of Rennes, Inria, CNRS, IRISA, Rennes, France (e-mail: silviu.filip@inria.fr).



(a) SCM with 23 (b) MCM with $\{7, 23\}$

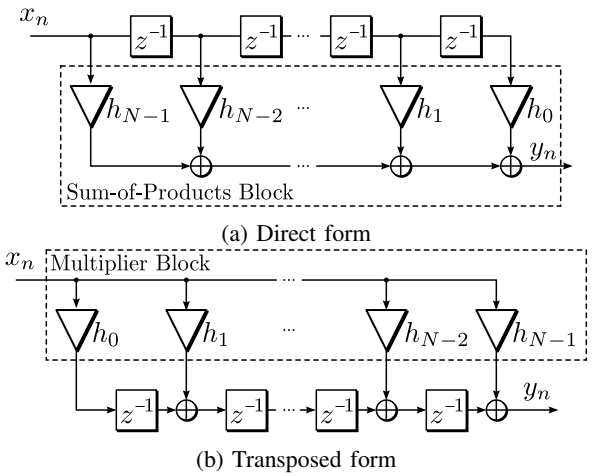
Fig. 1: Different adder circuits for constant multiplications.

For (1), this is illustrated in Fig. 1a. In general, the task of finding a minimal adder circuit for a given constant is known as the single constant multiplication (SCM) problem and is already an NP-complete optimization problem [24].

Such a problem extends to multiplication with multiple constants, which is necessary when implementing FIR filters. It is called multiple constant multiplication (MCM). Here, some of the intermediate factors like the adder computing $3x$ in Fig. 1a can be shared among different outputs. Take for example the coefficients $\{7, 23\}$; Fig. 1b shows a solution for multiplying with both coefficients at an adder cost of only two. The corresponding optimization problem is called the MCM problem and has been addressed by numerous heuristic [25]–[27] and optimal [28]–[30] approaches.

Fig. 2 shows the two most popular structures used to implement FIR filters: the direct and transposed forms. The result of an MCM solution can be directly placed in the multiplier block of the transposed form from Fig. 2b. The total adder cost can be modelled as the sum of the number of *multiplier block adders* and the remaining ones, commonly called *structural adders*. The transposed form can be obtained from the direct form by transposition [31]. As the transposition of a single-input-single-output system does not change the adder count, it leads to the same adder cost. In the end, it does not matter which one of these two filter structures is actually optimized with respect to the adder count.

In the MCM optimization problem, it is assumed that the coefficients are known and already quantized to a fixed-point (or integer) representation. The design of FIR filters with fixed-point coefficients and a minimum frequency response approximation error is itself a well-known optimization problem, going back to at least [32] with subsequent extensions and improvements [1]–[4], [6], [8]–[10], [12], [15], [16], [20], [22], [23], [33], [34]. It is however often the case in practice that a bounded frequency response is acceptable. In fact, there may



(a) Direct form

(b) Transposed form

Fig. 2: Structure of FIR filters.

be a large number (often hundreds or more) of different fixed-point coefficient sets that meet such a specification. Because of this, the most widely used filter design technique is a two-step approach in which: (a) a filter with real-valued coefficients is derived using standard approaches (*e.g.* windowing or Chebyshev) after which (b) the obtained values are quantized and optimized for a minimum number of adders using an MCM approach.

Still, when optimizing for resources (in this particular case, total number of adders), the obtained results are usually far from optimal (see Section VI-C1 for a comparison). Therefore, a lot of effort in fixed-point FIR filter design has gone into aggregate one-step methods that put emphasis on resource use. Finding a minimal adder circuit for a given filter specification has thus been approached by several authors [1]–[4], [6], [9], [10], [15], [16], [20], [23].

However, to the best of our knowledge, only one of the previous work [23] has actually addressed this one-step design of multiplierless FIR filters in an optimal way, using a custom branch and bound algorithm. Here, by *optimal* multiplierless filter we mean a direct/transposed form FIR filter requiring a minimum number of addition/subtraction operations to meet a target frequency specification, as well as constraints on the maximum coefficient word size and filter order.

This work presents for the first time a closed form ILP formulation for solving this problem. The authors believe that the beauty of having a mathematical ILP formulation allows for easy re-implementations and offers a convenient framework for further extensions. Examples are minimizing the power consumption [35]–[37], the inclusion of lookup table-based multipliers [38], targeting 3-input adders for FPGAs [39], pipelining [40], optimizing FIR cascaded filters [19], etc. Besides that, any performance improvements to ILP solvers directly translate into faster runtimes for solving the multiplierless FIR filter design problem.

The main contributions of our work can be summarized as follows:

- We present for the first time a solution for the optimal multiplierless design (in terms of adder count) of FIR filters from a frequency specification using a closed form ILP formulation.
- We provide another ILP formulation that is capable of additionally limiting the adder depth inside the FIR filter.
- We show that relevant problem sizes can be addressed by current ILP solvers and that the adder complexity of well-known FIR filters can be further reduced compared to the most advanced methods.

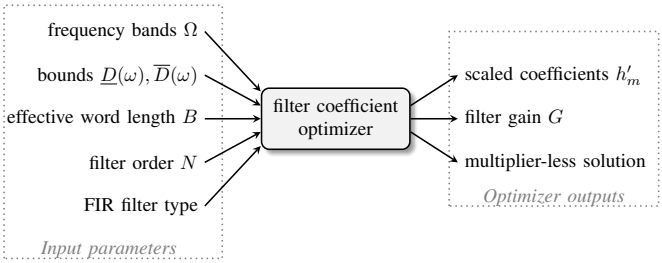


Fig. 3: Simplified multiplierless FIR filter design flow.

Our approach builds on previous ILP formulations [30], [40] for optimally solving the MCM problem, which we extend here to multiplierless FIR design. The major challenge is that such previous work assumes that the constants are known in advance, while in the FIR filter setting they are the unknowns. To address it we formalized the search space for the coefficients, incorporated the frequency domain constraints into the problem and "linked" the unknowns with the MCM problem inputs. We also formalized the notion of structural adders in order to exploit and privilege sparse filter implementations. In addition, the proposed tool takes advantage of a variety of techniques for reducing the search space of the obtained models.

In the following, we will give background information about previous work this paper is based on. In Section III and Section IV we describe the two ILP formulations that are at the core of the paper, whereas in Section V we talk about ideas meant to improve the practical runtime of the proposed algorithms. We then present experimental results accompanied by a comparison with the state-of-the-art (Section VI), followed by concluding remarks (Section VII).

II. Background

Multiplierless filter design problems usually start with a functional specification of the frequency domain behavior, together with the number of filter coefficients and their word lengths. An optimization procedure is applied to get a set of bounded integer coefficients together with their associated adder circuits for the constant multiplications needed in the final implementation. Summarized in Fig. 3, this section overviews these parameters and their interactions, together with the state-of-the-art design methods found in the literature.

TABLE I: Relation between filter order N, number of coefficients M and function $c_m(\omega)$ for different filter types

Type	Sym.	N	M	$c_m(\omega)$
I	sym.	even	$\frac{N}{2}+1$	$c_m(\omega) = \begin{cases} 1 & \text{for } m = 0 \\ 2\cos(\omega m) & \text{for } m > 0 \end{cases}$
II	sym.	odd	$\frac{N+1}{2}$	$c_m(\omega) = 2\cos(\omega(m+1/2))$
III	asym.	even	$\frac{N}{2}$	$c_m(\omega) = 2\sin(\omega(m-1))$
IV	asym.	odd	$\frac{N+1}{2}$	$c_m(\omega) = 2\sin(\omega(m+1/2))$

A. Linear Phase FIR Filters

An N-th order linear phase FIR filter can be described by its zero-phase frequency response

$$H_R(\omega) = \sum_{m=0}^{M-1} h_m c_m(\omega), \qquad \omega \in [0, \pi], \tag{2}$$

which has the property that its magnitude is identical to that of the transfer function, *i.e.,*

$$|H(e^{j\omega})| = |H_R(\omega)| \ . \tag{3}$$

The $c_m(\omega)$ terms are trigonometric functions and M denotes the number of independent coefficients after removing identical or negated ones due to symmetry. Both depend on the filter symmetry and on the parity of N as given in Table I.

Let $\underline{D}(\omega)$ and $\overline{D}(\omega)$ be the desired lower and upper bounds of the output frequency response $H_R(\omega)$. The associated frequency specification-based FIR filter design problem consists of finding coefficients h_m, $m = 0, \ldots, M-1$ that fulfill the constraints

$$\underline{D}(\omega) \leqslant H_R(\omega) \leqslant \overline{D}(\omega), \qquad \forall \omega \in \Omega, \tag{4}$$

where $\Omega \subseteq [0, \pi]$ is a set of target frequency bands (usually pass and stopbands). A standard approach in practice is to work with $\Omega_d \subseteq \Omega$, a uniform discretization of Ω. One number for the size of Ω_d found in the literature is $16M$ [34].

B. Fixed-point Constraints

Fixed-point (integer coefficient) FIR filter design problems further restrict the search space to integer variables $h'_m \in \mathbb{Z}$ with $|h'_m| < 2^B$, where the coefficients of $H_R(\omega)$ are

$$h_m = 2^{-B} h'_m \tag{5}$$

and $B \in \mathbb{N}$ is the *maximum effective word length* of each coefficient (excluding sign bit). Note that we do not rely on any number representations or other limited number spaces like many previous approaches [1], [2], [5]–[7], [9]–[11], [13], [14], [16].

