Design of optimal multiplierless digital filters

Rémi Garcia, Anastasia Volkova, Alexandre Goldsztejn

Design of optimal multiplierless digital filters

Rémi Garcia[1],
Supervised by
Anastasia Volkova and Alexandre Goldsztejn

Université de Nantes — LS2N
UFR Sciences – 2 rue de la Houssinière Nantes Cedex 03 – France

Abstract

In this work we solve the design of second order infinite impulse response (IIR) filters for the first time. This design is based on integer linear programming (ILP) methods. One of the underlying problem of this design is the multiple constant multiplication (MCM) problem. We propose variants to known ILP models that solve the MCM problem and illustrate our improvements with multiple examples. Our multiplierless filter design is optimal with respect to the number of adders used. Combined with MCM variants this allows addressing this optimal design in multiple contexts, in particular for a field-programmable gate array (FPGA) implementation.

Keywords: digital signal processing, filter design, multiple constant multiplication, ilp models, iir filters

1. Introduction

> In the beginning there was nothing, which exploded.
>
> Terry Pratchett

Digital circuits are replacing analog ones for many years, and analog systems find their digital counterparts. Filters are systems that perform manipulations to reduce or enhance certain characteristics of signals. Digital filters are less dependent of the environment as it is electronic. The design of such filters is of great interest for many reasons, for example, speeding up a filter allows reducing the delay between the input and the output signals. Decreasing the size of the hardware on which digital filters are implemented is of great interest, *e. g.* for embedded systems like drones. This can be done, in particular, using fixed-point arithmetic: it is a trade-off between precision and efficiency. In fixed-point arithmetic, the size of the circuit can be directly reduced by using smaller wordlengths. However, this worsen the precision and new design algorithms are in place.

Filters are designed to satisfy filter specifications. The traditional filter design process is decomposed into three steps: Find filter coefficients, quantize these coefficients and optimize the hardware implementation of the filter. However, it is hard to infer the best coefficients for hardware implementation as early as the first step. The quantization step makes unlikely the possibility to certify optimality of the hardware

Email address: Remi.Garcia@etu.univ-nantes.fr (Rémi Garcia)

implementation according to the filter specification. Given filter specifications, our goal is to directly find filter coefficients in fixed-point arithmetic. Volkova *et al.* [1] proposed a method that combines into a single procedure this whole design. Dropping the quantization and finding filter coefficients directly in fixed-point arithmetic allows combining the search for filter coefficients with the hardware optimization. This was applied for finite impulse response (FIR) filters and proved to be efficient [1] using integer linear programming (ILP) based models. Obtained solutions are certified optimal to the chosen criterion and our goal is to generalize this method to infinite impulse response (IIR) filters.

Sum of products are involved in filter evaluations. In order to be more efficient, multiplications can be replaced with bit shifts, additions and subtractions. This allows reducing the size of the circuits and speeding up the operations. The multiplierless hardware implementation of a filter through its coefficients relies on the multiple constant multiplication (MCM) problem. Solving this problem allows reducing the hardware cost and/or speed up computations. This problem was tackled in an optimal way with ILP based models by Kumm [2, 3] and efforts are being made to improve efficiency of MCM solving. To improve the hardware implementation, MCM builds on bit shifts and addition or subtraction. The cost and space needed for these operations is controlled by the fixed-point representation.

The MCM problem has been tackled in many ways and still has many layers to unfold. We participated to this by analyzing existing ILP models and proposing variants. The optimal design of FIR filters is not close to an end and any improvement is a step in the right direction. We used a similar approach for the design of second order IIR filters and overcame a few issues inherent to ILP modeling of IIR filters. Overall, the contributions of this work are:

- The generalization of a new filter design process to second order IIR filters;
- Contributions to the MCM problem.

Our work on mathematical models has been implemented and experimented using MILP solvers, as CPLEX [4] and Gurobi [5], and nonlinear programming (NLP) solvers, as SCIP [6], through JuMP [7], a package for linear and nonlinear modeling in Julia. ScaLP [8] was considered but discarded because of the absence of nonlinear modeling capabilities. For MCM those experiments showed that ILP formulations still have many interesting improvements to be think of and that NLP models are not a promising path to go on. Experimental results for the design of second order IIR filters using ILP models were encouraging and this lead is of great interest.

