Hardware-aware Design of Multiplierless Second-Order IIR Filters with Minimum Adders

Rémi Garcia, Anastasia Volkova, Martin Kumm, Alexandre Goldsztejn, Jonas Kühle

Hardware-aware Design of Multiplierless Second-Order IIR Filters with Minimum Adders

Rémi Garcia, Anastasia Volkova, Martin Kumm, Alexandre Goldsztejn, Jonas Kühle

***Abstract*—In this work we optimally solve the problem of multiplierless design of second-order Infinite Impulse Response filters with minimum number of adders. Given a frequency specification, we design a stable direct form filter with hardware-aware fixed-point coefficients where all multiplications are replaced by bit shifts and additions. The coefficient design, quantization and implementation, typically conducted independently, are now gathered into one global optimization problem, modeled through integer linear programming and efficiently solved using generic solvers. The optimal filters are implemented within the FloPoCo IP core generator and synthesized for field programmable gate arrays (FPGAs) and application specific integrated circuits (ASICs). With respect to state-of-the-art three-step filter design methods, our one-step design approach achieves, on average, 48% reduction in number of lookup tables, 27% delay reduction and 57% reduction in power on FPGAs. ASICs experiment illustrate similar 48% reduction in circuit area, 27% delay reduction and 65% power reduction for a 14 nm ASIC.**

***Index Terms*—Digital filters, IIR, optimal design, multiplierless hardware, ILP**

I. Introduction

DIGITAL filters are essential components of modern technology, from medical equipment to scientific instruments. Filter design is a core topic in digital signal processing and control, and efficient filter implementation in software and hardware has received a significant research interest for half a century. Infinite Impulse Response (IIR) filters are a class of widely-used recursive linear time-invariant filters (see Fig. 1). IIR filters can be relatively easily designed in software, but hardware implementation is essential for embedded systems, where performance/power constraints are critical. While classic mobile or medical applications often rely on Application Specific Integrated Circuits (ASICs), other application domains, such as 5G/6G backbones and autonomous vehicles, rely on reconfigurable hardware using Field Programmable Gate Arrays (FPGA).

Classic Fixed-Point (FxP) filter design and implementation flow follows three separate steps:

1) **Filter Design (FD)** consists in finding real (in practice, double precision) filter coefficients, adhering to the given frequency specification. IIR filters are defined by coefficients of a rational transfer function, for which a stability criteria must be also satisfied. In general, a large amount of different filter coefficient sets can realize a given frequency specification;
2) **Quantization (Q)** converts the coefficients to a FxP format such that they still respect the given frequency response and they still lead to a stable filter;
3) **Implementation (I)** consists in generating, using quantized coefficients, a valid hardware description. This step exposes a high number of parameters, *e. g.*, the type of multipliers used.

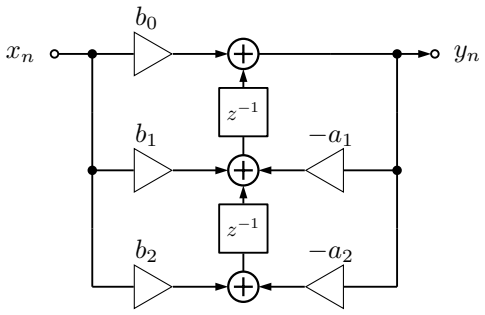


Fig. 1: Transposed Direct Form II for a second-order IIR filter.

The combination of FD and Q steps has been studied extensively since 1960s [1]–[3]. For certain structures of IIR filters it can even be considered solved [4], but with respect to a signal to quantization noise ratio, which is a probabilistic measure, not guaranteeing numerical safety. The general approach for quantization of a transfer function is nevertheless quite straightforward, passing through iterative increase of coefficient word length.

A large body of work exists for the I step. Hardware filters involve multiplications with constants, for which optimization techniques have been extensively explored. In the multiplierless *shift-and-add*-based methods, constant multiplications are replaced by additions, subtractions and bit shifts. First, simple digit-based methods, such as Canonical Signed Digit, were used, but later it was shown that graph-based approaches are more general and can achieve better, and even optimal, results [8], [40]. The associated optimization problem is known as the multiple constant multiplication (MCM) problem, for which heuristics [5]–[7] as well as optimal approaches exist [8]–[11]. Various cost functions are possible: the high-level ones count the number of adders required to perform all multiplications, and low-level ones counting the number of full adders. This problem has been successfully modeled as (Mixed) Integer Linear Programming (ILP) and solved using efficient solvers such as CPLEX or Gurobi. Another constant

R. Garcia, A. Volkova and A. Goldsztejn were with Nantes Université, Centrale Nantes, CNRS, LS2N, 44000 Nantes. Email: firstname.lastname@ls2n.fr

M. Kumm and J. Kühle were with Faculty of Applied Computer Science, Fulda University of Applied Sciences, Germany. Email: firstname.lastname@informatik.hs-fulda.de

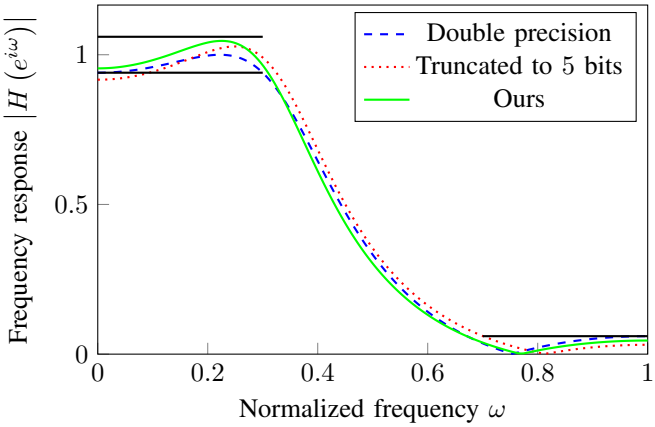


Fig. 2: Specification `lp14` and its reference Matlab, quantized to 5-bit and our 5-bit implementations.

multiplication method, especially relevant for FPGAs, is based on precomputed tables and is called Ken Chapman multiplier (KCM), after its inventor [12]. It has also been successfully applied to digital filtering [9], [13], [14] and using this method, an approach for optimization of combined Q & I steps has been proposed [15].

Both KCM and shift-and-add reduce arithmetic resources, by sharing intermediate results, and improve the delay of the circuit. As recursive filters are very problematic to pipeline [41], their multiplierless implementation is an attractive alternative to achieve a higher throughput without increasing the implementation cost. However, the resource reduction strongly depends on the coefficient values, which are typically fixed in the previous FD & Q steps. Hence, the obtained implementations are optimized only for one filter instance, or a small sub-set of the overall design space, not permitting overall optimal solution.

The most common model in design of IIR filters is based on the decomposition of a higher-order filter into a cascade of second-order sections implemented with the Direct Form structure [3], [16]–[19]. We focus here on the optimal design of individual second-order filters (see Fig. 1) as a building brick for methods using cascaded forms, *e. g.*, [20]. Our ambition is to **solve the combined FD & Q & I steps for a second-order IIR within one global optimization.** Even for one second-order filter this is a difficult non-linear combinatorial problem. Recent work [20] attempts to solve the IIR filter design and cascading problem, while targeting coefficients with the minimal binary writing, but does not lead to optimal solutions due to restricted design-space and a heuristic solver.

In contrast, in this paper we model the overall process as an ILP problem, which is consequently optimally solved using generic solvers. With this work, the whole design space for the FD & Q & I problem is covered: we search for FxP coefficients (*i. e.*, the quantization stage is implicit) that satisfy filter specifications while minimizing the cost of both structural and multiplier blocks in a multiplierless Transposed Direct Form II (tDFII) realization of a second-order IIR filter. Fig. 2 exhibits an example of filter specifications, for which the standard FD & Q design for 5-bit coefficients fails to satisfy the specification after truncation, whilst our approach, directly searching among 5-bit coefficients, finds a solution and it needs only 8 adders.

This paper presents a novel approach, implemented as an automated tool flow, for the simultaneous design of second-order IIR filters and their hardware implementation. The main contributions of this paper are:

- linearization of frequency constraints upon the rational transfer function. Based on ideas for linearization of FIR filters in [21] we formally rewrite the frequency constraints and to deal with inevitable nonlinearities we modify the integer product linearization technique [25];
- detection of ranges for all coefficients is indispensable to build an ILP model; we show that bounds on the transfer function's numerator depend on the target frequency specification and the corresponding optimization problem is quadratic convex, which is efficiently solved;
- as the IIR filter has two multiplier blocks, we propose a solution that ensures a simultaneous design of two independent multiplier blocks within one MCM model;
- we propose a novel search-space reduction method based on detection of symmetries in the solutions. We present and prove a symmetry breaking constraint that leaves the solved with only a quarter of the original design space;
- a new operator was developed within FloPoCo[1] [33], which is a state of the art code generator for hardware arithmetic units. Given the optimal filter coefficients computed with our ILP, FloPoCo now generates faithfully-rounded second-order IIR filters, *i. e.*, the internal data formats are automatically adapted such that the output is accurate up to its last bit.

The complete tool flow is available open-source[2]. We validate our approach on real-life and artificial examples by automatically generating and synthesizing VHDL code for FPGA and ASIC targets.