To broaden the feasible set of efficient designs, some applications allow the use of a real-valued scaling factor $G > 0$ when computing the quantized fractional coefficients h_m. Equation (4) thus becomes

$$G\underline{D}(\omega) \leqslant H_R(\omega) \leqslant G\overline{D}(\omega), \qquad \forall \omega \in \Omega \ . \tag{6}$$

When the frequency specification contains a passband, it is called the passband gain [33]. Finding adequate bounds for G is dependent on the set/format of feasible h'_m coefficient values. If these values are constrained to a power of two space, the ratio between the upper and the lower bound on G does not need to be larger than 2 [33, Lemma 1]. Even when this is not the case, the interval $[0.7, 1.4]$ is frequently used [14], [20], [33]. For our tests, unless otherwise stated, we project the polytope described by (6) onto G in order to obtain a sufficiently large search domain $[\underline{G}, \overline{G}]$ (see Section V-A). In case a unity or fixed-gain filter is required we set the gain to $G = 1$.

C. Multiplierless FIR Filters

Formulas (5) and (6) are easily expressed as constraints in an ILP formulation. However, to ensure an optimal multiplierless design, further constraints are needed.

The way these constraints are constructed and used has varied over the years. Early research in this direction looked at multiplierless designs where each coefficient was represented by a limited number of signed power-of-two terms, optimized using branch-and-bound techniques [1]. Later, minimum signed digit (MSD) representations characterized by a minimum number of non-zero power-of-two terms were quickly adopted for this purpose [2], [5], [11], [16].

MSD representations can be used to find sharing opportunities of intermediate computations like the $7x$ term shown in Fig. 1b. One way is by searching and eliminating redundant bit patterns common to several coefficients, a technique called common subexpression elimination (CSE). Savings are obtained by performing the computation specified by the bit pattern and distributing the result to all coefficients depending on it [6], [7], [25]. However, the CSE search cannot deliver all possible sharing opportunities due to its dependency on the number representation [26] and the effect of hidden non-zeros [37]. To avoid them, graph-based approaches are commonly used in state-of-the-art MCM methods [25]–[30]. Some early work on multiplierless FIR filter design already considered this by incorporating the graph-based MCM algorithm of [25] into a genetic algorithm that optimizes the filter coefficients according to the adder cost [4]. A different approach is followed by [9], where a branch-and-bound-based ILP optimization is used; here, a pre-specified set of integer terms, called the subexpression space, has to be provided that can be shared among the different coefficient expansions. This work was later extended with a dynamic subexpression space expansion algorithm [13], [14], which, at least in the case of [14], claims to usually produce designs with a minimal number of adders. In contrast to these potentially slow branch-and-bound approaches, in [17], a fast polynomial-time heuristic for the design of low complexity multiplierless linear-phase FIR filters was proposed.

Although many of the approaches described above use optimal branch-and-bound or ILP methods, they are only used on a limited search space corresponding to the selected number representation. The branch-and-bound method described in [15] and later refined in [23] is the only other work the authors are aware of, besides this one, which addresses the optimal multiplierless FIR filter design problem regardless of

the number representation. The optimal algorithm in [23] is called SIREN. It performs a depth-first search on a search tree where each level consists of the possible values of one of the coefficients. When reaching the bottom of the tree, their optimal MCM algorithm [29] is used to determine the adder cost. Lower and upper bounds of these coefficients are computed using linear programming (LP) and clever ways are proposed to further prune the search tree. The first objective in [23] is to find the minimal effective word length B, with the number of adders being a secondary objective. The method could work in principle with any B, but run-time will become prohibitive, as the search tree increases exponentially with B.

Recent work has also focused on integrating filter coefficient sparsity, which can also have a big impact on the complexity of the final design [18], [41], [42] by reducing the number of structural adders. Also, other structures than the direct and transposed forms (see Fig. 2) have been shown to possess good properties. The factoring of FIR filters into a cascade of relatively small subsections can lead to a lower bit-level complexity [19]. Alternative structures have also been proposed [21], [42]–[44]; they provide lower word sizes for the structural adders, reducing resource use.

Besides optimizing the adder count, it was shown early that the power consumption of the resulting filter also strongly depends on the adder depth (AD), which is defined as the number of cascaded adders in the multiplier block [35]. Since then, many works have focused on limiting the AD either in MCM algorithms [36], [37], [40] or directly in multiplierless filter design methods [14]. Again, all of those approaches are heuristics that provide minimal AD, but not guarantee minimal adder cost.

A low-level metric, e.g. minimizing the number of full adders [45] or gates [46], would lead to more hardware-efficient results. However, modelling with respect to these metrics significantly increases the size and complexity of the optimization problem, limiting it's practicality. While counting adders is a larger-grain approach, it nevertheless gives a good indication of the hardware resources needed in practice, and is better-suited for efficient ILP modelling.

III. Multiplierless FIR Filters with Fixed Number of Multiplier Block Adders

Our first ILP model targets the design of generic multiplierless FIR filters regardless of their adder depth. It is based on a recently proposed MCM ILP formulation [30], where the goal is to directly compute the parameters of an MCM adder graph, if feasible, for a given number of adders. This idea is extended here for multiplierless FIR filter design by adding constraints on the frequency specification. As a result, we get an ILP model to design a multiplierless filter for a fixed number of adders in the multiplier block. To optimize the total number of adders, this ILP model is solved several times using an overall algorithm discussed in Section III-B. In the following, we first present the ILP formulation for the fixed number of multiplier block adders.

A. ILP Formulation for Fixed Multiplier Block Adder Count

The proposed ILP formulation is given in ILP Formulation 1 and uses the constants and variables listed in Table II. The objective is, given a fixed number of multiplier block adders A_{M}, to minimize the number of structural adders A_{S} (which depend on the number of zero filter coefficients, encoded by the binary decision variables $h_{m,0}$).

The resulting constraints can be roughly divided into frequency response conditions (C1, C2), equations linking the filter coefficients with the coefficients of the multiplier block (C3) and formulas describing the multiplierless realization of the multiplier block (C4 – C8).

The integer coefficients h'_m $(m = 0, \ldots, M-1)$ of the FIR filter are directly used as integer variables in the ILP formulation. The resulting frequency response is constrained in C1a by setting (2) and (5) into (6). Constraints C1b are so-called *lifting constraints*. These are actually not required to solve the problem, but can significantly reduce the search space and improve runtime performance. Specifically, they limit the range of the coefficients to lower $\underline{h_m}$ and upper $\overline{h_m}$ bounds. Constraint C2 similarly limits the range of the gain G. The computation of these bounds is considered in Section V-A.

Constraints C3a to C3c provide the connection between the filter coefficient h'_m and the (potentially shifted and sign-corrected) multiples computed in the multiplier block c_a or a zero coefficient. For that, the binary decision variables $o_{a,m,s,\phi} \in \{0,1\}$ encode if h'_m is connected to adder a of the multiplier block, shifted by s, and either added ($\phi = 0$) or subtracted ($\phi = 1$) in the structural adders (C3a). In case the coefficient is zero, a single binary decision variable $h_{m,0}$ is used (C3b). This encoding allows the optimization of structural adders by considering the $h_{m,0}$ variables in the objective function. For every zero coefficient, the corresponding structural adder(s) can be saved depending on the coefficient and filter type. Table III shows the number of structural adders for the different filter types. Overall, constraints C3c ensure that only one of the above cases is valid.

The remaining constraints C4 – C8 are identical to the ones used for solving the MCM problem from [30]. We give a brief description here, but refer the reader to [30] for a more detailed presentation. The multiplier block input is viewed as a multiplication by factor one ($c_0 = 1$) and is defined with constraint C4. Constraints C5 represent the actual add operation of adder a and its corresponding factor c_a. It is obtained by adding the shifted and possibly sign corrected factors of its left input $c^{\mathrm{sh,sg}}_{a,\ell}$ and its right input $c^{\mathrm{sh,sg}}_{a,\mathrm{r}}$. The source of the adder inputs is encoded by C6a/b. Indicator constraints C6a are used to set the value $c_{a,i}$ of the adder input $i \in \{\ell, \mathrm{r}\}$ to the actual factor when the corresponding decision variable $c_{a,i,k}$ is set. Indicator constraints are special constraints in which a binary variable controls whether or not a specified linear constraint is active. They are in-fact non-linear but supported by many modern ILP solvers and are also simple to linearize for other solvers (see [30]). Constraints C6b make sure that only one source is selected. The actual shift is constrained by C7a/b in a similar way: indicator constraints C7a are used to set the shifted factor $c^{\mathrm{sh}}_{a,i}$ according to the