This work is organized into the following sections: First, the current state of the research about digital signal processing, MCM and FIR filter design is detailed in Section 2; second, in Section 3, we give an analysis of ILP formulations that are used to solve MCM problems and our improvements to these. Then, in Section 4, we extend the FIR filter design process to second order IIR filters by dealing with quadratic terms in the modeling and stability constraints. Finally, in Section 5 we propose our perspectives on leads that we think have great potential.

Notation

For the rest of this work, we will use the following notations. Intervals are denoted by $x \in [a, b]$, $]a, b]$, $[a, b[$, $]a, b[$ and correspond to $a \leq x \leq b$, $a < x \leq b$, $a \leq x < b$ and $a < x < b$ respectively. The integer interval is denoted by $[\![a, b]\!] := [a, b] \cap \mathbb{N}$, with $a, b \in \mathbb{R}$. The notation $|a|$ corresponds to the modulus if $a \in \mathbb{C} \setminus \mathbb{R}$ and to the absolute value if $a \in \mathbb{R}$. Sequences of u_n's with n the index which takes its values in X are denoted by $(u_n)_{n \in X}$. The set of natural numbers is denoted by $\mathbb{N}$. The set $\mathbb{N}^*$ corresponds to $\mathbb{N} \setminus \{0\}$.

2. Background

> Wisdom comes from experience. Experience is often a result of lack of wisdom.
>
> Terry Pratchett

2.1. Digital Filters

The design of digital filters is an area of research that emerged from the advancements of digital systems. Digital signal processing (DSP), ranging from communications to the analysis of astronomical, seismic signals, has a lot of applicable uses in our modern world. Despite increasing computational power, DSP methods still need to be improved. Actually, in order to obtain a denser time-discretization or faster computation, simply increasing computational power does not suffice and theoretical results are awaited. Furthermore, reducing the circuit area does not only rely on miniaturization advances but on theoretical results too. Systems which do manipulations on spectra of signals are called *digital filters*[1]. A filter takes an input signal, *i. e.* a sequence of numbers, and outputs a modified signal. The faster the filter computes its output the more inputs it can take and the smaller is the delay. It allows to work on a closer spectrum of the continuous signal, which is a frequent objective, and to be more reactive to the input. A signal will be seen as a sequence of numbers. Filters that we will focus on are linear time-invariant (LTI) discrete-time systems with a single input and a single output (SISO). A causal filter uses past and present inputs and past outputs to output a single value.

Constant-coefficient difference equations (CCDE) define the filter behavior. A CCDE defines the relation between an input signal $u_n \in (u_k)_{k\in\mathbb{Z}}$ and an output signal $y_n \in (y_k)_{k\in\mathbb{Z}}$, it states that the output y_n can be computed using past and present inputs and past outputs:

$$\sum_{k=0}^{N_1} b_k u_{n-k} = \sum_{k=0}^{N_2} a_k y_{n-k}, \qquad a_0 \neq 0. \tag{1}$$

By normalizing, $a_0 = 1$ can be ensured. Hence CCDE is also the next relation which explicitly shows the output at time n with respect to past data:

$$y_n = \sum_{k=0}^{N_1} b_k u_{n-k} - \sum_{k=1}^{N_2} a_k y_{n-k}. \tag{2}$$

Clearly, $\max\{N_1, N_2\}$ points in the past are needed, thus there is a delay which grows with this value which is the filter *order*. If the filter does not rely on outputs, *i. e.* $a_k = 0\ \forall k \in [\![1, N_2]\!]$, then the filter is called a *finite response impulse* (FIR) filter. If past output values are needed to compute the present output value, then the form of a such filter is called recursive or *infinite impulse response* (IIR).

For most applications, CCDE is not the method used to compute filter outputs or to find the right design. The impulse signal, denoted by δ, permits to obtain another description of a filter and is defined this way:

$$\delta(k) := \begin{cases} 1 & \text{if } k = 0 \\ 0 & \text{if } k \neq 0 \end{cases}. \tag{3}$$

[1] In the following the term *digital* will be implicit, *e. g. signal* and *filter* will stand for *digital signal* and *digital filter*.