II. ILP Formulation for Second-Order IIR Filter Design

A. Problem Definition

The tDFII structure of second order digital filters is represented in Fig. 1. Its output is computed as

$$y_n = \sum_{k=0}^{2} b_k u_{n-k} - \sum_{k=1}^{2} a_k y_{n-k}, \tag{1}$$

where $a_k, b_k \in \mathbb{R}$ are the filter coefficients. The corresponding transfer function is

$$H(z) = \frac{B(z)}{A(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}, \tag{2}$$

with $z \in \mathbb{C}$. Designing a filter means finding the coefficients a_k and b_k such that the filter is stable and given frequency constraints hold. These constraints can be expressed as bounds on the filter frequency response

$$\underline{\beta}(\omega) \leq \left|H\left(e^{j\omega}\right)\right| \leq \overline{\beta}(\omega), \quad \forall \omega \in [0; \pi], \tag{3}$$

[1] http://flopoco.org/

[2] https://gitlab.univ-nantes.fr/volkova-a/jiir2hw

where $\underline{\beta}$ and $\overline{\beta}$ are the given frequency dependent lower and upper bounds. They typically encode constant bounds valid inside frequency intervals, but our definition (3) allows to model them as functions of ω.

There exists a large number of real-valued coefficient sets satisfying the frequency specifications (3) but only a subset of those is representable in FxP arithmetic with a given word length. In our approach, we directly search for a_k and b_k in a FxP representation and introduce their integer counterparts, $a'_k, b'_k \in \mathbb{Z}$, that are linked with the real-valued coefficients via

$$a_k = 2^{-w+1}\, 2^{g_a}\, a'_k, \qquad a'_k \in \left[-2^{w-1}; 2^{w-1}-1\right], \tag{4}$$

$$b_k = 2^{-w+1}\, 2^{g_b}\, b'_k, \qquad b'_k \in \left[-2^{w-1}; 2^{w-1}-1\right], \tag{5}$$

where w is the word length (including sign bit), g_a and g_b are the largest Most Significant Bit (MSB) positions of the FxP representations of a_k and b_k, respectively. It is typical to use the same word length for a_k and b_k but we allow different FxP formats in the numerator and denominator of the transfer function. The optimal design process proposed here is cast into the integer programming framework by searching for the integer counterparts a'_k and b'_k within the set of all representable numbers defined by w, g_b and g_a. While the word length w is a user-given parameter in our setting and is typically desired to be as small as possible in order to save resources, the values of the MSB positions g_b and g_a are not known. Hence, in order to provide an ILP model one must first predict the precision and the range of the FxP coefficients. More precisely, the goal is to predict the MSB positions that, on the one hand, encompass all representable coefficients, *i. e.*, do not lose solutions, and that maximize the precision, *i. e.*, do not overestimate the MSB, on the other hand. We find upper bounds for g_a and g_b via a pre-processing routine described in Section II-C.

The FxP coefficients must satisfy frequency specifications (3), which are non-linear and involve a complex modulus and variable products. Section II-B presents linearized frequency specification constraints, for which we detect a symmetry in the obtained solution. We present and prove a symmetry-breaking constraint in Section II-E, significantly reducing the search-space.

Finally, we search for FxP coefficients realizing the filter with the optimal number of adders in its hardware implementation. As can be inferred from Fig. 1, two MCM blocks, for coefficients a_k and b_k, need to be optimized. In Section II-D we demonstrate how to combine or improve the state-of-the-art ILP formulations [10], [11] in order to efficiently model both multiplier blocks in the IIR.

Finally, a short summary of the workflow is presented in Section II-F.

B. Formulating Linear Frequency Constraints

In order to obtain an ILP model, we have to solve two issues related to the frequency constraints (3). First of all, (3) actually enforces one constraint for each $\omega \in [0;\pi]$. There are infinitely many such constraints to be satisfied. The constraint (3) is called semi-infinite (see [22] for a survey on this class of constraints). The standard discretization approach to handling semi-infinite constraints consists in discretizing $[0;\pi]$ into a finite set of frequencies Ω_d, leading to a finite number of constraints. This discretization can be dynamically updated, but we consider here a fixed discretization and a *a posteriori* verification of the semi-infinite frequency constraint [23]. In the rare cases where the verification fails, the verification procedure proposed in [23] outputs a faulty frequency that can be added to the discretized set of frequencies for a new trial. Continuous variables generically lead to active constraints, which require some theory and accurate algorithms with some adaptive discretization to allow discovering active frequencies for optimal coefficients (see [22], [24] for details). Integer variables generically do not lead to active constraints. This means that only a finite number of faulty frequencies can happen after the ILP design, before we obtain coefficients verifying rigorously the frequency specifications. This reduction to a finite problem of the semi-infinite constraint is a positive side effect of searching directly FxP representations of the coefficients.

After this discretization of the semi-infinite constraint, a finite number of nonlinear constraints needs to be handled. For a fixed $\omega \in \Omega_d$, the constraint (3) includes a complex absolute value, which involves a square-root of the sum of squared terms, and a fraction. By incorporating (2) into (3), then multiplying with the denominator $\left|A\left(e^{j\omega}\right)\right|$, and finally squaring the result we obtain a constraint equivalent to (3)

$$\left|A\left(e^{j\omega}\right)\right|^2 \underline{\beta}(\omega)^2 \le \left|B\left(e^{j\omega}\right)\right|^2 \le \left|A\left(e^{j\omega}\right)\right|^2 \overline{\beta}(\omega)^2, \tag{6}$$

where,

$$\left|B\left(e^{j\omega}\right)\right|^2 = \sum_{k=0}^{2}\sum_{l=0}^{2} b_k b_l \cos((k-l)\,\omega), \tag{7}$$

$$\left|A\left(e^{j\omega}\right)\right|^2 = \sum_{k=0}^{2}\sum_{l=0}^{2} a_k a_l \cos((k-l)\,\omega), \text{ with } a_0 = 1. \tag{8}$$

Yet, in (7) and (8), the filter coefficients are still involved in bilinear terms $b_k b_l$ and $a_k a_l$. Billionnet *et al.* [25] proposed a method that allows linearizing products of positive integers assuming that bounds on these numbers are known: consider the product $z = xy$ where $x, y, z \in \mathbb{N}$, with $x \le \overline{x}$ and $y \le \overline{y}$. The linearization [25] basically consists in first rewriting one of the positive integers into its binary representation

$$x = \sum_{i=0}^{\lceil \log_2 \overline{x} \rceil + 1} 2^i\, t_{x,i}, \tag{9}$$

where $t_{x,i}$ are binary auxiliary variables. This constraint ensures that the bits $t_{x,i}$ encode the value of x. Then, the product $z = xy$ becomes a sum of products between the binary variables $t_{x,i}$ and the positive integer y. Finally, such a binary by integer product is common and its well-known linearization involves indicator or big M constraints [26], [27].

Here however, x and y correspond to filter coefficients that have no sign restriction. We extend the linearization exposed above to signed integers by adding the auxiliary

variables $x^+, y^+ \in \mathbb{N}$ and $x^{\text{sg}}, y^{\text{sg}} \in \{0,1\}$, and link them by following constraints

$$x^+ = |x|\,, \qquad y^+ = |y|\,, \tag{10}$$

$$x^{\text{sg}} = \text{sign}(x)\,, \qquad y^{\text{sg}} = \text{sign}(y)\,, \tag{11}$$

$$z^+ = x^+ y^+, \tag{12}$$

where the linearization of the absolute values (10) and the sign constraints (11) are well-known and involve indicator or big M constraints [28], [29], and where $z^+ = x^+y^+$ is the positive case we already presented. Finally, $z = \pm z^+$ and the sign is determined by the values of x^{sg} and y^{sg} directly in the model. This whole linearization relies on the fact that bounds on x and y are known. This is addressed in the next subsection.

C. Stability and Bounds on Filter Coefficients

Necessary and sufficient stability conditions for second-order filters are well-known [30, Section 16.8] to be

$$-2 < a_1 < 2, \tag{13}$$

$$|a_1| - 1 < a_2 < 1. \tag{14}$$

As explained before, the absolute value is standardly linearized using indicator or big M constraints, thus (13)-(14) actually fit in an ILP model. From these constraints it is straightforward to derive bounds on a_k: $a_1 \in]-2;2[$ and $a_2 \in]-1;1[$. These bounds are independent of the frequency specification of the filter and yield an upper bound, $g_a = 1$, for the MSB of the coefficients a_k.

Bounds on b_k, however, cannot be obtained independently of the filter specifications. Using the bounds (13)-(14) and the fact the cosine in (8) belongs to $[-1;1]$, we deduce that $\left|A\left(e^{j\omega}\right)\right|^2 \leq 16$. This bound together with the frequency specification constraints (6) lead to the constraint

$$\left|B\left(e^{j\omega}\right)\right|^2 \leq 16\overline{\beta}(\omega)^2\,, \tag{15}$$

that needs to be satisfied by the coefficients b_k

As can be seen from (7), $\left|B\left(e^{j\omega}\right)\right|^2$ is a quadratic form $b^T Q b$ with respect to the variables b_k. Its characteristic matrix Q, whose entries are $Q_{kl} = \cos((k-l)\omega)$, is symmetric and its spectrum is $\{0, 1-\cos(2\omega), 2+\cos(2\omega)\}$. Its eigenvalues being non-negative, the inequality constraint (15) is convex. As a consequence, some lower and upper bounds on the coefficients b_k can be computed by solving the convex quadratic problems consisting in minimizing or maximizing b_k subject to the convex quadratic constraints (15) for all frequencies $\omega \in \Omega_{d'}$ where $\Omega_{d'}$ is a discretization of Ω. The global minimum and maximum of these problems are required to be used as valid lower and upper bounds, such global extrema being easily computed by local solvers since the quadratic constraints are all convex and the cost is linear. In particular, common mixed ILP solvers can solve this kind of nonlinear problem. In addition to allowing the linearization of frequency constraints, these bounds permit to decide a first upper bound, g_b, for the MSB of the coefficients b_k.