ILP Formulation 1 Multiplierless FIR filters with fixed A_M

$$\text{minimize } A_\mathrm{S}(h_{m,0})$$

subject to

$$\text{C1a:} \quad G2^B\underline{D}(\omega) \leqslant \sum_{m=0}^{M-1} h'_m c_m(\omega) \leqslant G2^B\overline{D}(\omega),\ \forall\omega\in\Omega_d$$

$$\text{C1b:} \quad \underline{h_m} \leqslant h'_m \leqslant \overline{h_m},\ \forall m = 0,\dots,M-1$$

$$\text{C2:} \quad \underline{G} \leqslant G \leqslant \overline{G}$$

$$\text{C3a:} \quad h'_m = (-1)^\phi 2^s c_a \text{ if } o_{a,m,s,\phi} = 1 \quad \forall a = 0,\dots,A_\mathrm{M}, m = 0,\dots,M-1$$

$$\text{C3b:} \quad h'_m = 0 \text{ if } h_{m,0} = 1,\ \forall m = 0,\dots,M-1$$

$$\text{C3c:} \sum_{a=0}^{A_\mathrm{M}} \sum_{s=S_\mathrm{min}}^{S_\mathrm{max}} \sum_{\phi=0}^{1} o_{a,m,s,\phi} + h_{m,0} = 1,\ \forall m = 0,\dots,M-1$$

$$\text{C4:} \quad c_0 = 1$$

$$\text{C5:} \quad c_a = c_{a,\ell}^{\mathrm{sh,sg}} + c_{a,\mathrm{r}}^{\mathrm{sh,sg}},\ \forall a = 1,\dots,A_\mathrm{M}$$

$$\text{C6a:} \quad c_{a,i} = c_k \text{ if } c_{a,i,k} = 1,\ \forall a = 1,\dots,A_\mathrm{M}, i\in\{\ell,\mathrm{r}\}, k = 0,\dots,a-1$$

$$\text{C6b:} \quad \sum_{k=1}^{a-1} c_{a,i,k} = 1,\ \forall a = 1,\dots,A_\mathrm{M}, i\in\{\ell,\mathrm{r}\}$$

$$\text{C7a:} \quad c_{a,i}^{\mathrm{sh}} = 2^s c_{a,i} \text{ if } \varphi_{a,i,s} = 1 \quad \forall a = 1,\dots,A_\mathrm{M}, i\in\{\ell,\mathrm{r}\}, s = S_\mathrm{min},\dots,S_\mathrm{max}$$

$$\text{C7b:} \quad \sum_{s=S_\mathrm{min}}^{S_\mathrm{max}-1} \varphi_{a,i,s} = 1,\ \forall a = 1,\dots,A_\mathrm{M}, i\in\{\ell,\mathrm{r}\}$$

$$\text{C7c:} \quad \varphi_{a,\ell,s} = 0\ \forall s > 0$$

$$\text{C7d:} \quad \varphi_{a,\ell,s} = \varphi_{a,\mathrm{r},s}\ \forall s < 0$$

$$\text{C8a:} \quad c_{a,i}^{\mathrm{sh,sg}} = -c_{a,i}^{\mathrm{sh}} \text{ if } \phi_{a,i} = 1,\ \forall a = 1,\dots,A_\mathrm{M}, i\in\{\ell,\mathrm{r}\}$$

$$\text{C8b:} \quad c_{a,i}^{\mathrm{sh,sg}} = c_{a,i}^{\mathrm{sh}} \text{ if } \phi_{a,i} = 0,\ \forall a = 1,\dots,A_\mathrm{M}, i\in\{\ell,\mathrm{r}\}$$

$$\text{C8c:} \quad \phi_{a,\ell} + \phi_{a,\mathrm{r}} \leqslant 1,\ \forall a = 1,\dots,A_\mathrm{M}$$

corresponding decision variable $\varphi_{a,i,s}$.

Constraints C7c and C7d are both optional lifting constraints used to reduce the search space. As the filter coefficients can be shifted in constraint C3a, we can limit the constants of the multiplier block to odd numbers. This allows us to use the well-known fact that odd coefficients can be computed from odd numbers using one addition with either one operand left shifted while the other operand is not shifted or both operands are right shifted by the same value [47, Theorem 3].

To support subtractions, indicator constraints C8a/b are used to set the sign according to decision variable $\phi_{a,i}$. Finally, constraints C8c ensure that at most one input of the adder can be negative, as subtracting both inputs is typically more hardware demanding.

All of the integer variables from Table II are computed from integer constants or booleans and represent integer values. So they can be relaxed to real numbers to speed up the optimization.

TABLE II: Used constants (top) and variables (bottom) in ILP Formulation 1

Constant/Variable	Meaning
$A_\mathrm{M}\in\mathbb{N}$	Number of adders in the multiplier block
$M\in\mathbb{N}$	Number of filter coefficients
$S_\mathrm{min}, S_\mathrm{max}\in\mathbb{Z}$	Minimum and maximum shift
$A_\mathrm{S}\in\mathbb{N}$	Number of structural adders
$h'_m\in\mathbb{Z}$	Integer representation of filter coefficient
$h_{m,0}\in\{0,1\}$	true, if coefficient h'_m is zero
$c_a\in\mathbb{N}$	Constant computed in adder a
$c_{a,i}\in\mathbb{N}$	Constant of input $i\in\{\ell,\mathrm{r}\}$ of adder a
$c_{a,i}^{\mathrm{sh}}\in\mathbb{N}$	Shifted constant of input $i\in\{\ell,\mathrm{r}\}$ of adder a
$c_{a,i}^{\mathrm{sh,sg}}\in\mathbb{N}$	Shifted, sign corrected constant of input $i\in\{\ell,\mathrm{r}\}$ of adder a
$\phi_{a,i}\in\{0,1\}$	Sign of input $i\in\{\ell,\mathrm{r}\}$ of adder a (0:'+', 1:'−')
$c_{a,i,k}\in\{0,1\}$	true, if input i of adder a is connected to adder k
$\varphi_{a,i,s}\in\{0,1\}$	true, if input i of adder a is shifted by s bits
$o_{a,m,s,\phi}\in\{0,1\}$	true, if coefficient h'_m is connected to adder a, shifted by s and sign ϕ
$\underline{h_m}, \overline{h_m}\in\mathbb{Z}$	Lower and upper bound for filter coefficient $m = 0\dots M-1$
$G\in[\underline{G},\overline{G}]$	Gain of a variable gain filter ($G = 1$ when the gain is fixed)

TABLE III: Structural adder count for the different filter types

Type	no. of structural adders, $A_\mathrm{S}(h_{m,0})$
I	$N - h_{0,0} - 2\sum_{m=1}^{M-1} h_{m,0}$
II	$N - 2\sum_{m=0}^{M-1} h_{m,0}$
III	$N - 2\sum_{m=1}^{M-1} h_{m,0}$
IV	$N - 2\sum_{m=0}^{M-1} h_{m,0}$

B. Minimizing the Total Number of Adders

As the number of adders in the multiplier block A_M is fixed in ILP Formulation 1, we need to iterate over various values A_M to find the minimum number of total adders

$$A = A_\mathrm{M} + A_\mathrm{S}\ . \tag{7}$$

For that, we first search for a solution with minimal number of multiplier block adders by solving ILP Formulation 1 for $A_\mathrm{M} = 0, 1, 2, \dots$ until we obtain the first feasible solution.

This solution with minimum multiplier block adders $A_\mathrm{M,min}$ is not necessarily the global optimum as there might be a solution with $A_\mathrm{M} > A_\mathrm{M,min}$ and a smaller A_S. To account for this, we need a lower bound for the structural adders $A_\mathrm{S,min}$. This is obtained once at the beginning of the overall algorithm by solving the problem for a maximally sparse FIR filter, which we do by taking ILP Formulation 1 where only the constraints C1 – C3 are considered.

In case the structural adders A_S of solution with $A_\mathrm{M} = A_\mathrm{M,min}$ are not identical with $A_\mathrm{S,min}$, we continue to further increment A_M until we find a solution with $A_\mathrm{S} = A_\mathrm{S,min}$.

This is a safe stopping point since, by the optimality of ILP Formulation 1, there is no solution with larger A_{M} and smaller A_{S}. The solution with minimum total adders A found so far is hence also globally optimal. Typically, only a few iterations are necessary to reach this point.

IV. Multiplierless FIR Filters with Bounded Adder Depth

As discussed above, limiting the AD is important to reduce the power consumption of a filter. Unfortunately, adapting ILP Formulation 1 to limit the AD is not straightforward, as the topology of the adders and thus the AD is left open. We present in this section a novel ILP model for the design of multiplierless FIR filter with limited AD which is based on a formulation that was initially designed for optimizing pipelined MCM (PMCM) circuits [38], [40].

In contrast to ILP Formulation 1, the possible coefficients are precomputed for each adder stage s and selected using binary decision variables. The computation of the corresponding coefficient sets is given next.

A. Definition of Coefficient Sets

We use some notation and definitions originally introduced in [26]. First, we define the generalized add operation called $\mathcal{A}$-operation, which includes the shifts. An $\mathcal{A}$-operation has two input coefficients $u, v \in \mathbb{N}$ and computes

$$\mathcal{A}_q(u,v) = |2^{l_u}u + (-1)^{s_v}2^{l_v}v|2^{-r}, \tag{8}$$

where $q = (l_u, l_v, r, s_v)$ is a configuration vector which determines the left shifts $l_u,\ l_v \in \mathbb{N}_0$ of the inputs, $r \in \mathbb{N}_0$ is the output right-shift and $s_v \in \{0,1\}$ is a sign bit which denotes whether an addition or subtraction is performed.