The impulse response of a filter is its output to δ [9]. Given a filter $\mathcal{F}$, applying the z-transform [10] on its impulse response gives the *transfer function*. The transfer function denoted by H fully characterizes $\mathcal{F}$ and is equal to:

$$H(z) = \frac{b(z)}{a(z)} = \frac{\sum_{k=0}^{N_1} b_k z^{-k}}{1 + \sum_{k=1}^{N_2} a_k z^{-k}}, \tag{4}$$

where $z \in \mathbb{C}$, $b_{N_1} \neq 0$, $a_{N_2} \neq 0$, $a_k \in \mathbb{R}$, $\forall k \in [\![0, N_2]\!]$ and $b_k \in \mathbb{R}$, $\forall k \in [\![0, N_1]\!]$. Delays and past inputs or outputs are still visible in the transfer function: z^{-k} corresponds to the k-th delay of the input. In other words z^{-k} is a point in the past, the greater is k the further is this point.

The *frequency response* is the evaluation of its transfer function H on the unit circle: $H\left(e^{i\omega}\right)$, $\omega \in [-\pi, \pi]$. Designing a filter basically means constraining the frequency response for different frequencies. The design relies on *filter specifications* that are bounds, $\underline{D}$ and $\overline{D}$, on the modulus of the frequency response:

$$\underline{D}(\omega) \leq \left|H\left(e^{i\omega}\right)\right| \leq \overline{D}(\omega) \qquad \forall \omega \in [0, \pi]\,. \tag{5}$$

A *gain*, denoted by $G \in \mathbb{R}$, can be added to the previous equation:

$$G\underline{D}(\omega) \leq \left|H\left(e^{i\omega}\right)\right| \leq G\overline{D}(\omega) \qquad \forall \omega \in [0, \pi]\,. \tag{6}$$

This gain might be necessary to find filter coefficients for givens filter type and specifications. It can be fixed to one in some cases.

Example 1. *For a lowpass filter specification, low frequencies are preserved while higher frequency are suppressed, filter specifications will resemble to this:*

$$\underline{D}(\omega) = \begin{cases} 1 - \varepsilon \text{ if } \omega < \omega_1 \\ 0 \text{ if } \omega > \omega_2 \end{cases}, \tag{7}$$

$$\overline{D}(\omega) = \begin{cases} 1 + \varepsilon \text{ if } \omega < \omega_1 \\ 0 + \varepsilon \text{ if } \omega > \omega_2 \end{cases}, \tag{8}$$

where $\omega_1, \omega_2 \in [0, \pi]$ *and* $\omega_1 < \omega_2$. *For* $\omega_1 = 0.2\pi$, $\omega_2 = 0.5\pi$ *and* $\varepsilon = 0.01$ *this could lead to the 16th order FIR filter with* $(b_k)_k = (3, 6, 0, -16, -19, 12, 76, 128, 128, 76, 12, -19, -16, 0, 6, 3)$ *and a gain of* $G = 376.999805$ *[11]. This filter is represented by its transfer function in Figure 1. The passband and the stopband of this filter are* $[0, \omega_1]$ *and* $[\omega_2, \pi]$ *respectively.*

Another important aspect of filter design is the *stability*. Despite that the absence of stability can be wanted for oscillators, stable filters generally are what is needed. A filter is said to be *stable* if any bounded input results in a bounded output (BIBO). This definition does not lead to a simple way to test for stability in the design process. *Zeros* and *poles* are, respectively, the roots of the numerator and the denominator of the transfer function (4). From poles, a brief characterization of stability can be obtained which is more suitable in practice [10, Section 6.1.4]:

$$\text{a filter } \mathcal{F} \text{ is stable} \Leftrightarrow \text{poles of } \mathcal{F} \text{ are inside the complex unit circle.} \tag{9}$$

This means that a filter is stable if and only if for all λ, poles of the filter, $|\lambda| < 1$. It is unambiguous that FIR filters are inherently stable: since there are no poles, they all are inside the unit circle. On the other hand,