These bounds do not fully take into account the specificity of the filter we are designing and use the worst case bounds in (15). In any case, it is possible to compute tighter bounds on the coefficients in order to reduce the search space as explained in Section II-E.

D. MCM for Direct-Form IIR Filters

The FxP coefficients satisfying the above frequency and stability constraints are also subject to hardware constraints. We use the ILP-based hardware models for optimal multiplierless constant multiplication presented in [10], [11]. These models, given a set of constant coefficients, solve either a minimization w. r. t. number of adders [10] (with bounded adder graph depth) or a satisfiability problem [11] (whether an implementation with a given number of adders is feasible).

For the global IIR filter design problem, the coefficients are, however, the unknowns. Similarly to [21], we first design a number of linking constraints that bind the coefficient variables for filter design with the inputs of an MCM problem. We refer to them as *glue constraints* in Fig. 3.

The challenge is that for the IIR filters two multiplier blocks should be optimized simultaneously, one for coefficients a_k and one for b_k, with the goal of minimization of the *total* number of adders, *i. e.*, in both multiplier blocks and in the filter structure. For the minimization model [10] this can be achieved by incorporating twice the MCM constraints, one for each multiplier block, and combining the objective functions

$$\min A_{M_a} + A_{M_b} - \sum_{k=0}^{2} \zeta_k^b - \sum_{k=1}^{2} \zeta_k^a, \tag{16}$$

where A_{M_a} and A_{M_b} denote the number of adders in multiplier blocks for a_k and b_k, respectively; and ζ_k^a and ζ_k^b are binary variables validating whether the respective filter coefficients are zero. In other words, the number of zero-valued coefficients is maximized, privileging sparsity in the implemented filters.

If a general ILP model [11] solving a satisfiability problem is used, a simple duplication of constraints is not possible. When a target number of total adders is given to the satisfiability problem, a trade-off problem arises: how are they distributed between the two multiplier blocks? In order to explore the trade-off space, a simple but costly solution would be to verify all possible distributions via an external loop, *e. g.*, distributing four adders between two multiplier blocks would require solving different models five times.

We propose a more efficient solution, where we enhance the MCM model such that the distibution of adders among multiplier blocks is done by the solver, instead. The idea is to produce one adder graph, which can be separated into two independent multiplier blocks, one for a_k and one for b_k. In other words, we propose an MCM model that, given a certain number of input coefficient variables, adds the constraint that one part of those is separable from the other. Then, the satisfability problem will be solved for both MCM block simultaneously, by the solver.

This is achieved in the following way. In the orginal model, a variable c_p is associated to every adder p and these variables are linked together with binary variables $c_{p,i,q}$ where $c_{p,i,q} = 1$ if, and only if, the adder q is the i-th input of adder p. In the new model, we need to ensure that the adders related to the

coefficients a_k are not computed using the adders related to the coefficients b_k, and *vice versa*.

We propose to add binary variables mcm_p^a and mcm_p^b, for each adder p, in order to ensure that the adder can be used for one adder graph or the other, but not both. We add a set of binary variables, $s_{p,q}$, which encodes the information about two adders p and q being used in the same adder graph, and a set of the following constraints

$$c_{p,i,q} \leq s_{p,q} \qquad \forall p, q, i, \tag{17}$$

which ensure that an adder cannot be used as the input of another if they are not in the same adder graph.

In a similar way, using mcm_p^a and mcm_p^b, we ensure that the glue between the filter coefficients and the adders is only possible for adders in the corresponding adder graph.

Both solutions, for the satisfibality or minimzation problems, prioritize coefficient sparsity and even permit an FIR filter design when possible, as a_k can be equal to 0.

E. Search Space Reduction

1) Linearized Specifications Projections: The bounds for a_k and b_k computed in the previous section allow implementing an ILP model by fixing an FxP format and linearizing the bilinear terms involved in the frequency specifications. This first incomplete ILP model does not include the geometry of the adder graph that represents the multiplierless solution as defined in Section II-D. Yet, in order to speed up the solving of the complete ILP model, we tighten the bounds on the coefficients by solving these incomplete ILPs that are simpler than the complete ILP model in the sense that it does not include the geometry of the adder graph defined in Section II-D: they consist in minimizing or maximizing b'_k subject to the stability constraints and the linearized frequency constraints. These ILPs are obviously more difficult to solve than the continuous convex quadratic problems used in the previous section to obtain crude bounds, but still much easier to solve than the final complete model. As it has been shown in [21] in the context of the design of FIR filters, solving these simpler ILPs to obtain tighter bounds for the solving of the final ILP is worthwhile.

2) Symmetry Breaking: The model contains some symmetries that can be broken in order to reduce the search space: the first symmetry consists in simultaneously changing the sign of the values taken by b_0, b_1 and b_2, the second symmetry consists in exchanging the values taken by b_0 and b_2. These two symmetries leave the constraints on the coefficients b_k unchanged, as can be easily seen on the explicit expression

$$b_0^2 + b_1^2 + b_2^2 + 2b_0b_1\cos(\omega) + 2b_1b_2\cos(\omega) + 2b_0b_2\cos(2\omega), \tag{18}$$

of (6) involved in the constraint (15): indeed, (18) is insensitive to changing all coefficients sign simultaneously and to exchanging variables b_0 and b_2. Both symmetries have no incidence on the MCM problem defined in Section II-D, the existence of symmetric adder graphs being obvious for opposed or exchanged coefficients.

As a consequence, from an arbitrary solution with

$$b_0 = b_0^*,\ b_1 = b_1^* \text{ and } b_2 = b_2^*, \tag{19}$$

we can build three new solutions by simply applying these symmetries to obtain

$$b_0 = -b_0^*,\ b_1 = -b_1^* \text{ and } b_2 = -b_2^*, \tag{20}$$

$$b_0 = b_2^*,\quad b_1 = b_1^* \quad \text{and } b_2 = b_0^*, \tag{21}$$

$$b_0 = -b_2^*,\ b_1 = -b_1^* \text{ and } b_2 = -b_0^*. \tag{22}$$

The fourth solution (22) is obtained by applying the two symmetries consecutively, in any order. Note that some of these four symmetric solutions (19)-(22) may be equal in some special cases, *e. g.*, when $b_0 = b_1 = b_2 = 0$.

Breaking these symmetries means finding additional constraints, called symmetry breaking constraints (SBCs), that remove some symmetric solutions but keep at least one of them [31]. If necessary, symmetric solutions that have not been calculated due to the SBCs can be built afterward. In the best case, SBCs keep only one solution among all symmetric solutions, in that case the SBCs are called total. Therefore, we expect here to reduce the size of the search space by a ratio of four since solutions come within symmetry classes containing four symmetric solutions. Finding SBCs for general symmetry groups is difficult, *e. g.*, SBC generation for symmetries consisting only of variable permutations rely on modern group theoretic algorithms [32]. In our case, the symmetry group is generated by one variable permutation and one central symmetry. Up to our knowledge, SBCs involving both variable permutations and central symmetries have not yet been investigated and no SBC defined for them.

In order to derive these SBCs, the search space for b_0, b_1 and b_2, which is $\mathbb{R}^3$, is divided into four areas:

$$\Sigma_1 = \left\{(b_0, b_1, b_2) \in \mathbb{R}^3 \mid b_0 \geq |b_2|\right\}, \tag{23}$$

$$\Sigma_2 = \left\{(b_0, b_1, b_2) \in \mathbb{R}^3 \mid -b_0 \geq |b_2|\right\}, \tag{24}$$

$$\Sigma_3 = \left\{(b_0, b_1, b_2) \in \mathbb{R}^3 \mid b_2 \geq |b_0|\right\}, \tag{25}$$

$$\Sigma_4 = \left\{(b_0, b_1, b_2) \in \mathbb{R}^3 \mid -b_2 \geq |b_0|\right\}. \tag{26}$$

Then, one can verify that: a solution lying inside Σ_2 moves to Σ_1 applying the sign symmetry; a solution lying inside Σ_3 moves to Σ_1 applying the exchange symmetry; a solution lying inside Σ_4 moves to Σ_1 applying both symmetries consecutively. As a consequence, one can restrict the search to Σ_1 and reconstruct all solutions using symmetries. This restriction to Σ_1 is achieved by adding the SBC

$$b_0 \geq |b_2| \tag{27}$$

to the model. Restricting to another Σ_i would be lead to another SBC, with an equivalent improvement of the resolution process.

For completeness, it should be noted that on the frontier $\Sigma_k \cap \Sigma_l$ between areas, two equivalent solutions might still be kept despite the SBC. However, resolving this issue is counterproductive, because the great majority of the search space lies in the interior of the sets Σ_i, and removing the symmetries on the boundaries would introduce many additional constraints.

F. Wrapping Up

To wrap it up, our approach is to model both the filter design and the design of a constant multiplication scheme through

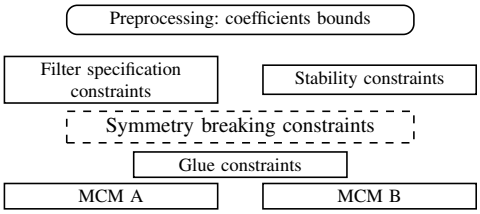


Fig. 3: High-level structure of the global ILP model IIRoptim.