Next, we define the set $\mathcal{A}_*(u,v)$ containing all possible coefficients which can be obtained from u and v by using exactly one $\mathcal{A}$-operation:

$$\mathcal{A}_*(u,v) := \{\mathcal{A}_q(u,v) \mid q \text{ is a valid configuration}\}\ . \tag{9}$$

A *valid* configuration is a combination of l_u, l_v, r and s_v such that the result is a positive odd integer $\mathcal{A}_q(u,v) \leqslant c_{\max}$. The reason for limiting the integers to odd values is that we can compute every even multiple by shifting the corresponding odd multiple to the left. The $c_{\max}$ limit is used to keep $\mathcal{A}_*(u,v)$ finite. It is chosen as a power-of-two value which is usually set to the maximum coefficient bit width B plus one [25], [26]

$$c_{\max} := 2^{B+1}\ . \tag{10}$$

For convenience, the $\mathcal{A}_*$ set is also defined for an input set $X \subseteq \mathbb{N}$ as

$$\mathcal{A}_*(X) := \bigcup_{u,v \in X} \mathcal{A}_*(u,v)\ . \tag{11}$$

We can now define the coefficients that can be computed at adder stage s, denoted as $\mathcal{A}^s$, by recursively computing the $\mathcal{A}_*$ sets

$$\mathcal{A}^0 := \{1\} \tag{12}$$

$$\mathcal{A}^s := \mathcal{A}_*(\mathcal{A}^{s-1})\ . \tag{13}$$

In addition, let $\mathcal{T}^s$ denote the set of (u,v,w) *triplets* for which $w \in \mathcal{A}^s$ can be computed using u and v from the previous stage (*i.e.,* $u, v \in \mathcal{A}^{s-1}$). $\mathcal{T}^s$ can be computed recursively, starting from the last stage s, which is equal to the maximum allowable AD:

$$\mathcal{T}^s := \{(u,v,w) \mid w = \mathcal{A}_q(u,v), \\ u,v \in \mathcal{A}^s,\ u \leqslant v,\ w \in \mathcal{A}^{s+1}\}. \tag{14}$$

To give an example, the first elements of $\mathcal{T}^1$ are $\mathcal{T}^1 = \{(1,1,1),(1,1,3),(1,1,5),(1,1,7),(1,1,9),(1,1,15),\ldots\}$. This set contains all the possible rules for computing multiples from the input within one stage of additions, while set $\mathcal{T}^2 = \mathcal{T}^1 \cup \{(1,3,11),(1,5,11),\ldots,(3,5,11),\ldots\}$ contains all the combinations of how elements in the next stage can be computed.

B. ILP Formulation for Fixed Adder Depth

The bounded AD model is given in ILP Formulation 2, while the corresponding constants and variables are given in Table IV.

In contrast to ILP Formulation 1, the objective is to directly minimize the total number of adders A, which is separated into adders in the multiplier block (A_{M}) and structural adders A_{S}. Similar to ILP Formulation 1, the constraints are divided into frequency response conditions (C1, C2), the link between the filter coefficients and the coefficients of the multiplier block (C3, C4) and the equations describing the multiplierless realization of the multiplier block (C5 – C8).

Constraints C1a/b and C2 are identical to the ones in ILP Formulation 1. Now, the connection between the odd multiplier block coefficients of the pre-computed sets and the filter coefficients is performed using binary decision variables. Let $h_{m,w} \in \{0,1\}$ be a binary decision variable that is true if the magnitude of h'_m is identical to w, *i.e.,*

$$h_{m,w} = \begin{cases} 1 & \text{when } |h'_m| = w \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

for $m = 0,\ldots,M-1$ and $w = 0,\ldots,2^B-1$. Furthermore, let ϕ_m determine the sign of h'_m as follows

$$\phi_m = \begin{cases} 0 & \text{when } h'_m \geq 0 \\ 1 & \text{otherwise}\ . \end{cases} \tag{16}$$

The value of each integer coefficient h'_m is selected by the indicator constraints C3a. In addition, constraints C3b make sure that only one value per filter coefficient is selected.

Next, we distinguish between coefficients that are computed for the selected stage (by using an addition) and coefficients that are just replicated from a previous stage. This replication can be either implemented by a simple wire (as this was implied in ILP Formulation 1) or in case of a pipelined implementation of the multiplier block, it will be implemented by a register. This allows to also model the register cost in the latter case (not treated here but it is a trivial extension of the objective). Hence, we introduce two new decision variables for each w and stage: a^s_w and r^s_w, which are true, if w in stage s is realized using an adder or register/wire, respectively.

ILP Formulation 2 Multiplierless FIR filters with depth limit

$$\text{minimize} \underbrace{\sum_{s=1}^{S} \sum_{w \in \mathcal{A}^s} a_w^s}_{=A_\text{M}} + A_\text{S}(h_{m,0})$$

subject to

$$\text{C1a:} \quad G2^B \underline{D}(\omega) \leqslant \sum_{m=0}^{M-1} h'_m c_m(\omega) \leqslant G2^B \overline{D}(\omega), \ \forall \omega \in \Omega_d$$

$$\text{C1b:} \quad \underline{h_m} \leqslant h'_m \leqslant \overline{h_m}, \ \forall m = 0, \ldots, M-1$$

$$\text{C2:} \quad \underline{G} \leqslant G \leqslant \overline{G}$$

$$\text{C3a:} \quad h'_m = \begin{cases} \sum_{w=0}^{2^B-1} w h_{m,w} & \text{if } \phi_m = 0 \\ -\sum_{w=1}^{2^B-1} w h_{m,w} & \text{if } \phi_m = 1 \end{cases} \quad \forall m = 0, \ldots, M-1$$

$$\text{C3b:} \quad \sum_{w=0}^{2^B-1} h_{m,w} = 1, \ \forall m = 0, \ldots, M-1$$

$$\text{C4:} \quad r^S_{\text{odd}(w)} + a^S_{\text{odd}(w)} \geqslant \frac{1}{M} \sum_{m=0}^{M-1} h_{m,w}, \ \forall w = 0, \ldots, 2^B - 1$$

$$\text{C5:} \quad r_w^s = 0 \ \forall w \in \mathcal{A}^s \setminus \bigcup_{s'=0}^{s-1} \mathcal{A}^{s'} \text{ with } s = 1, \ldots, S-1$$

$$\text{C6:} \quad r_w^s - a_w^{s-1} - r_w^{s-1} \leqslant 0, \ \forall w \in \mathcal{A}^s \setminus \{0\}, \ s = 2, \ldots, S$$

$$\text{C7:} \quad a_w^s - \sum_{(u,v,w') \in \mathcal{T}^s \mid w'=w} x^{s-1}_{(u,v)} \leqslant 0 \quad \forall w \in \mathcal{A}^s, s = 2, \ldots, S$$

$$\text{C8:} \quad \begin{aligned} x^s_{(u,v)} - r_u^s - a_u^s &\leqslant 0 \\ x^s_{(u,v)} - r_v^s - a_v^s &\leqslant 0 \\ \forall (u,v,w) \in \mathcal{T}^s \text{ with } s &= 1, \ldots, S-1 \end{aligned}$$

The connection to the filter coefficients $h_{m,w}$ is made through C4. As several of the M h_m coefficients can have the same w value, the right hand side of C4 is scaled by $1/M$ to keep it less than one. Whenever the right hand side of C4 is non-zero it forces the realization of coefficient w in the output stage S, either as an adder or as a register/wire.

Constraints C5 and C6 consider the realization as register/wire: they require that a value w can only be replicated from a previous stage if it was computed or replicated before.

The realization as an adder computing constant w from the inputs u and v requires the presence of both inputs in the previous stage. For that, the binary variables $x^s_{(u,v)}$ are introduced which determine if both are available in stage s:

$$x^s_{(u,v)} = \begin{cases} 1 & \text{if both } u \text{ and } v \text{ are available in stage } s \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

Now, constraint C7 specifies that if w is computed by $w = \mathcal{A}(u,v)$ in stage s, the pair (u,v) has to be available in the previous stage. If a pair (u,v) is required in stage s, constraints C8 make sure that u and v have been realized in the previous stage either as register or adder.

TABLE IV: Used constants (top) and variables (bottom) in ILP Formulation 2

Constant/Variable	Meaning
$M \in \mathbb{N}$	Number of filter coefficients
$\mathcal{A}^s \subseteq \mathbb{N}$	Coefficients that can be computed in adder stage s
$\mathcal{T}^s \subseteq \mathbb{N}^3$	Tripplets (u,v,w) from which $w \in \mathcal{A}^s$ can be computed using $u, v \in \mathcal{A}^{s-1}$
$\underline{h_m}, \overline{h_m} \in \mathbb{Z}$	Lower and upper bound for filter coefficient $m = 0, \ldots, M-1$
$h'_m \in \mathbb{Z}$	Value of filter coefficient $m = 0, \ldots, M-1$
$h_{m,w} \in \{0,1\}$	true, if $\lvert h'_m \rvert = w$ for $m = 0, \ldots, M-1$ and $w = 0, \ldots, 2^B - 1$
$\phi_m \in \{0,1\}$	true, if h'_m is negative
$a^s_w \in \{0,1\}$	true, if $w \in \mathcal{A}^s$ in stage $s = 1, \ldots, S-1$ is realized using an adder
$r^s_w \in \{0,1\}$	true, if $w \in \mathcal{A}^s$ in stage $s = 1, \ldots, S-1$ is realized using a register or wire
$x^s_{(u,v)} \in \{0,1\}$	true, if u and v are available in stage $s = 1, \ldots, S-2$
$G \in [\underline{G}, \overline{G}]$	Gain of a variable gain filter ($G = 1$ when the gain is fixed)

Note that instead of using constraint C5 it is more practical to remove all variables r^s_w which are zero from the cost function and their related constraints. Also note that the binary variables $x^s_{(u,v)}$ and the integer variables h_m can be relaxed to real numbers to speed up the optimization.

Finally, this model can be extended to optimize an approximate low-level cost. For that, the A_S and a^s_w terms have to be weighted with their corresponding low-level cost. While this is exact for the multiplier blocks (as their coefficients are pre-computed) the costs for the structural adders have to be approximated. This can be done by taking the input word size for computing the cost of each structural adder and considering their word size increase due to the multiplier block in the a^s_w terms. This works, of course, only when the structural adder is actually realized.