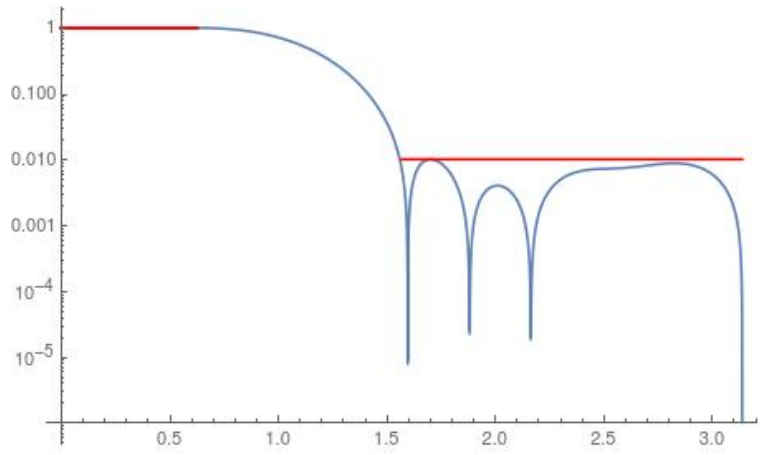


Figure 1: Example of a Low-Pass filter. In red its filter specifications moved by the gain.

IIR filters require a special attention with their design due to this stability requirement. This is usually done by focusing on pole, to ensure the stability, and zero positions as the first design step.

The design of filters can be done in compliance with field-programmable gate array (FPGA) requirements. FPGAs are integrated circuits that it is possible to reconfigured after manufacturing. Despite FPGAs have a strong potential with floating-points [12], fixed-point arithmetic is privileged for filter implementations. This permits to speed up the computation replacing multiplications, $b_k x_{n-k}$'s and $a_k y_{n-k}$'s, with faster operations, Section 2.2 will largely tackle this aspect. It is possible to convert b_k's and a_k's into fixed-point representation with a wordlength that will, in an approximate manner, be included in the filter specifications. Let d be that expected wordlength. To ease the use of optimization methods, it can be ensure that b_k's and a_k's are in the interval $[-1, 1]$. To achieve that, initial bounds must be found for b_k's and a_k's: $\underline{b}_k$'s, $\overline{b}_k$'s, $\underline{a}_k$'s and $\overline{a}_k$'s. Then, the transfer function can be "normalized" by multiplying it with:

$$\frac{2^{\left\lceil \log_2 \max\limits_{k\in[\![0,N_1]\!]} \left\{-\underline{b}_k, \overline{b}_k\right\} \right\rceil}}{2^{\left\lceil \log_2 \max\limits_{k\in[\![0,N_2]\!]} \left\{-\underline{a}_k, \overline{a}_k\right\} \right\rceil}}. \tag{10}$$

The use of log_2 and power of two permits to avoid numerical error since it basically corresponds to a shift which is possible in fixed-point arithmetic.

Furthermore, a_i's and b_i's can be multiplied by 2^d such that $b'_k = 2^d b_k$'s and $a'_k = 2^d a_i$'s are in $[\![-2^d, 2^d]\!]$. Actually, CCDE can be rewritten into the following:

$$2^d y_n = \sum_{k=0}^{N_1} b'_k x_{n-k} - \sum_{k=1}^{N_2} a'_k y_{n-k}, \tag{11}$$

where $a'_k = 2^d a_k$, $\forall k \in [\![0, N_2]\!]$ and $b'_k = 2^d b_k$, $\forall k \in [\![0, N_1]\!]$. As a consequence the fixed-point coefficient design can be formulated as an integer coefficient design.

FIR filters are the most studied filters for their robustness with respect to fixed-point implementation and, as already mentioned, their unconditional stability [10, Section 7.1.1]. However FIR filters come with two disadvantages: longer input-output delay and still a high computational cost. On the other hand, IIR filters are faster but stability is not *a priori* guaranteed thus it has to be guaranteed in the design process.

Furthermore, numerical precision is a more frequent issue than with FIR filters and the design is much more complex. Yet, if a way to control those drawbacks is found then the achieved speedup is of great interest for DSP applications. In either case we want to evaluate filters with FPGAs to gain in efficiency and to reduce the memory usage we could optimize the multiplication between the filter coefficients and the inputs and past outputs.