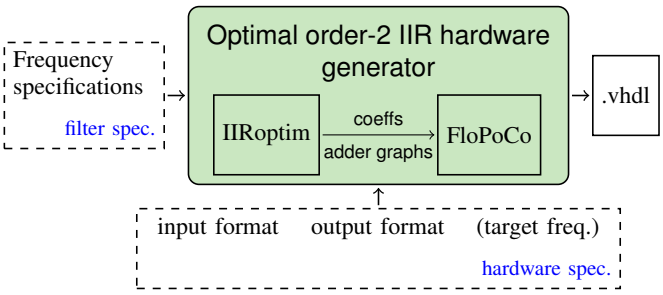


Fig. 4: Interface of the proposed tool.

shift-and-add using a global ILP problem. On top of that, we search for coefficients directly as integers, *i. e.*, in a FxP format with user-given word length. Since filter coefficients, depending on the filter specification and word length, can have different MSB positions, we first perform a pre-processing. This pre-processing consists in solving a quadratic convex optimization problem yielding tight and rigorous bounds on filter coefficients and permitting to define coherent MSB positions and construct the design space for the main model. As Fig. 3 shows, the high-level IIR model consists of the following constraints:

- linearized frequency-specification constraints (6)-(8) ;
- stability constraints (13)-(14);
- (optional) symmetry breaking constraints (27) enabling a significant reduction of the design space;
- constraints responsible for the design of optimal constant-multiplication blocks for a_k and b_k using [10] or our modification of [11];
- so-called glue constraints, connecting the unknown filter coefficients with multiplier blocks analogously to [21].

The above ILP model can now be solved using any available ILP solver. Moreover, despite its non-linearity, the pre-processing problem can be solved using most generic ILP solvers as well, thanks to its convexity.

III. Experimental Results and Discussion

A. Implemented Toolflow

We implemented our approach into a tool, whose minimalist interface is shown in Fig. 4. The input specification includes the frequency-domain specification of the filter, and also the information for hardware implementation, *i. e.*, the coefficient word length, the input/output formats and performance parameters, such as the required target frequency. All the other parameters (filter coefficients and their FxP format, the multiplierless operator architecture, the representation of intermediate data, etc.) are determined automatically as a part of a global optimization process. One of the goals of our tool is to bring the attention of the filter designer to the higher-level parameters, *e. g.*, filter specifications, while relying on our optimal implementations.

TABLE I: Sets of lowpass filters used for the IIR experiments. First the set with decreasing δ, next the sets with increasing passband and stopband. Finally, a lowpass filter.

Benchmarks	$\texttt{lp1}_k$	$\texttt{lp2}_k$	$\texttt{lp3}_k$	lp4
k	$\{0,1,\ldots,6\}$	$\{0,1,\ldots,4\}$	$\{0,1,\ldots,4\}$	–
passband/π	$[0;0.3]$	$[0;0.3+0.05k]$	$[0;0.3]$	$[0;0.5]$
stopband/π	$[0.7;1]$	$[0.7;1]$	$[0.7-0.05k;1]$	0.91
δ	$0.1-0.01k$	0.1	0.1	0.1

Our new ILP model and the front-end of the proposed tool are implemented in the tool `IIRoptim` using the `julia` language, which offers a unified access to the major ILP solvers through the `JuMP` library. Given the user input, we first construct an ILP model and solve it with one of many open-access or commercial solvers available through `JuMP`, such as Gurobi, CPLEX, GLPK, etc. The result of the global optimization problem, *i. e.*, the list of filter coefficients and the adder graphs defining the optimal shift-and-add architectures, is then passed on to the FloPoCo tool [33], which was extended for that. FloPoCo is the state-of-the-art tool for the design and automatic generation of fixed/floating-point arithmetic cores. We implemented a new operator `FixIIRShiftAdd` within FloPoCo, generating faithfully-rounded multiplierless IIR filters, *i. e.*, only the last output bit might be erroneous and all other bits are guaranteed to be correct. The new operator alleviates the filter designer from all internal architectural decisions and presents a final VHDL code. FloPoCo also requires a target frequency to perform an automatic pipelining [42]. However, this parameter has only effect on the non-recursive parts of the filter as no additional registers can be added within the recursive part, so this parameter has limited effect.

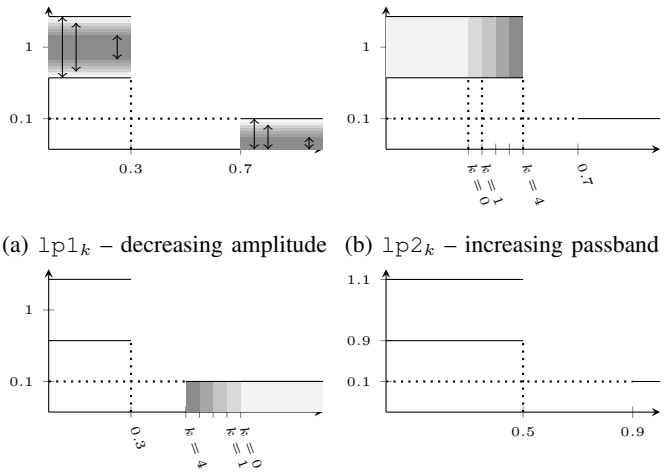
B. Set of Benchmarks and Comparison Approaches

Benchmarks: Although the design of second-order IIR filters is an important part of the design of larger order filters, benchmarks are rarely targeting frequency specifications of individual second-order sections. Hence, we use three sets of filter specifications with increasing filter design difficulty that could without doubt be used in real-life applications. In addition to that, we add another artificial low-pass filter, and a real-life example from [15].

The normalized low-pass filter specifications are here defined as

$$\begin{aligned} 1-\delta \le \left|H(e^{i\omega})\right| &\le 1+\delta, \qquad \forall \omega \in [0;\omega_p], \quad \text{(passband)} \\ 0 \le \left|H(e^{i\omega})\right| &\le \delta, \qquad \forall \omega \in [\omega_s;1]. \quad \text{(stopband)} \end{aligned}$$

We fix the initial passband to $[0;0.3]$, stopband to $[0.7;1]$ and $\delta = 0.1$. Then, for each of the families of filter specifications, we vary one of the parameters in dependence of a variable k to increase the filter design difficulty. The detailed frequency specifications for each family of filters are given in Table I, and their graphical representation is sketched in Fig. 5. For example, in family `lp1`, the δ varies from 0.1 to 0.04 with step



(a) $\texttt{lp1}_k$ – decreasing amplitude (b) $\texttt{lp2}_k$ – increasing passband

(c) $\texttt{lp3}_k$ – increasing stopband (d) $\texttt{lp4}$

Fig. 5: Proposed families of benchmarks.

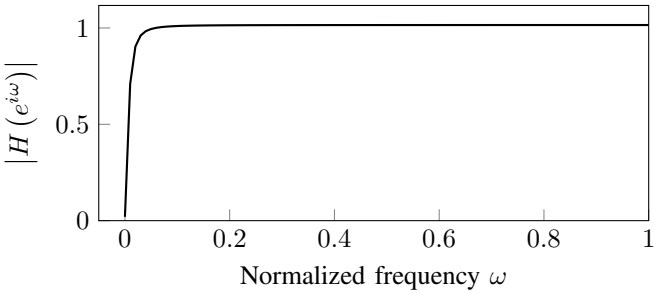


Fig. 6: Frequency response of the compensator $\texttt{hp0}$ [35], [36].

$0.01k$ where $k = 0, \ldots, 6$. However, the designs were possible only up to $k = 5$, reaching the maximum design possibilities for second-order IIR filters. Analogously, the families $\texttt{lp2}$ and $\texttt{lp3}$ increase/reduce the pass/stopband, respectively. The filter specification $\texttt{lp4}$ is a lowpass with a short stopband.

Finally, our last benchmark $\texttt{hp0}$ is a real-life application, it is a highpass filter (Fig. 6), which is a compensator used in a magnetic-bearing control system. It was initially derived by discretizing the analog controller [34] and further used as an example in word length optimization literature [15], [35]. It is most recently used in [15] to demonstrate a KCM-based faithfully-rounded implementation of IIR filters. This filter is not defined in terms of frequency specifications but by its frequency response (sole poles and zeros are given in [34]), hence to implement the three-step approaches we approximate the frequency specification in Matlab using the coefficients from [15], [35]. For our implementation, the versatility of an ILP modeling permits to easily integrate frequency response bounds as functions of ω and similarly approximate the filter $\texttt{hp0}$.

Comparison approaches: We aim at comparing with the classical and state-of-the-art approaches for the IIR design comparing high level metrics like adder counts and implementation results for FPGAs and ASICs. The classical approach passes through three steps

1) **FD:** in our case double-precision filter coefficients are obtained, if possible, with Matlab's elliptic method;

2) **Q:** as in Matlab's Fixed-Point design Toolbox, we convert the double coefficients to FxP with the user-specified word length and round-to-nearest mode. Post-design quantization often leads to errors in frequency response or instabilities, hence we increase the word length s. t. the frequency-domain error is below a threshold ($< 10^{-7}$) and the filter is stable;

3) **I:** generic multipliers provided by hardware manufacturers are often used. On FPGAs, the digital signal processing (DSP) blocks (whose availability and number on each FPGA vary), are allowed but it might be interesting to disable them to provide comparison with purely LUT-based implementations. We also compare with the state-of-the-art constant multiplications based on KCM [15] and MCM [11] methods.

Different combinations of the above steps are possible. Each of the benchmark filter specifications will be implemented using the following approaches:

- 3-step Generic: Matlab's FD + Q + implementation using generic multipliers (using the VHDL '$\star$' operator, potentially using DSP blocks on FPGA);
- 3-step Generic NoDSP: same as above with disabled DSP blocks for synthesis and implementation;
- 3-step MCM: Matlab's FD + Q + optimal shift-and-add implementation for FxP coefficients;
- 2-step KCM: Matlab's FD + approach from [15], which directly obtains a KCM-based implementation for real coefficients;
- Ours: 1-step combined approach, performing directly FD & Q & I using optimal shift-and-add multipliers.