C. Selecting the Adder Depth

The AD is often selected to be as small as possible, typically at the expense of a higher adder cost. It is well known that the minimal AD needed when multiplying with a given coefficient can be realized by using a binary tree [48]. Therefore, it cannot be lower than the base two logarithm of the non-zero digit count of its MSD representation. Unfortunately, as the coefficients are not known in advance, the minimum AD cannot be derived from the filter specification. However, the upper bound of the AD can be computed from the coefficient word size B as follows. A B bit binary number can have up to $B+1$ digits when represented as an MSD number and up to $\lfloor (B+1)/2 \rfloor + 1$ non-zeros in the worst case [40]. This leads to a maximum adder depth of

$$\text{AD}_\text{max} = \log_2 \left(\left\lfloor \frac{B+1}{2} \right\rfloor + 1 \right) . \tag{18}$$

Using this bound, a search from $\text{AD} = 0, \ldots, \text{AD}_\text{max}$ can be performed until the first feasible solution is found.

For practical FIR filters, early studies have shown that coefficient word sizes between 15 bit to 20 bit are sufficient to achieve approximation errors between -70 and -100 dB. Using (18), this translates to ADs of at most three to four. In our experiments, we found very good solutions with AD $= 2$ for most of the filters from practice.

V. Reducing the Problem Complexity

A. Reducing the Coefficient Range

Following [23, Sec. 3], we bound the search space for the gain and coefficient values, respectively, by projecting the polytope corresponding to the discretized versions of (4) or (6) onto G and each h'_m. For instance, in the case of the coefficients, the goal is a tight interval enclosure $[\underline{h_m}, \overline{h_m}]$ for the feasible values of h'_m. This corresponds to the LPs:

$$\text{minimize } h'_m$$

or

$$\text{maximize } h'_m$$

subject to

$$G\underline{D}(\omega) \leqslant \sum_{k=0}^{M-1} h'_k c_k(\omega) \leqslant G\overline{D}(\omega), \qquad \forall \omega \in \Omega_d,$$

where $h'_k \in \mathbb{R}$ for $k = 0, \ldots, M-1$ and $G \in \left[\underline{G}, \overline{G}\right]$ (or $G = 1$ when unity gain is used). We get $[\underline{h_m}, \overline{h_m}]$ by taking

$$\underline{h_m} = \lceil h'_m \rceil \text{ from minimize } h'_m,$$
$$\overline{h_m} = \lfloor h'_m \rfloor \text{ from maximize } h'_m.$$

We do the same for G in computing the enclosure $[\underline{G}, \overline{G}]$, which is done before bounding the filter coefficients.

B. Discretizing the Frequency Domain

Even though Ω is replaced by a finite set Ω_d, we perform a rigorous posteriori validation of the result over Ω [49]. Still, the typically large size of Ω_d ($16M$ is a common value found in the literature) can have a big impact on the runtime of the filter design routine. This is shown for instance in the context of an optimal branch-and-bound algorithm for FIR filter design with fixed-point coefficients [34, Table 2]. A too small number of points can, on the other hand, lead to an invalid solution over Ω and a larger feasible set, potentially incurring a larger runtime as well.

It is thus important to consider a discretization of reasonable size that is unlikely to lead to invalid solutions over Ω (*i.e.,* equations (4) or (6) do not hold) and does not increase the search space by a too large factor. To this effect, we use so-called approximate Fekete points (AFPs), which contain the most critical frequencies for a given filter that needs to fit a target frequency response. They have recently been used to improve the robustness of the classic Parks-McClellan Chebyshev FIR filter design algorithm [50] and for a fast and efficient heuristic for FIR fixed-point coefficient optimization [51]. They are efficient choices when performing polynomial interpolation/approximation on domains such as Ω. This is relevant in our context since $H_R(\omega)$ in (2) is a polynomial in $\cos(\omega)$. For details on how to compute them we refer the reader to [50], [51] and the references therein.

C. An Adaptive Search Strategy

Even if the current Ω_d leads to a solution that does not pass a posteriori validation, it might still be possible to rescale the gain factor G such that (6) holds. By taking a point $\omega_{\max} \in \Omega$ where $G\underline{D}(\omega_{\max}) - H_R(\omega_{\max})$ or $H_R(\omega_{\max}) - G\overline{D}(\omega_{\max})$ is largest (*i.e.,* the point of largest deviation from the specification) we first update G to take a value close to $H_R(\omega_{\max})/\underline{D}(\omega_{\max})$ or $H_R(\omega_{\max})/\overline{D}(\omega_{\max})$, depending on where the deviation occurs. If this new gain leads to a valid solution over Ω, then it is optimal. If not, we update Ω_d by adding the points of largest deviation for each frequency subdomain. We rerun the optimization with this new Ω_d, repeating until either (a) there are no more invalid frequency points or (b) the problem becomes infeasible, meaning no solution with the imposed constraints over Ω exists.

We should mention that running the result validation code of [49] at each iteration of the adaptive routine is computationally expensive. This is why at each iteration we perform a fast, non rigurous test consisting of verifying (6) on a much denser discretization of Ω than Ω_d. We found this to usually be sufficient in ensuring that the a posteriori validation [49] done at the end of optimization is successful. This is in stark contrast with the rest of the literature, which generally only uses a small discretization of Ω throughout the design process. While good for performance, such an approach will sometimes lead to designs which actually fail to satisfy the specification (see results in Table VI).

VI. Experimental Results

To test the ILP formulations discussed above, we have implemented them in a C++ filter design tool[1]. It features a flexible command-line interface.

A. Experimental Setup and Parameter Choices

All experiments were run on a Linux machine with an Intel Xeon E5-2690 v4 CPU with 56 cores and 252 GB of RAM. The proposed implementation supports several popular open source and commercial (M)ILP solvers, such as SCIP[2], Gurobi[3] and CPLEX[4][5]. For convenience, these solvers are accessed through the ScaLP [52] library, which acts as a frontend. Based on our experiments, Gurobi usually proved to be the fastest backend, which is why, apart from a few exceptions, use it on all the examples below.

All experiments use the AFP-based frequency grid discretization mentioned in Section V-B. As discussed before, the number of frequency points in Ω_d is run-time critical. To determine an appropriate size, we ran an experiment using a typical design scenario with an Ω_d size of kM points and $k = 1, \ldots, 32$. They start large for very low k, as in these cases the frequency grid usually has to be extended to address violations, which require re-running the optimization routine on a new grid. If k is large enough (*e.g.* $k \geqslant 4$), invalid results

[1] Available as an open-source project at: https://gitlab.com/filteropt/firopt.
[2] https://scipopt.org
[3] https://www.gurobi.com
[4] https://www.ibm.com/analytics/cplex-optimizer
[5] Free academic licenses for Gurobi 8.1 and CPLEX 12.6 are used.

TABLE V: Specifications of the reference filters

Name	Source	Ω_p/π	Ω_s/π	δ_p	δ_s
S1	[4]	$[0, 0.3]$	$[0.5, 1]$	0.00636	0.00636
S2	[9], [14], [23]	$[0, 0.042]$	$[0.14, 1]$	0.026	0.001
L1	[9], [11], [23]	$[0.8, 1]$	$[0, 0.74]$	0.0057	0.0001
L2	[1], [16], [23]	$[0, 0.2]$	$[0.28, 1]$	0.02800	0.001
L3	[1], [16]	$[0; 0.15]$ $[0.15; 0.1875]$ $[0.1875; 0.2125]$	$[0.2875; 1]$	0.0165 0.0296 0.0546	0.0316
X1	[10], [14], [23]	$[0, 0.2]$	$[0.8, 1]$	0.0001	0.0001
G1	[6], [14], [23]	$[0, 0.2]$	$[0.5, 1]$	0.01	0.01
Y1	[14], [23]	$[0, 0.3]$	$[0.5, 1]$	0.00316	0.00316
Y2	[14], [23]	$[0, 0.3]$	$[0.5, 1]$	0.00115	0.00115

become rare, meaning just one optimization pass is sufficient. Further increasing k at this point just leads to more constraints in the model and likely a larger runtime for the optimizer. Based on these results, we start with $4M$ points. This choice usually delivers a good balance between optimizer runtime and number of iterations needed to obtain a valid solution over Ω.

B. Benchmark Set

Several multiplierless filter designs were computed to evaluate our methods. They are introduced next.

1) A Family of Specifications from [4, Example 1]: We consider a family of low-pass linear-phase filter specifications from Redmill et al. [4]. These specifications are defined by:

$$\begin{aligned} 1-\delta \leqslant H_R(\omega) \leqslant 1+\delta, \quad & \omega \in [0, 0.3] \quad \text{(passband)} \\ -\delta \leqslant H_R(\omega) \leqslant \delta, \quad & \omega \in [0.5, 0.1] \quad \text{(stopband)} \end{aligned}$$

where δ is a parameter regulating error. We set $\delta = 10^{-\frac{p}{20}}$, where $p > 0$ is the error in decibels (dB).

Our goal with this benchmark is to explore the tradeoff between the error (p), the filter order (N) and the word length (B) in terms of the total number of adders.