2.2. Multiplierless Constant Multiplication

Multiple Constant Multiplication (MCM) is a problem that consists in the speedup of the multiplication of an input number by multiple integer constants. Using generic multipliers in integrated circuits, like FPGAs, is costly since those multipliers do not have any knowledge of the predefined constants and potential improvements. The initial complexity, which is roughly a cost of n^2 with n the number of bits, can be reduced, especially in fixed-point arithmetic. In fixed-point arithmetic context we mentioned that coefficients a_k's and b_k's can be treated as integers. Next, MCM could be used to optimize the filter evaluation and finally coefficients could be converted back into their fixed-point arithmetic writing. Those integer/fixed-point conversions simply rely on bit shifts which do not degrade the numerical error in fixed-point arithmetic.

Obviously, the research started with a Single Constant Multiplication (SCM). Premisses of SCM problem were given by Bernstein in 1986 [13]. Since then, SCM has been well studied. It can be tackled using canonical signed digit (CSD) [14, 15], heuristics, bounds [16], optimal approaches [17], etc. Next, MCM problem quickly followed since it is a generalization of SCM problem and it has many applications.

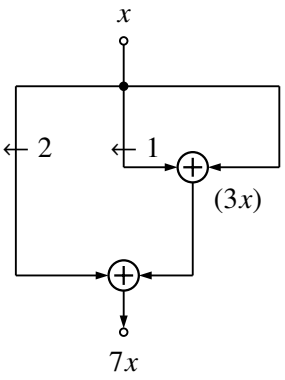
Multiplication has a computational cost that can be reduced using only "*shifts*", "*add*" and "*subtract*" instructions with dedicated hardware. A *shift* consists in the bit shift of the binary representation of the current number. A left 1-shift corresponds to the multiplication of the initial number by 2. In general a left s-shift of an integer a, written $a << s$, corresponds to a multiplication of a by 2^s. This operation can be hardwired on FPGAs thus its cost remains negligible. A first straightforward example of a speedup that can be obtained by MCM problem is the replacement of the multiplication by 16 with a 4-shift, multiply by 16 does not have a negligible cost while a shift does. However, 17 or 15 cannot be obtained using only shifts. *Add* and *subtract* operations combined with shifts permit to obtain any integer. Yet, those operations have a cost, therefore the overall objective is to replace the initial multiplication with a minimum number of add and subtract operations to reduce the total cost of the multiplications. Those operations, additions and subtractions, will be referred to as *adders*. In other words, an adder takes two inputs and computes the sum or subtraction of those. To be more visual we will say that an adder has a left input and a right input, this should be clearly visible in figures along the text. Both operations have an equivalent cost, hence SCM and MCM consist in the rewriting of a multiplication using a minimum number of adders and as many shifts as needed. Furthermore, since multiple constants are involved in MCM problem it is possible to share adders between constants to speed up the multiplications even more.

An easy way to rewrite any multiplication is using the binary notation and deducing shifts and adds from it, *e. g.* for $7x$:

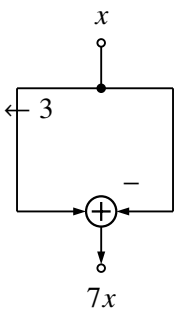
$$7x = 0111_2 x = 2^2 x + 2^1 x + 2^0 x = (x << 2) + (x << 1) + (x << 0)\,, \tag{12}$$

so two adders are required to compute $7x$ this way. This corresponds to the number of 1's in their binary representation minus one. The corresponding circuit is shown in Figure 2a.

There are other ways to represent numbers such as signed digit (SD) or canonical signed digit (CSD) representation [14]. A ternary representation system in which $\bar{1}$ corresponds to $-2^{\text{bit position}}$ is called SD representation. It is direct that a similar addition/subtraction and shift circuit naturally derives from this representation. A non-adjacent form is the particular case in which no two non-zero values are adjacent.



(a) Binary multiplication



(b) CSD multiplication

Figure 2: Shifts are represented with labeled arrows and adders as $\bigoplus$ with an optional minus on the left or the right side.

The CSD representation[2] is a ternary representation system in which only non-adjacent forms are allowed. For a given integer its CSD representation is unique and can be computed in polynomial time, the CSD representation of 7 is:

$$100\overline{1}_{CSD}. \tag{13}$$

Figure 2b shows the corresponding circuits to compute $7x$, which is optimal. This representation gives a minimal Hamming weight (the number of non-zero values in the representation), and therefore it permits to obtain a good bound on the number of adders in general. Although CSD representations have this important property, it does not ensure a minimal number of adders, *e. g.* with an example from [3, p. 9–11]:

$$93x = 0101\ 1101_2 x = 10\overline{1}0\ 0\overline{1}01_{CSD}x, \tag{14}$$

naturally uses three adders. Yet,

$$93x = ((x << 1) + x) + (((x << 1) + x) << 5)\,, \tag{15}$$

and despite three addition terms, the term $(x << 1) + x$ is redundant thus can be computed only one time and used twice as shown in Figure 3: two adders suffice. This is why solving SCM problem is needed. However, when multiple constants are involved, the solving of multiple SCM problems does not give an optimal solution as shown in Figure 4.