The 3-step Generic (NoDSP) can be seen as a baseline while 3-step MCM and 2-step KCM is the state-of-the-art. With such a setting, our goal is to analyze the benefits of the global approach compared to 2- or 3-step approaches that first fix filter coefficients to some values and only then optimize for implementation. It should be noted that beside the 3-step MCM approach, the existing 2- and 3-step approaches take less than a second for second order IIR filter design, and even the 3-step MCM approach should produce a solution in a few minutes in the worst case.

Bit widths: The coefficient word length is an input parameter for our tool, hence, for each specification we will explore a range of different coefficient word lengths. For the input/output data in hardware implementation, we used 8-, 12- and 16-bit configurations but, due to the lack of space, only the results for 16-bit experiments are reported in the paper. The same observations hold for the 8- and 12-bit experiments.

All benchmarks are available and reproducible, see the tool's git repository for full benchmark information.

C. Evaluation of the ILP Model and Design Results

In the following we evaluate the performance of our ILP model, explore the design space and compare with the 3-step MCM-based approach to see the benefits w. r. t. number of adders in a shift-and-add implementation. All experiments were performed on a Linux laptop with i7-10810U processor and 32 GB RAM. Any generic solver with an

TABLE II: Results for our global optimization method vs. applying optimal MCM upon quantized coefficients. The total number of adders A ($A = A_M + A_S$) consists in the multiplier block A_M, and structural A_S adders. Results are reported for the smallest coefficient word length W possible.

Benchmark	Our method				3-step MCM			
	W	A_M	A_S	A	W	A_M	A_S	A
$\texttt{lp1}_0$	4	1	4	5	10	8	4	12
$\texttt{lp1}_1$	4	1	4	5	16	11	4	15*
$\texttt{lp1}_2$	4	2	4	6	6	5	4	9
$\texttt{lp1}_3$	4	3	4	7	10	7	4	11
$\texttt{lp1}_4$	5	4	4	8	9	7	4	11
$\texttt{lp1}_5$	5	4	4	8	–	–	–	–
$\texttt{lp2}_0$	4	1	4	5	10	8	4	12
$\texttt{lp2}_1$	4	1	4	5	10	8	4	12
$\texttt{lp2}_2$	5	3	4	7	10	8	4	12
$\texttt{lp2}_3$	6	4	4	8	–	–	–	–
$\texttt{lp3}_0$	4	1	4	5	10	8	4	12
$\texttt{lp3}_1$	4	2	4	6	4	2	4	6
$\texttt{lp3}_2$	4	3	4	7	23	18	4	22*
$\texttt{lp3}_3$	5	3	4	7	–	–	–	–
lp4	4	1	4	5	4	3	4	7
hp0	6	1	2	3	11	6	4	10

* heuristic solution using [7]

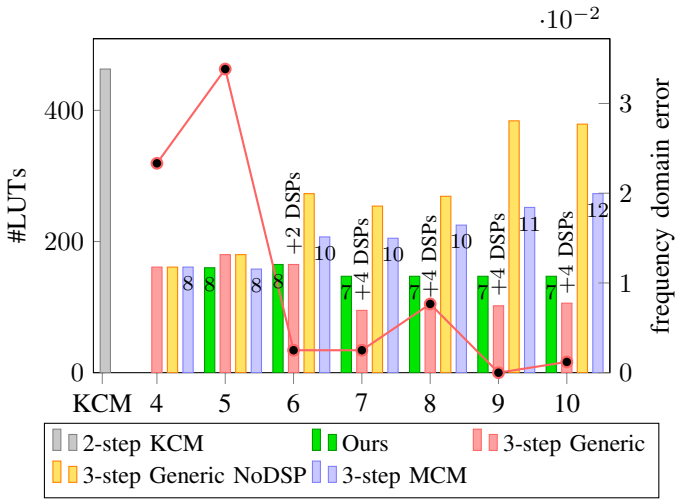


Fig. 7: Implementations for $\texttt{lp1}_4$ benchmark with coefficient word lengths varying from 4 to 10 bits. The bars correspond to the number of LUTs (left axis) and labels on bars denote number of adders. The frequency-domain error of 3-step methods is the red line (right axis).

interface for `JuMP` library [37] can be used, here we used CPLEX 12.10 [38].

The first remark concerning our tool is that the running times are quite reasonable, varying from 10 seconds for small word lengths (4-5 bits) and going up to a few minutes in the worst case if we push the word length to unnecessarily large values. These solving time results are similar to the ones for the 3-step MCM approach. By default, we use the symmetry breaking constraints as in general we observed a significant improvement (around 2×-20×) in running times, depending on problem complexity. Obviously, the complexity of the ILP model is increasing with increasing the word length, since the ranges of integer variables are doubled with each new coefficient bit, and a few additional variables and constraints are added as well. However, it is not the model complexity but numerical instabilities that represent the main bottleneck in pushing the coefficient word lengths further than 10-11 bits. Indeed, our ILP model for the MCM design makes intensive use of the so-called big-M constraints, that are limited by certain floating-point tolerances internal to the solver, beyond which the solver cannot use efficient floating-point arithmetic for integer programming. An alternative to the big-M are the indicator constraints but the drawback is the increased computational time, leading to a similar bound on the maximum coefficient word lengths.

The second remark is that with our tool, an infeasibility of the design problem can be quickly proven. For example, for the specification $\texttt{lp1}_4$ in just a few seconds we prove that no implementation that perfectly fits the specification with word length 4 is possible. This is an important feature, since when trying to lower the coefficient word length as much as possible, the filter designer can quickly stop the exploration. Inversely, when searching the smallest feasible word length, the design iteration will quickly move on from infeasible ones.

The goal of our tool is to provide optimal architectures w. r. t. the number of adders in the multiplierless implementation. The number of adders is not a fine-grained metric but it enables the design-space exploration a priori, before any hardware synthesis and experiments. It is a good indicator of the performance of implemented systems, as the number of adders is correlated with the number of LUTs.

We claim that optimizing the filter coefficients simultaneously for the filter design and MCM problem yields smaller total number of adders than the 3-step MCM approach, as it covers the whole design space. Table II presents the optimization results for our method, and for the 3-step MCM method. We report results for the smallest word lengths possible, in which our method has a feasible result (and, by construction, no frequency domain error) and the Q step in 3-step MCM yields stable filters with frequency-domain error smaller than 10^{-7}.

It can be noticed that our method finds solutions for smaller coefficient word lengths than the 3-step MCM, and with significantly smaller total number of adders for most of the cases (≈46% on average in Table II). In some cases, marked with asterisk, target word lengths for 3-step MCM were too big s. t. the optimal MCM timed-out and the RPAG heuristic [7] was used to obtain adder graphs instead. For the specification `lp4` both approaches find designs with 4-bit coefficients, but thanks to efficiently covering the whole design space of all possible FxP coefficients, our method determines coefficients that require fewer adders in the multiplier block and the total of 5 adders, instead of 7. Finally, the benchmark $\texttt{lp3}_1$ is the only case when the coefficients in 3-step MCM coincide with the ones found by our tool.

Another advantage of the proposed ILP formulation is privileging sparse implementations, which even for second-order filters largely improves performance due to smaller number

of structural adders. For instance, the benchmark hp0 was traditionally implemented with all non-zero coefficients [15], [35]. We have approximated the frequency response of this compensator using the poles and zeros from the original paper [34] and used it as reference in our ILP. As a result, a sparse implementation with 6-bit coefficients was possible, having as coefficients $b_0 = 1$, $b_1 = -1$, $b_2 = 0$, $a_1 = -31/32$, $a_2 = 0$ and leading to the total of 3 adders for the whole filter. To compare, the 3-step MCM can provide at minimum 11-bit coefficients implemented with 10 adders. Moreover, our design has poles further apart from the unit circle, which improves the numerical quality of the time-domain implementation.

Our tool provides results for difficult specifications even when FD with Matlab fails. For benchmarks $\texttt{lp1}_5$, $\texttt{lp2}_3$ and $\texttt{lp3}_3$ no IIR filter design method in Matlab could find double-precision coefficients for a frequency response of sufficient quality. Our method, however, successfully completes the task.

It is important to note that our tool provides optimal implementations w. r. t. the total number of adders *for a given coefficient word length.* In other words, finding the best coefficient word length is still the filter designer task. While typical flow is to stop at the smallest possible coefficient word length, as in Table II, it does not necessarily lead to the optimal implementation w. r. t. all possible word lengths, and increasing the coefficient word length can actually lead to fewer adders.

To illustrate this, Fig. 7 shows synthesis results for the $\texttt{lp1}_4$ benchmark with coefficient word lengths varying from 4 to 10 bits (the hardware implementation is described in details in Section III-D). First, one can note that our results start at 5-bit coefficients. Indeed, our tool proved that no stable solution satisfying the given frequency specifications is possible for coefficients with word length 4. It can be seen that increasing the word length from 5 bits to 7 bits actually reduces the total number of adders from 8 to 7 and improves the number of LUTs. Moreover, our approach provides a stable behavior: once a 7-adder solution is found, increasing coefficient word length will never lead to a worse implementation. In practice, the ILP either finds different coefficients with the same number of adders, or simply multiplies values by two for each additional bit (which is the case in Fig. 7 starting 7 bits). In other words, even if the user specifies a larger word length than required, our tool finds the best coefficients that might fit in a smaller format and guarantees that trying smaller word length will not give smaller number of adders.