2) A Set of State-of-the-art Specifications: We also test our tool on a set of reference specifications from the literature [1], [2], [4], [6], [9], [10], [14], [16], [23], referred to as $S1$, $S2$, $L1$, $L2$, $L3$, $X1$, $G1$, $Y1$ and $Y2$. They are all low-pass filters defined by

$$\begin{aligned} 1-\delta_p \leqslant H_R(\omega) \leqslant 1+\delta_p, \quad & \omega \in \Omega_p \quad \text{(passband)}, \\ -\delta_s \leqslant H_R(\omega) \leqslant \delta_s, \quad & \omega \in \Omega_s \quad \text{(stopband)}, \end{aligned}$$

where the values of $\delta_p, \delta_s, \Omega_p, \Omega_s$ for each specification are given in Table V. Over time, some of these reference filter specifications were slightly modified by different publications. We compare against the most recent version in each case, updating it to account for any violation of the specification in the results reported in the literature.

The restriction to low-pass filters comes only from the existing literature. Our tool can also be used to design other types of filters, such as multiband filters or decimators (since we generalize constraints on the frequency response as *functions* of frequencies).

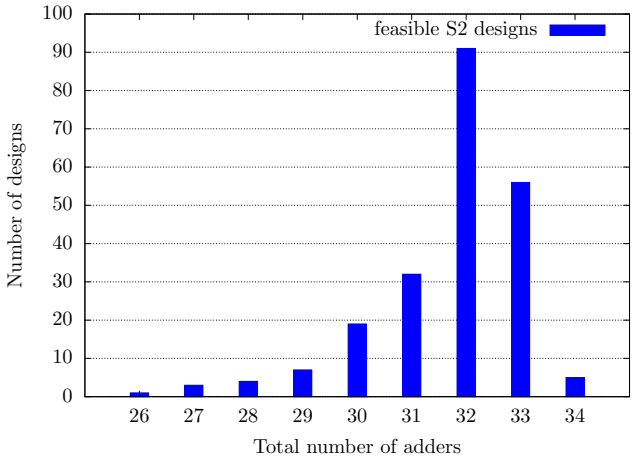


Fig. 4: Histogram demonstrating the total adder cost of feasible filter designs for S1 specification with wordlength $B = 9$.

C. Results

1) 1-step vs 2-step design optimization approach: We start by demonstrating the advantages of an overall 1-step optimization over the classic 2-step filter design process. For each frequency specification, there exist numerous coefficient sets satisfying it. Take, for example, frequency specification S1, when realized as type-I filter with $N = 24$: there exist 237 sets of coefficients of word length $B = 9$ that satisfy the constraints. We brute-force explored the design space and applied the optimal MCM solver [30] on each possible filter coefficient set to design the multiplierless implementations. For this particular example, the total adder cost varies from 26 up to 34 adders. Figure 4 presents the histogram illustrating the number of coefficient sets falling into each adder cost category. It can be interpreted in the following way: when selecting coefficient sets out of the pool of feasible results, with high probability one obtains the design of cost 32, 33 or perhaps 31 adders. Indeed, only 15 out of 237 filters have cost less than 30 adders and the lowest cost is achieved by only one filter. Generally speaking, for an arbitrary filter specification, the two-step approach that first selects a filter coefficient set (without a priori knowledge of the cost) and then designs an optimal MCM architecture, has a high probability to be far from optimal. Brute-force design exploration as presented here is not practical for higher-order and high-wordlength solutions, hence a one-step optimization approach is necessary to navigate the search towards an optimal solution.

2) ILP Formulation 1 vs. ILP Formulation 2: In the first experiment, we compare ILP Formulation 1 (in its overall form discussed in Section III-B) to ILP Formulation 2. Both models can be used to optimize for the total number of adders (MB and structural) given fixed parameters like filter order N, filter type and the effective word length B (see Fig. 3). In case of ILP2, the adder depth is an additional constraint. Therefore, in practice, the two approaches can sometimes lead to different results.

This is exemplified in Fig. 5, where we design a set of filters using the family of specifications from [4, Example 1]

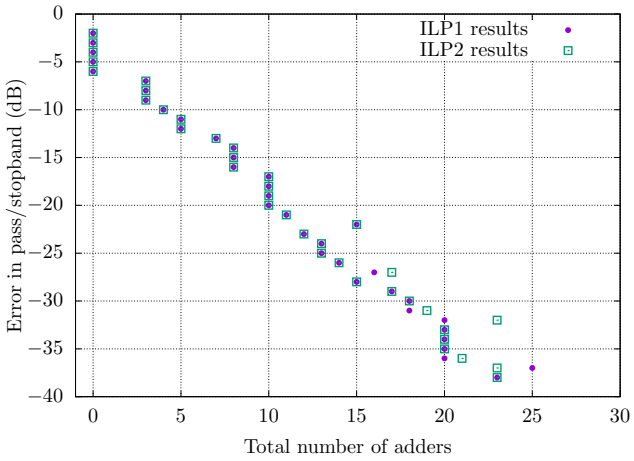


Fig. 5: Total adder count comparison when using ILP1 (adapted to minimize the total number of adders) and ILP2 (with AD limit set to 2) on the Redmill set of filters [4].

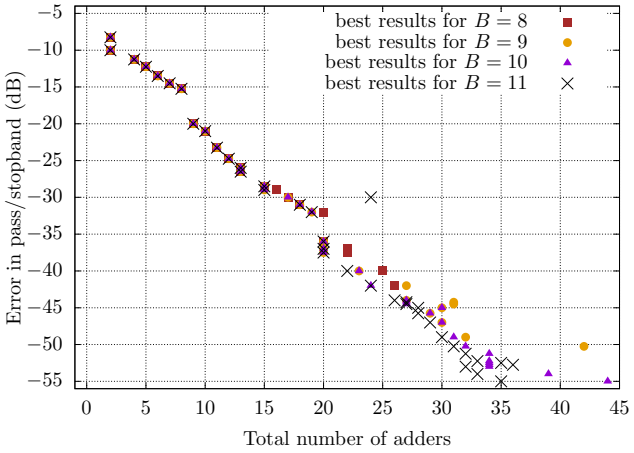


Fig. 6: Our designs with effective word lengths varying from 8 to 11, filter type I and smallest feasible filter order.

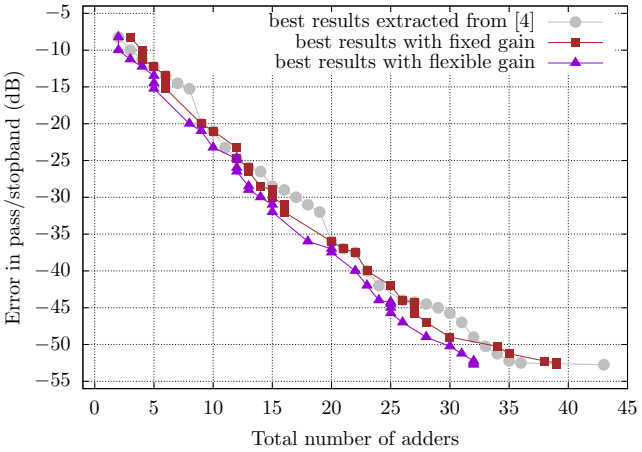


Fig. 7: Comparison between our best design space exploration results and the best results from [4, Example 1]. Our tool improves designs from [4] or proves them optimal.

as described in Section VI-B1. We consider 2×37 filters corresponding to $p = 2, \ldots, 38$ (*i.e.,* error is varied from -2 dB to -38 dB), with a 9-bit effective word length and fixed gain $G = 1$. In each case, a type I filter with smallest N that leads to a feasible solution under the given constraints was used. For ILP2, the upper bound on the AD is set to 2 as this turned out to be sufficient for all test instances.

For most error targets the resulting total adder count is identical between the two. The exceptions are $-27, -31, -32$ and -36 dB, where ILP1 gives a better total adder count, and -37 dB, where ILP2 is better. The four cases where ILP1 gives a better result are not surprising considering that the limited AD in ILP2 restricts the coefficient search space. For -37 dB, the difference comes from the fact that the ILP1 solver is able to find an optimal solution with $N = 20$, while an AD $= 2$ solution for ILP2 is only possible starting with $N = 22$. Taking AD $= 3$ with ILP2 gives the $N = 20$ solution found with ILP1.

Due to the different nature of the constraints and objective values of both ILP models, it is hard to do a runtime comparison between the two. We only mention that when there is a feasible filter with small number of MB adders, ILP1 can be quite fast for moderate size problems ($N < 50$), but turns out to struggle for problems with many MB adders. Similarly, for feasible designs with small AD (up to 3), ILP2 will be fast for $N < 50$ and overall scales better than ILP1.

In the rest of the paper, for comparison with previous work, we use ILP Formulation 2 with AD $= 2$ (unless otherwise stated).

3) Design Space Exploration for [4, Example 1]: The experiment setting from Section VI-C2 is expanded upon. We compare our best results (with effective word lengths $B \in \{8, 9, 10, 11\}$) with those from [4, Example 1]. We start off by considering only type I filters (just like in [4]), variable gain $G \in [2/3, 4/3]$ and minimal order N for each error target. The results are illustrated in Fig. 6. We note that there are certain cases where, for a given B, taking the minimal filter order leading to a feasible solution *does not* minimize the adder cost. This is most visible for $B = 11$ and a -30dB error target, where a minimal order $N = 14$ filter requires 24 adders. For $B = 10$, the minimal N is 16, leading to only 17 total adders, a 7 adder improvement. Taking $N = 16$ for $B = 11$ also results in a 17 adder solution. A lower implementation cost is sometimes possible when increasing the filter order leads to a sparser filter and/or a more economical MCM design. Such solutions better optimize the objective functions in the proposed ILP models. We nevertheless remark that increasing the filter order beyond a certain threshold will not lead to different solutions.