Usually, MCM problem is presented in its decision form:

$$\textit{Given multiple integers, is there an adder graph that ouputs these integers with N adders,} \tag{16}$$

where an *adder graph* is an adder circuit for constant multiplication. Adder graphs are directed acyclic graph (DAG) and Figures 2, 3 and 4 are examples of such graphs. Each node, except the input node, corresponds to an adder. Nodes have an in-degree of two. Each edge weight, which can go along with an arrow, represents a bit shift. No weight or zero means the absence of a bit shift. Each node has an associated value which corresponds to the multiple computed in the node, this constant is called *fundamental*. It can be desirable to minimize the number of adders N instead of dealing with a decision problem: this gives an optimization problem. It should be noted that both problems, the decision one and its minimization counterpart, have similar complexities.

[2]Sometimes this is referred to as *Booth's recoding* or *Booth's encoding*, *e. g.* in [18, 14]

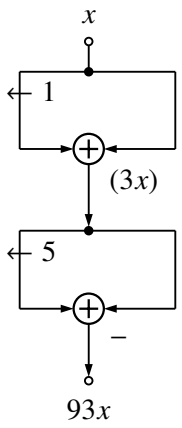


Figure 3: DAG of 93 in CSD representation needs three adders while minimum adder realization only takes two.

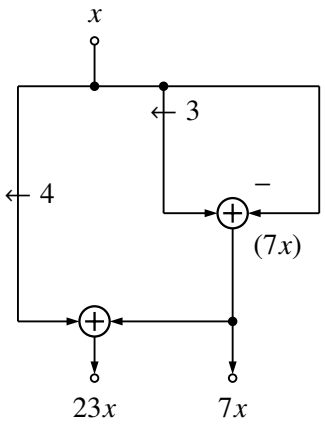


Figure 4: One adder is needed for $7x$ and $23x$ needs two. Sum of SCMs needs three while an optimal solution for MCM only needs two in total.

We focus our efforts on exact approaches, precisely on ones that use ILP based models to solve this problem [2, 3]. In particular, two ILP based models were developed. The first model will be referred to as ILP Formulation 1 and the second is denoted by ILP Formulation 2. ILP Formulation 1 tackles the theoretical MCM problem while ILP Formulation 2 deals with a more practicable version of MCM problem in which a constraint on the *adder depth* is added. We present here this variant and a few useful terms.

2.2.1. Terminology

The adder depth, denoted by AD(·), is a property of nodes in an adder graph. It is the delay before a node is computed. The adder depth of the input node is 0. Let c be a node, with inputs n_1 and n_2, its adder depth is defined as follows:

$$\mathrm{AD}(c) = \max\{\mathrm{AD}(n_1), \mathrm{AD}(n_2)\} + 1. \tag{17}$$

The adder depth of a graph is the maximum of its nodes' adder depth. This metric is essential for two reasons: First, note that computation time is proportional to the adder depth; second, a greater adder depth might imply more additions and subtraction to compute a given constant, thus a bigger numerical error inherent to computer arithmetic. For example it is clear that in Figure 2a the second adder depends on the first adder's computation: Second adder has to wait for the first one and the computation error that the first adder outputs is passed on and amplified by the second. For these reasons minimizing the adder depth as a second objective in MCM problem can be considered. This variant is denoted $\mathrm{MCM}_{\mathrm{MAD}}$. Furthermore, bounding the adder depth, *i. e.* adding a constraint on the problem, allows new possible ways to speed up the resolution, *e. g.* it naturally limits the time-delay. This problem, which is the one that is useful for FPGAs, will be referred to as $\mathrm{MCM}_{\mathrm{BAD}}$. This last variant is the one covered by ILP Formulation 2.