For the classical 3-step approaches, design-space exploration is more difficult and irregular. For these methods, coefficient quantization introduces a frequency-domain error meaning that the quantized filter does not satisfy the frequency specifications any more (see the red line in Fig. 7). This error is highly non-linear, and a typical intuition that increasing coefficient word length improves the quality of filter is simply not true (see the frequency-domain error for 7- and 8-bit coefficients in Fig. 7). Hence, the search for the best coefficient size must be exhaustive for 3-step methods.

D. Hardware Implementation and Discussion

In the following, we first describe in details the faithfully-rounded architectures that we implement in FloPoCo and then discuss the synthesis results obtained for our benchmarks and each approach.

Implemented architecture: The addition in the center of the high-level description of the tDFII filter in Fig. 1 requires the addition of three inputs, hence it must be first separated into two consecutive two-input add operators. The critical path then contains one multiplication and two additions. To cut the critical path, pipeline registers are usually added, leading to three additional registers for tDFII.

In the perspective of extending this work to higher-order IIR filters decomposed into second-order sections, we propose to slightly modify the classic tDFII structure and separate the recursion into its own branch as in Fig. 8. The number of two-input adders stays the same, and the two additional registers are compensated by the fact that only one pipeline register is required to cut the critical path, instead of three for the tDFII. Hence, at the same hardware cost we get a better modularity of the feedforward/feedback loops.

We aim at providing faithfully-rounded implementations, *i. e.*, the precision of the output y (the Least Significant Bit (LSB) position ℓ_{out}) serves as the accuracy constraint. There are different ways to assign the sizes of data paths for a filter implementation, and it is important to not underestimate the sizes (otherwise overflow occurs) but, on the other hand, assigning larger data sizes will waste resources to compute unnecessary bits. Hence, our goal is to provide a code generator that, given the input/output format and filter coefficients, automatically computes the word sizes of all internal data paths to guarantee the time-domain error smaller than $2^{\ell_{\text{out}}}$ but not more.

Fig. 8 presents our approach for multiplierless hardware IIR on the example of $\texttt{lp1}_4$ benchmark, fulfilling frequency specifications in Fig. 2 with 7 adders. Its transfer function was obtained as

$$H_{\texttt{lp1}_4}(z) = \frac{25 \times 2^{-7} + 40 \times 2^{-7}\, z^{-1} + 25 \times 2^{-7}\, z^{-2}}{1 - 40 \times 2^{-6}\, z^{-1} + 20 \times 2^{-6}\, z^{-2}}. \tag{28}$$

The inputs to the architecture generator are the MSB and LSB positions of the input x and output y, the adder graphs for multiplier blocks a_k and b_k, and their corresponding LSBs. For instance, here $\ell_b = -7$, $\ell_a = -6$.

Obviously, one cannot compute exactly (or with some fixed precision) on each iteration, truncate to ℓ_{out} and simply feed truncated values back into the loop, as this will degrade tremendously numerical quality and accumulated errors will explode. The Worst-Case Peak Gain (WCPG) measure for IIR filters [39], which has been applied for hardware IIR filters implemented with KCM multipliers [15], permits to determine the necessary extended internal precision ℓ_{ext} s.t. the propagated error never reaches the LSB of the output. For example, in Fig. 8 $\ell_{\text{ext}} = \ell_{\text{out}} + G$, where the number of guard bits G for the filter $\texttt{lp1}_4$ determined with its WCPG is $G = 3$. Then, the output of the multiplier blocks needs to guarantee its result with accuracy ℓ_{ext}. In our architecture we perform all

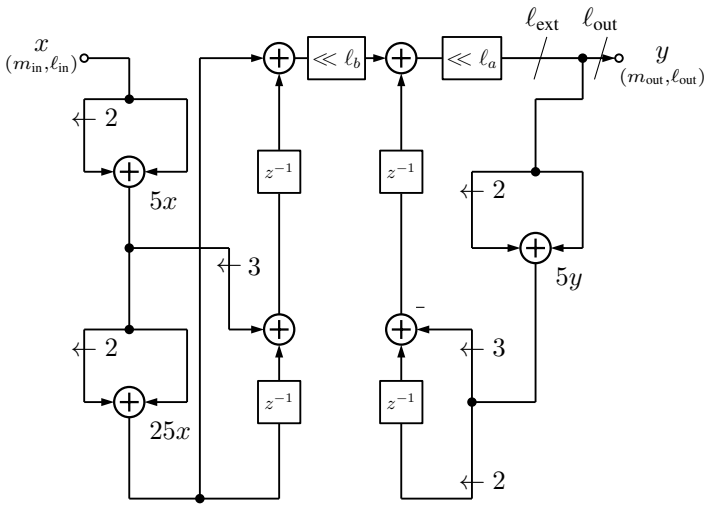


Fig. 8: The `lp14` benchmark can be implemented with mere 7 adders for 7-bit coefficients. All additions are exact, the truncation to internal extended format ℓ_{ext} is determined s.t. the output is faithfully rounded to ℓ_{out}.

additions and shifts exactly, increasing the size of data paths until their truncation to ℓ_{ext}.

For the generic approach, based on plain VHDL multipliers (using the $\star$ operator), we adopt a similar approach.

Synthesis results: FPGA synthesis was performed using Vivado v2020.2 for a Kintex 7 device (xc7k70tfbv484-3). The delay and power results are obtained after place and route. The power was evaluated for a 100 MHz clock for all dynamic parts incl. clocks, logic, signals and DSPs but no static or I/O power to ease the comparison of the cores. ASIC synthesis was performed using Cadence Genus v18.13 targeting a commercial standard-cell library in 14 nm FinFET technology from Global Foundry with 0.8 V supply voltage and regular VTH. Power dissipation is estimated by simulating the final netlist with 1000 random input vectors using Cadence NCSIM to obtain the switching activity. The timing constraints were set to 10 ns to obtain rather compact circuits.

To leave the process of (non-optimal) search for the best coefficient size out of scope, we further compare the designs with the smallest coefficient word lengths, for which our ILP has a feasible solution, and the 3-step approach has a frequency-domain error smaller than 10^{-7}.

Fig. 9 summarizes the obtained resources, the critical path delay and the power for FPGA and ASIC targets. For each benchmark, on x-axis we see the coefficient word length for our result (left) and for the 3-step quantization (right). For instance, we recognize the values 5 and 9 for the `lp14`. While for FPGA targets comparison w. r. t. all approaches is reasonable, for ASICs comparison with implementations using DSPs or the KCM table-based method is not representative due to technology differences.

We can observe that on both FPGAs and ASICs our method is always superior to any of the 3-step or KCM-based methods w. r. t. all metrics. On our FPGA benchmarks we observed the average LUT improvement of 48% compared to the best results of other methods, excluding the one using DSPs. In addition to that, the proposed approach offers the lowest delay with an improvement of 27%, which is not as drastic as LUT improvement but expected due to the similar number of delays. Finally, the resource reductions translate to the significant power reduction of 57%, on average, and up to 95% in the best case.

On ASICs benchmarks, our designs yield similar results, reducing the area by 48% and the delay by 27% on average, with the peak reduction by 76% and 51%, respectively. Our new designs demonstrate an impressive average power improvement of 65%, and in the best cases of 90%. This means that our designs divide the power consumption by 4 on average, and by almost 10 in the best observed cases.

Of course, our method is more efficient partly due to the smallest possible coefficient word lengths, but even when using the same word sizes (see benchmark `lp4`) our design requires less LUTs and has smaller delay.

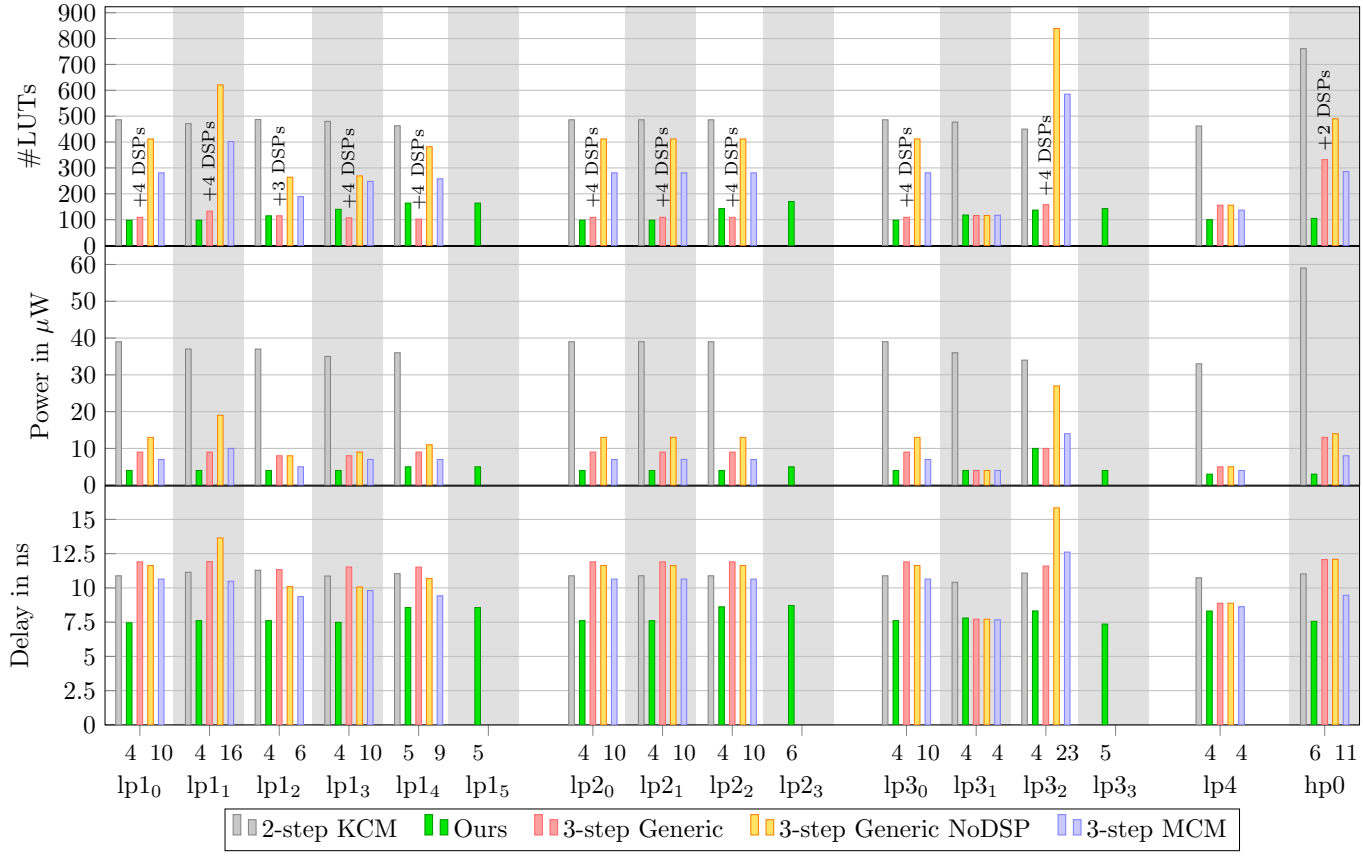
The KCM-based approach, representing the state-of-the-art for faithfully-rounded IIR filters, had worse general performance than our method in all benchmarks. While the LUT consumption of KCM-based IIR filters is significantly bigger than for our approach, it is comparable to the 3-step Generic NoDSP approach. However, in terms of delay, KCM-based multipliers prove to be generally faster or comparable to 3-step methods. It should be noted that the superiority of our approach over KCM-based IIR filters is not a surprise for the small word lengths, according to the recent result [14], and while it can be expected that our approach is possible only for word lengths less than 16 bits due to optimization time-out, the KCM-based multipliers will have no issue dealing with large word lengths.