Of course, increasing the word length can also lead to a significant improvement in the results. For instance, the optimal -50dB atteanuation results for $B \in \{9, 10, 11\}$ require $41, 32$ and 31 adders, respectively.

This nonlinearity of the word length/cost relation means that the user should favor a comprehensive exploration of the design space, varying the design parameters (especially B, filter type and N) and examine the various trade-offs. This

TABLE VI: Comparison between our method and the state-of-the-art results for the specifications in Table V.

Name	Source	N	Type	A_{M}	A_{S}	$\boldsymbol{A}$	AD	G	B	Coefficients
S1	[4]	24	I	6	20	**26**	3	2.4570	9	2 8 0 −16 −14 20 43 0 −80 −71 112 377 502
S1	ours	23	II	5	19	**24**	2	2.46492	9	6 6 −8 −21 0 36 32 −42 −96 0 248 472
S2	[14]	59	II	17	59	**76**	3	10.47032	10	5 5 6 5 2 −2 −10 −20 −32 −48 −64 −78 −92 −98 −87 −65 −26 26 93 174 267 368 472 575 672 757 826 874 898
S2	ours	59	II	15	51	**66**	2	7.5904	10	0 0 0 −2 −5 −10 −16 −23 −32 −40 −50 −58 −64 −64 −61 −50 −29 0 38 86 143 206 274 344 412 476 532 576 608 624
L2	[9]	62	I	17	62	**79**	3	2.6668	11	4 9 13 12 4 −10 −26 −36 −32 −12 18 44 52 32 −10 −56 −80 −64 −4 74 130 128 48 −86 −215 −263 −168 88 460 854 1153 1265
L2	ours	62	I	16	62	**78**	3	2.6668	11	4 9 13 12 4 −10 −26 −36 −32 −12 18 44 52 32 −10 −56 −80 −64 −4 74 130 128 48 −86 −215 −263 −168 88 460 854 1153 1265
L3	[14]	35	II	4	31	**35**	1	2.627	7	3 0 −2 −5 −5 0 3 7 8 3 −6 −14 −16 −7 12 40 65 80
L3	ours	35	II	3	31	**34**	2	2.60564	7	4 0 −2 −4 −4 0 3 8 8 3 −6 −13 −16 −8 13 40 64 80
X1	[23], ours	14	I	5	8	**13**	2	1.6404	10	−4 0 28 0 −113 0 509 840
G1	[23], ours	15	II	2	15	**17**	2	2.6338	6	1 2 −1 −7 −7 7 34 56
Y1	[23], ours	29	II	6	23	**29**	2	2.505	9	−1 −4 0 8 8 −10 −22 0 40 33 −44 −99 0 254 479
Y2	[23]	29	II	9	29	**38**	3	2.6361	10	−1 0 4 4 −5 −13 0 24 20 −26 −55 0 91 73 −96 −213 0 536 1008
Y2	ours	29	II	9	29	**38**	3	2.6259	10	−1 0 4 4 −5 −13 0 24 20 −26 −55 0 91 73 −96 −212 0 534 1004

is possible with our tool. Fig. 7 shows the results of such an experiment where, with respect to the setting of Fig. 6, we additionally allow N to vary and also consider type II filters. We also added the flexible gain, genetic algorithm results produced in [4]. Compared to [4], we could improve all of the results, except two of them where we obtained the same adder cost (-9 dB and -25 dB). It is clearly visible that allowing variable gain designs can have a major influence on the quality of the results.

4) The Set of State-of-the-art Specifications: Table VI presents the comparison between our designs and the best results from literature [1], [2], [4], [9], [14], [16] for the filter specifications from Table V. The following information for each implementation is given: filter order (N), filter type, number of multiplier adders (A_M), number of structural adders (A_S), total number of adders (A), adder depth of the multiplier block (AD), gain (G), effective coefficient word length (B) and, finally, coefficients of the filter. In the following, we discuss each instance in detail.

S1: for this specification we show that the AD can be reduced from 3 to 2 stages, while keeping the same effective word length. We also reduce the number of adders by two (one structural and one MB).

S2: in [9], implementations with adder depths 3 and 2 are proposed, at the cost of 78 and 80 adders, respectively. These results are improved in [14], with the authors claiming that a 3-stage implementation at the cost of 76 adders has a high probability to be optimal. We demonstrate that a 2-stage design with a cost of only 66 adders is possible. Again, our result has higher sparsity than previous designs.

L1: the tool timed out for this specification before giving a feasible result, hence it is not presented in Table VI. The best known result from literature is a 120-tap filter [9] and the size of an instance of the corresponding ILP formulation goes beyond the current capabilities of the solvers we tried, showing its limitations.

L2: the result of [9] can be improved by one adder with the same coefficient set. Similar to *L1*, we had to timeout before the solver could ascertain if the feasible filter obtained is indeed optimal.

L3: in [14], an implementation with 35 adders is provided. We show that 34 adder solutions are possible.

X1,G1,&Y1: we obtain the same results as the optimal ones reported in [23].

Y2: we obtain a result with same adder count as the optimal one reported in [23].

Overall, the proposed tool achieves improvements to the majority of the considered filter design problems. Moreover, the user can explore a large design space by setting different implementation parameters, *e.g.* adder depth, coefficient word length, filter type, etc. The required runtime however, will depend greatly on the problem. In the case of Table VI, it varied from several seconds for the smallest filters (S1) up to several days for the largest ones (S2 and L2).

VII. Conclusion and Perspectives

In this paper we have introduced two new algorithms for the design of optimal multiplierless FIR filters. Relying on ILP formulations stemming from the MCM literature, our algorithms minimize either (a) the number of structural adders given a fixed budget of multiplier block adders or (b) the total

number of adders (multiplier block + structural adders) given a fixed adder depth. We further show how (a) can be applied iteratively to optimally minimize the total number of adders (without any adder count or adder depth constraints). Extensive numerical tests with example design problems from the state-of-the-art show that our approaches can offer in many cases better results. We also make available an open-source C++ implementation of the proposed methods.

In the future we plan to extend the current solution to other filter structures and more efficiency metrics, in particular to optimize the number of full adders. Regarding new metrics, this calls for modeling the impact of the precision choice on the numerical quality of the result, which is non-trivial due to its highly non-linear nature. Developing a dedicated branch and bound solver for instance, inspired by [23], has the potential of helping us deal with this heterogenous design problem. Regarding filter structures, we plan to adapt our framework to cascaded forms and recursive filters.

REFERENCES

[1] Y. Lim and S. Parker, "FIR filter design over a discrete powers-of-two coefficient space," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 31, no. 3, pp. 583–591, 1983.

[2] H. Samueli, "An improved search algorithm for the design of multiplierless FIR filters with powers-of-two coefficients," *IEEE Transactions on Circuits and Systems*, vol. 36, no. 7, pp. 1044–1047, Jul. 1989.

[3] C. Chen, "A trellis search algorithm for the design of FIR filters with signed-powers-of-two coefficients," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 1999.

[4] D. Redmill, D. Bull, and E. Dagless, "Genetic synthesis of reduced complexity filters and filter banks using primitive operator directed graphs," *Circuits, Devices and Systems, IEE Proceedings* -, vol. 147, no. 5, pp. 303–310, 2000.

[5] J. Yli-Kaakinen and T. Saramaki, "A systematic algorithm for the design of multiplierless FIR filters," *IEEE International Symposium on Circuits and Systems*, vol. 2, pp. 185–188, 2001.

[6] O. Gustafsson and L. Wanhammar, "Design of linear-phase FIR filters combining subexpression sharing with MILP," in *Midwest Symposium on Circuits and Systems*. IEEE, Aug. 2002, pp. III–9–III–12.

[7] A. P. Vinod and E.-K. Lai, "On the implementation of efficient channel filters for wideband receivers by optimizing common subexpression elimination methods," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 2, pp. 295–304, 2005.

[8] L. Aksoy, E. Costa, P. Flores, and J. Monteiro, "Optimization of Area Under a Delay Constraint in Digital Filter Synthesis Using SAT-Based Integer Linear Programming," *ACM/IEEE Design Automation Conference (DAC)*, pp. 669–674, 2006.

[9] Y. J. Yu and Y. C. Lim, "Design of Linear Phase FIR Filters in Subexpression Space Using Mixed Integer Linear Programming," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 54, no. 10, pp. 2330–2338, 2007.

[10] F. Xu, C. H. Chang, and C. C. Jong, "Design of Low-Complexity FIR Filters Based on Signed-Powers-of-Two Coefficients With Reusable Common Subexpressions," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, no. 10, pp. 1898–1907, 2007.

[11] M. Aktan, A. Yurdakul, and G. Dundar, "An algorithm for the design of low-power hardware-efficient FIR filters," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 55, no. 6, pp. 1536–1545, 2008.

[12] L. Aksoy, "Optimization Algorithms for the Multiple Constant Multiplications Problem," Ph.D. dissertation, Istanbul Technical University, 2009.

[13] Y. Yu and Y. Lim, "Optimization of Linear Phase FIR Filters in Dynamically Expanding Subexpression Space," *Circuits, Systems, and Signal Processing*, vol. 29, no. 1, pp. 1–16, 2009.

[14] D. Shi and Y. J. Yu, "Design of Linear Phase FIR Filters With High Probability of Achieving Minimum Number of Adders," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 58, no. 1, pp. 126–136, 2011.