It can be noted that with the increase in frequency specification difficulty, the 3-step methods generally degrade in performance, while our design methods have a more regular behavior, providing small and fast implementations even for complicated filters.

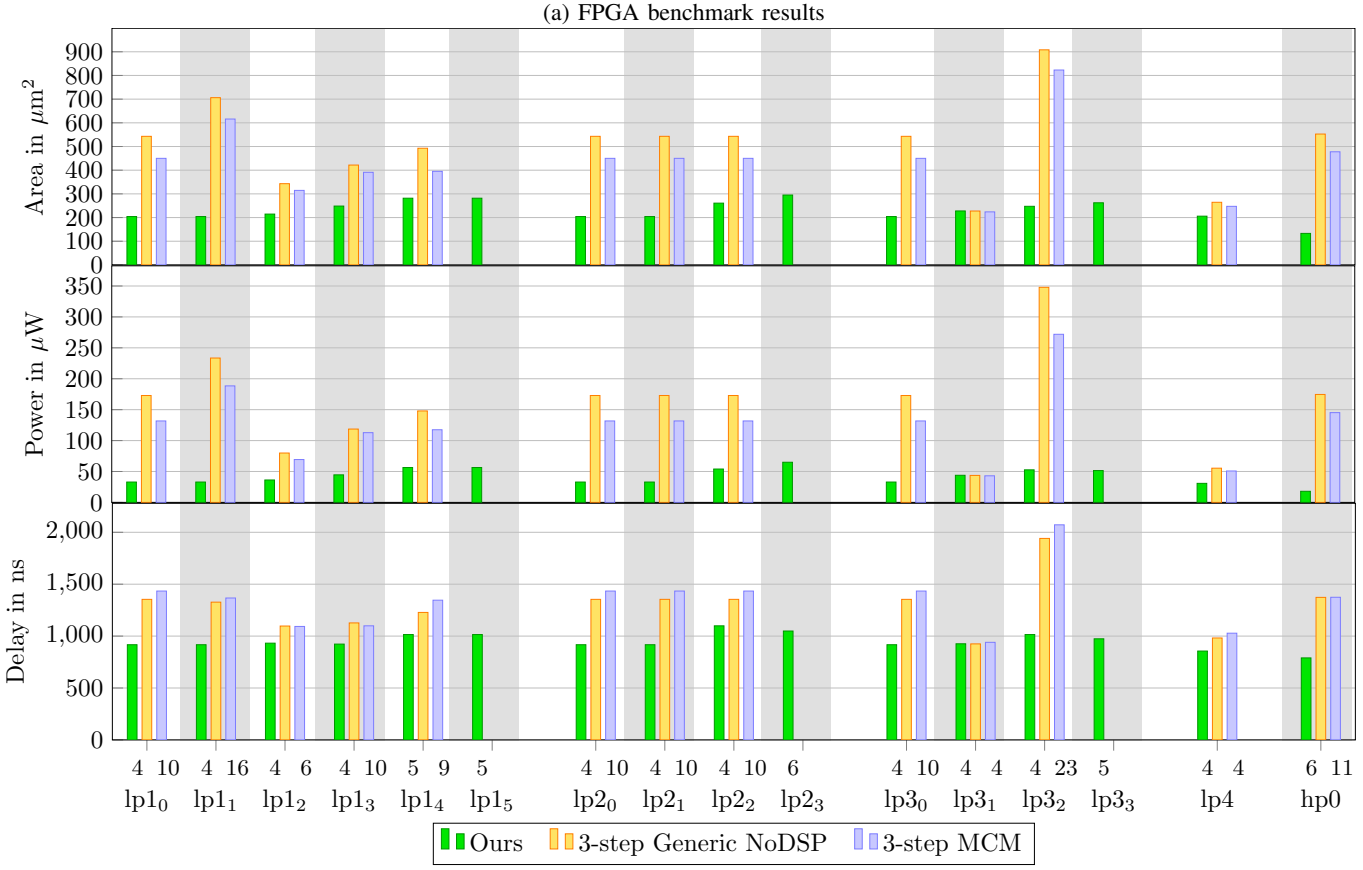
One of the best performance improvements was achieved for the `hp0` compensator. This is due to the much smaller coefficient word length than in all the previous literature, the sparsity and the fact that we succeeded in finding a filter with poles further from the unit circle. As a consequence, the number of internal guard bits was smaller too, resulting in only 105 LUTs compared to 286 in the 3-step MCM and 760 in the KCM-based state of the art implementation. Remark that in [15], `hp0` is implemented for 10-bit I/O word lengths but for the uniformity of experiments and without loss of generality, Fig. 9 features the 16-bit I/O implementation, generated with the FloPoCo operator presented in [15].

IV. Conclusion and Perspectives

We proposed a new method for the optimal design of multiplierless second-order IIR filters w. r. t. the number of adders. Our approach is based on a combined global optimization problem, which searches for stable filter coefficients directly in FxP format such that the number of adders in a shift-and-add implementation is minimized. Furthermore, we proposed an automated tool which combines our approach with



(a) FPGA benchmark results



(b) ASIC benchmark results

Fig. 9: Benchmark results of resources, power and critical path delay using I/O word length 16 bits for FPGA (top) and ASIC (bottom) targets. The x-axis features the benchmark name and the smallest possible coefficient word length for our designs (left) and 3-step approaches (right).

FloPoCo and provides automatic code generation for FPGAs and ASICs. With our tool, the instabilities and coefficient quantization effects on second-order IIR filters become a thing of the past, and efficient faithfully-rounded implementations automatically generated.

We proposed a linearized formulation of the combined filter design and MCM problem as one ILP model, which provides a convenient way for extensions. Several design space reduction techniques were proposed, including a novel symmetry breaking constraint, which we formally proved. As a result, the filter design and optimization takes mere seconds.

Our approach has several useful and important consequences. First, with our ILP it is easy and quick to prove the impossibility of a design, with a given coefficient word length and large enough range of MSBs, such that the filter specifications are *fully* satisfied. Moreover, this would be true for any smaller coefficient word length, providing a filter designer the unprecedented assurance that the design space exploration in that direction can be stopped. Second, if the filter design is possible with a certain given word length, increasing the word length will never yield a larger number of adders, and often the coefficients will remain the same or shifted by one bit. With this property and the fact that all our designs are guaranteed to have zero frequency-domain error, the design-space exploration becomes a more regular process.

For hardware experiments, we provided a faithfully-rounded multiplierless operator for IIR filters within FloPoCo. The synthesis results confirmed that a global optimization approach is superior to the multi-step FD & Q & I classical methods, and even to the state-of-the-art KCM-based IIR filters. With the ILP formulation we search directly the FxP coefficient design space, privileging sparse implementations and sometimes finding the filters that even double-precision Matlab fails to find. After testing the tool on numerous benchmarks, we observed a 48% reduction in number of LUTs, a 27% reduction in delay and a 57% reduction in power, on average, for FPGA targets. Implementing the same benchmarks on 14 nm ASIC confirmed the efficiency of our tool, demonstrating a 48% reduction in area, 27% reduction in delay and $4\times$ smaller power consumption, on average.

The superiority over the KCM-based IIR [15], which introduced faithfully-rounded filters using analysis of the worst-case rounding errors, advances the progress towards reliable IIR filters and demonstrates again that numerical guarantees do not necessarily come at a higher cost.