[15] L. Aksoy, E. Costa, P. Flores, 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 (GLSVLSI)*. ACM, 2011, pp. 79–84.

[16] A. Shahein, Q. Zhang, N. Lotze, and Y. Manoli, "A Novel Hybrid Monotonic Local Search Algorithm for FIR Filter Coefficients Optimization," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 59, no. 3, pp. 616–627, 2012.

[17] W. Bin Ye and Y. J. Yu, "A polynomial-time algorithm for the design of multiplierless linear-phase FIR filters with low hardware cost," *IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 970–973, 2014.

[18] W. B. Ye and Y. J. Yu, "Bit-level multiplierless FIR filter optimization incorporating sparse filter technique," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 61, no. 11, pp. 3206–3215, 2014.

[19] A. Mehrnia and A. N. Willson, "Optimal Factoring of FIR Filters," *IEEE Transactions on Signal Processing*, vol. 63, no. 3, pp. 647–661, Feb. 2015.

[20] W. B. Ye, X. Lou, and Y. J. Yu, "Design of Low Power Multiplierless Linear-Phase FIR Filters," *IEEE Access*, 2017.

[21] X. Lou, P. K. Meher, Y. Yu, and W. Ye, "Novel Structure for Area-Efficient Implementation of FIR Filters," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 64, no. 10, pp. 1212–1216, 2017.

[22] L. Aksoy, P. Flores, and J. Monteiro, "A Tutorial on Multiplierless Design of FIR Filters: Algorithms and Architectures," *Circuits, Systems, and Signal Processing*, vol. 33, no. 6, pp. 1689–1719, Jan. 2014.

[23] ——, "Exact and Approximate Algorithms for the Filter Design Optimization Problem," *IEEE Transactions on Signal Processing*, vol. 63, no. 1, pp. 142–154, 2015.

[24] P. Cappello and K. Steiglitz, "Some Complexity Issues in Digital Signal Processing," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 32, no. 5, pp. 1037–1041, Oct. 1984.

[25] A. Dempster and M. Macleod, "Use of Minimum-Adder Multiplier Blocks in FIR Digital Filters," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 42, no. 9, pp. 569–577, 1995.

[26] Y. Voronenko and M. Püschel, "Multiplierless Multiple Constant Multiplication," *ACM Transactions on Algorithms*, vol. 3, no. 2, pp. 1–38, 2007.

[27] O. Gustafsson, "A Difference Based Adder Graph Heuristic for Multiple Constant Multiplication Problems," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2007, pp. 1097–1100.

[28] ——, "Towards Optimal Multiple Constant Multiplication: A Hypergraph Approach," in *Asilomar Conference on Signals, Systems and Computers (ACSSC)*. IEEE, Oct. 2008, pp. 1805–1809.

[29] L. Aksoy, E. Günes, and P. Flores, "Search Algorithms for the Multiple Constant Multiplications Problem: Exact and Approximate," *Microprocessors and Microsystems*, vol. 34, no. 5, pp. 151–162, 2010.

[30] M. Kumm, "Optimal Constant Multiplication using Integer Linear Programming," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 65, no. 5, pp. 567–571, 2018.

[31] R. E. Crochiere and A. V. Oppenheim, "Analysis of Linear Digital Networks," in *Proceedings of the IEEE*, 1975, pp. 581–595.

[32] D. M. Kodek, "Design of optimal finite wordlength FIR digital filters using integer programming techniques," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 28, no. 3, pp. 304–308, 1980.

[33] Y. Lim, "Design of discrete-coefficient-value linear phase FIR filters with optimum normalized peak ripple magnitude," *IEEE Transactions on Circuits and Systems*, vol. 37, no. 12, pp. 1480–1486, 1990.

[34] D. M. Kodek, "Design of optimal finite wordlength FIR digital filters," in *Proc. of the European Conference on Circuit Theory and Design, ECCTD '99.*, vol. 1, Aug. 1999, pp. 401–404.

[35] S. Demirsoy, A. Dempster, and I. Kale, "Transition Analysis on FPGA for Multiplier-Block Based FIR Filter Structures," *IEEE International Symposium of Circuits and Systems (ISCAS)*, vol. 2, pp. 862–865 vol.2, 2000.

[36] A. G. Dempster, S. S. Demirsoy, and I. Kale, "Designing Multiplier Blocks with Low Logic Depth," in *IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2002, pp. V–773–V–776.

[37] M. Faust and C.-H. Chang, "Minimal Logic Depth Adder Tree Optimization for Multiple Constant Multiplication," *IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 457–460, 2010.

[38] M. Kumm, D. Fanghänel, K. Möller, P. Zipf, and U. Meyer-Baese, "FIR Filter Optimization for Video Processing on FPGAs," *Springer EURASIP Journal on Advances in Signal Processing*, pp. 1–18, 2013.

[39] M. Kumm, O. Gustafsson, M. Garrido, and P. Zipf, "Optimal Single Constant Multiplication using Ternary Adders," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 65, no. 7, pp. 928–932, 2018.

[40] M. Kumm, "Multiple Constant Multiplication Optimizations for Field Programmable Gate Arrays," Ph.D. dissertation, Springer Wiesbaden, Wiesbaden, Oct. 2015.

[41] W. Chen, M. Huang, and X. Lou, "Design of sparse fir filters with reduced effective length," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 4, pp. 1496–1506, 2018.

[42] W. Chen, M. Huang, W. Ye, and X. Lou, "Cascaded Form Sparse FIR Filter Design," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 5, pp. 1692–1703, 2020.

[43] H. Wang, X. Cheng, P. Song, L. Yu, W. Hu, and L. Zhao, "An extrapolated impulse response filter design with sparse coefficients based on a novel linear approximation of matrix," *Circuits, Systems, and Signal Processing*, vol. 34, no. 7, pp. 2335–2361, 2015.

[44] H. Wang, Z. Zhao, and L. Zhao, "Matrix decomposition based low-complexity fir filter: Further results," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 2, pp. 672–685, 2019.

[45] K. Johansson, O. Gustafsson, and L. Wanhammar, "A Detailed Complexity Model for Multiple Constant Multiplication and an Algorithm to Minimize the Complexity," *European Conference on Circuit Theory and Design*, vol. 3, pp. III/465–III/468 vol. 3, 2005.

[46] L. Aksoy, E. Costa, P. Flores, and J. Monteiro, "Optimization of Area in Digital FIR Filters using Gate-Level Metrics," in *ACM/IEEE Design Automation Conference (DAC)*, 2007, pp. 420–423.

[47] A. Dempster and M. D. Macleod, "Constant Integer Multiplication Using Minimum Adders," *IEE Proceedings of Circuits, Devices and Systems*, vol. 141, no. 5, pp. 407–413, 1994.

[48] H.-J. Kang and I.-C. Park, "FIR Filter Synthesis Algorithms for Minimizing the Delay and the Number of Adders," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 48, no. 8, pp. 770–777, 2001.

[49] A. Volkova, C. Lauter, and T. Hilaire, "Reliable verification of digital implemented filters against frequency specifications," in *2017 IEEE 24th Symposium on Computer Arithmetic (ARITH)*, July 2017, pp. 180–187.

[50] S.-I. Filip, "A robust and scalable implementation of the Parks-McClellan algorithm for designing FIR filters," *ACM Transactions on Mathematical Software (TOMS)*, vol. 43, no. 1, p. 7, 2016.

[51] N. Brisebarre, S.-I. Filip, and G. Hanrot, "A Lattice Basis Reduction Approach for the Design of Finite Wordlength FIR Filters," *IEEE Transactions on Signal Processing*, vol. 66, no. 10, pp. 2673–2684, May 2018.

[52] P. Sittel, T. Schönwälder, M. Kumm, and P. Zipf, "ScaLP: A Light-Weighted (MI)LP Library," in *Methoden und Beschreibungssprachen zur Modellierung und Verifikation von Schaltungen und Systemen (MBMV)*, 2018, pp. 1–10.

Martin Kumm received the Dipl.-Ing. degree in electrical engineering from the University of Applied Sciences Fulda, Germany, and the Technical University of Darmstadt, Germany, in 2003 and 2007, respectively. From 2003 to 2009, he was with GSI Darmstadt, working on digital RF control systems for particle accelerators. In 2015 he received his Ph.D. (Dr.-Ing.) degree from the University of Kassel, Germany. He is currently a Professor for Embedded Systems at the Fulda University of Applied Sciences, Germany. His research interests are arithmetic circuits and their optimization as well as high-level synthesis, all in the context of reconfigurable systems.

Anastasia Volkova obtained her Master's Degree in Applied Mathematics from Odessa National University, Ukraine, in 2014. She obtained a PhD in Computer Science from Sorbonne Université in Paris, France in 2017. She was a postdoctoral researcher at Inria, France and an AI research resident at Intel Corporation. In 2019 she joined University of Nantes, France, as an Associate professor. Her research interests include computer arithmetic, validated numerical computing and design of optimized software/hardware for Floating- and Fixed-Point algorithms.

Silviu-Ioan Filip received his bachlor's degree in computer engineering from the Technical University of Cluj-Napoca, Romania, in 2012. He received his Ph.D. in computer science from the École Normale Supérieure de Lyon, France, in 2016. From October 2016 to December 2017 he was a postdoctoral researcher in the Numerical Analysis group at the University of Oxford. He is currently an INRIA junior researcher at the IRISA research institute in Rennes, France. His research interests are in computer arithmetic, approximation theory, convex and integer optimization.