Some efforts are still required to extend our method, in particular to higher order filters. We see two possible directions for that extension: first, a single ILP model which would permit the design of cascaded second order sections and, second, an external loop for the decomposition of specifications into simpler specifications that are reachable by a second order filter. Extension to other structures than the Direct Forms is also a promising direction that should be tackled in the future.

Although the number of adders is a reliable high level metric, optimizing the number full adders instead would be better. We are fairly optimistic on the fact that our method can be refined to minimize that criteria, first as a post-design optimization of truncations, and then as one global optimization problem. Furthermore, we plan to introduce the truncations of data paths and model the rounding-error in the ILP model. This will permit further performance gains in the implemented filters.

REFERENCES

[1] Z. Smékal and R. Vích, "Optimized models of IIR digital filters for fixed-point digital signal processor," in *ICECS'99. Proceedings of ICECS '99. 6th IEEE International Conference on Electronics, Circuits and Systems*, vol. 1, Sep. 1999, pp. 145–148.

[2] G. Vanuytsel, P. Boets, L. Van Biesen, and S. Temmerman, "Efficient hybrid optimization of fixed-point cascaded IIR filter coefficients," in *IMTC/2002. Proceedings of the 19th IEEE Instrumentation and Measurement Technology Conference*, vol. 1, May 2002, pp. 793–797.

[3] Y. Wang, B. Li, and Z. Li, "Fixed-point digital IIR filter design using multi-objective optimization evolutionary algorithm," in *2010 IEEE Youth Conference on Information, Computing and Telecommunications*, Nov. 2010, pp. 174–177.

[4] M. Gevers and G. Li, *Parametrizations in Control, Estimation and Filtering Problems: Accuracy Aspects*, 01 1993.

[5] 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.

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

[7] M. Kumm, P. Zipf, M. Faust, and C.-H. Chang, "Pipelined adder graph optimization for high speed multiple constant multiplication," in *2012 IEEE International Symposium on Circuits and Systems*, May 2012.

[8] L. Aksoy, E. O. Güneş, and P. Flores, "Search algorithms for the multiple constant multiplications problem: Exact and approximate," vol. 34, no. 5, pp. 151–162, Aug. 2010.

[9] 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.

[10] M. Kumm, *Multiple Constant Multiplication Optimizations for Field Programmable Gate Arrays*. Springer Fachmedien Wiesbaden, 2016.

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

[12] K. D. Chapman, "Fast Integer Multipliers Fit in FPGAs," *Electronic Design News*, 1994.

[13] M. Faust and C.-H. Chang, "Bit-parallel Multiple Constant Multiplication using Look-Up Tables on FPGA," *IEEE International Symposium of Circuits and Systems (ISCAS)*, pp. 657–660, 2011.

[14] F. d. Dinechin, S.-I. Filip, L. Forget, and M. Kumm, "Table-Based versus Shift-And-Add Constant Multipliers for FPGAs," in *IEEE Symposium on Computer Arithmetic (ARITH)*, 2019.

[15] A. Volkova, M. Istoan, F. De Dinechin, and T. Hilaire, "Towards Hardware IIR Filters Computing Just Right: Direct Form I Case Study," *IEEE Transactions on Computers*, vol. 68, no. 4, pp. 597–608, 2019.

[16] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*, 3rd ed. USA: Prentice Hall Press, 2009.

[17] H. Leich, "Toolbox for the design of IIR digital filters," in *Proceedings of 13th International Conference on Digital Signal Processing*, vol. 2, Jul. 1997, pp. 621–624.

[18] W. K. Jenkins and M. Nayeri, "Adaptive filters realized with second order sections," in *ICASSP '86. IEEE International Conference on Acoustics, Speech, and Signal Processing*. Institute of Electrical and Electronics Engineers, 1986.

[19] A. G. Dempster and M. D. Macleod, "Comparison of IIR filter structure complexities using multiplier blocks," in *Proceedings of ISCAS'95 - International Symposium on Circuits and Systems*, vol. 2, Apr. 1995, pp. 858–861.

[20] S. Ansari, G. Kishor, P. K. Verma, N. Agrawal, I. Sharma, and A. Kumar, "Design of multiplierless digital iir filter using modified cuckoo search algorithm," in *2018 International Conference on Communication and Signal Processing (ICCSP)*, 2018, pp. 0405–0410.

[21] M. Kumm, A. Volkova, and S.-I. Filip, "Design of Optimal Multiplierless FIR Filters with Minimal Number of Adders," May 2021, working paper or preprint. [Online]. Available: https://hal.archives-ouvertes.fr/hal-02392522

[22] R. Hettich and K. O. Kortanek, "Semi-infinite programming: Theory, methods, and applications," *SIAM Review*, vol. 35, no. 3, pp. 380–429, 1993.
[23] 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)*. IEEE, Jul. 2017.
[24] A. Marendet, A. Goldsztejn, G. Chabert, and C. Jermann, "A standard branch-and-bound approach for nonlinear semi-infinite problems," *EJOR*, vol. 282, no. 2, pp. 438–452, 2020.
[25] A. Billionnet, S. Elloumi, and A. Lambert, "Linear Reformulations of Integer Quadratic Programs," in *Modelling, Computation and Optimization in Information Systems and Management Sciences*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 43–51.
[26] F. Glover, "Improved Linear Integer Programming Formulations of Nonlinear Integer Problems," *Management Science*, vol. 22, no. 4, pp. 455–460, Dec. 1975.
[27] M. Oral and O. Kettani, "A Linearization Procedure for Quadratic and Cubic Mixed-Integer Problems," *Operations Research*, vol. 40, no. 1-supplement-1, pp. S109–S116, Feb. 1992.
[28] D. Bertsimas and J. Tsitsiklis, *Introduction to linear optimization*. Belmont, Mass: Athena Scientific, 1997.
[29] O. L. Mangasarian, "Absolute value programming," *Computational Optimization and Applications*, vol. 36, no. 1, pp. 43–53, Nov. 2006.
[30] A. Antoniou, *Digital Filters: Analysis, Design, and Signal Processing Applications*. New York: McGraw-Hill Education, 2018.
[31] T. Walsh, "General Symmetry Breaking Constraints," in *Principles and Practice of Constraint Programming - CP 2006*. Springer Berlin Heidelberg, 2006, pp. 650–664.
[32] A. Goldsztejn, C. Jermann, V. Ruiz de Angulo, and C. Torras, "Variable symmetry breaking in numerical constraint problems," *Artificial Intelligence*, vol. 229, pp. 105–125, 2015.
[33] F. de Dinechin and B. Pasca, "Designing Custom Arithmetic Data Paths with FloPoCo," *IEEE Design & Test of Computers*, vol. 28, no. 4, pp. 18–27, Jul. 2011.
[34] F. Krach, B. Frackelton, J. Carletta, and R. Veillette, "FPGA-based implementation of digital control for a magnetic bearing," in *Proceedings of the 2003 American Control Conference, 2003.* IEEE, 2003.
[35] O. Sarbishei, K. Radecka, and Z. Zilic, "Analytical Optimization of Bit-Widths in Fixed-Point LTI Systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, no. 3, pp. 343–355, Mar. 2012.
[36] J. Carletta, R. Veillette, F. Krach, and Z. Fang, "Determining appropriate precisions for signals in fixed-point IIR filters," in *Proceedings 2003. Design Automation Conference*, Jun. 2003, pp. 656–661.
[37] I. Dunning, J. Huchette, and M. Lubin, "Jump: A modeling language for mathematical optimization," *SIAM Review*, vol. 59, no. 2, pp. 295–320, 2017.
[38] CPLEX, "CPLEX User's Manual," 2020. [Online]. Available: https://www.ibm.com/analytics/cplex-optimizer
[39] A. Volkova, T. Hilaire, and C. Lauter, "Reliable Evaluation of the Worst-Case Peak Gain Matrix in Multiple Precision," in *2015 IEEE 22nd Symposium on Computer Arithmetic*. IEEE, Jun. 2015.
[40] 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.
[41] K. K. Parhi, *VLSI Digital Signal Processing Systems: Design and Implementation*, ser. John Wiley & Sons. John Wiley & Sons, 00 1999.
[42] F. d. Dinechin and M. Istoan, "Automating the pipeline of arithmetic datapaths," 09 2016. [Online]. Available: https://hal.inria.fr/hal-01373937

Rémi Garcia obtained his Bachelor's Degree in Mathematics and his Master's Degree in Computer Science from University of Nantes, France, in 2016 and 2020, respectively. He joined the LS2N, France, in 2020 as a PhD student in Computer Science. His research interests include optimization applied to signal processing and computer arithmetic.

Anastasia Volkova received her Master's Degree in Applied Mathematics from Odessa National University, Ukraine, in 2014. She obtained a PhD in Computer Science from Sorbonne University in Paris, France in 2017. She was a postdoctoral researcher at Inria, France and an AI research resident at Intel Corporation, USA. 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-point and fixed-point algorithms.

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.

Alexandre Goldsztejn received the Engineer degree in computer science and mathematics from the Institut Supérieur d'Electronique et du Numérique, Lille, France, in 2001, and the Ph.D. degree in computer science from the University of Nice Sophia Antipolis, Nice, France, in 2005. He has spent one year as a Postdoctoral Fellow with the University of Central Arkansas, Conway, AR, USA, and the University of California, Irvine, CA, USA. Since 2007, he has been a full-time CNRS Researcher with the Laboratoire des Sciences du Numérique de Nantes, Nantes, France. His research interests include interval analysis and its applications to constraint satisfaction, nonlinear global optimization, robotics, and control.

Jonas Kühle received his Bachelor's degree in Applied Computer Science from University of Applied Sciences Fulda in 2017 where he is about to complete his Master's degree in 2021 and will start as a PhD student in 2022.