Remark. *The adder depth of DAGs represented Figures 2a and 3 is two while the adder depth of the DAG represented Figure 2b is equal to one.*

An *odd fundamental graph* is an adder graph for which all fundamentals are odd. Dempster and Macleod showed that only odd fundamental graphs can be considered without any loss of generality [19]. Additionally, any MCM instance can be transformed into an equivalent MCM problem with only odd positive constants, this transformed MCM problem will be referred to as $\mathrm{MCM}_{\mathrm{odd}}$ problem. Consequently, solving MCM problem in this restricted scope is possible and reduce the search space. Reducing the search space might speed up the solving and $\mathrm{MCM}_{\mathrm{odd}}$ can be combined with other variants. For that reason, in general by

MCM we will refer to MCM_{odd}. The following provides understanding of how MCM_{odd} problem is derived from an instance of MCM problem. Let $T \subset \mathbb{Z}$ be the set of target constants of a given MCM problem,

$$T_{\text{odd}} := \left\{ t \in \mathbb{N} \mid t \bmod 2 = 1 \wedge \exists k \in \mathbb{N} \text{ such that } 2^k t \in T \right\}, \tag{18}$$

is the set of constants for the equivalent MCM_{odd} instance. This set of positive odd constants T_{odd} corresponds to the elements of T divided by two until they are odd numbers. It can be rewritten in a more compact and readable way using the *odd part* notion [20, Sequence A000265]. The odd part of a positive integer n is the biggest odd number that divides n. It is written $\text{odd}(n)$ and can be defined as follows:

$$\text{odd}(n) := \frac{n}{\gcd(2^n, n)}. \tag{19}$$

The odd part of a number n is computed by dividing n by two until the obtained number is odd. With that definition of $\text{odd}(n)$ it can be deduced that:

$$T_{\text{odd}} = \{\text{odd}(|t|) \mid t \in T\}. \tag{20}$$

Example 2. *Given a type II low-pass FIR filter with* 3, 0, −25, 0, 150, 256, 150, 0, −25, 0, 3 *its coefficients, target constants,*

$$T = \{0, 3, -25, 150, 256\}, \tag{21}$$

becomes,

$$T_{\text{odd}} = \{0, 1, 3, 5, 75\}. \tag{22}$$

In both cases, 0 *can be exclude from the target constants, in the second case* 1 *can be exclude too: It corresponds to the absence of shifts and adders. The remaining set to work on is smaller, both in size and in maximum value.*

2.2.2. ILP Formulation 1

Kumm gave in [2] a first model for MCM problem: ILP Formulation 1. This model has no objective function, it corresponds to the decision problem. However it was done to minimize the number of adders, to tackle the optimization problem. Hence, the solving process is embedded in a while loop which generates a model with a fixed number of adders and solves it. While it is infeasible, it increases the number of adders and repeats. An optimal solution for the optimization problem is found when a model has a solution, *i. e.* when a decision problem has a solution. The initial number of adders could be one or any greater known lower bound of the given instance.

Formulations are defined by a set of data, that partly depends on each instance and solving choices, a set of variables and a set of constraints. Eventually, an objective might be added. The input data and variables

Table 1: Constants (top) and variables (bottom) used for ILP Formulation 1

Constants/Variables
$N_A \in \mathbb{N}$: number of adders;
$N_O \in \mathbb{N}$: number of outputs;
$C \in \mathbb{N}^{N_O}$: target constants;
$S_{\min}, S_{\max} \in \mathbb{Z}$: minimum and maximum shift;
$d \in \mathbb{N}$: wordlength.
$c_a \in [\![0, 2^d]\!]$, $\forall a \in [\![0, N_A]\!]$: constant obtained in adder a;
$c_{a,i} \in [\![0, 2^d]\!]$, $\forall a \in [\![1, N_A]\!]$, $i \in \{l, r\}$: constant of adder from input i (left or right) before adder a;
$c_{a,i}^{\text{sh}} \in [\![0, 2^d]\!]$, $\forall a \in [\![1, N_A]\!]$, $i \in \{l, r\}$: constant of adder from input i before adder a after the shift;
$c_{a,i}^{\text{sh,sg}} \in [\![-2^d, 2^d]\!]$, $\forall a \in [\![1, N_A]\!]$, $i \in \{l, r\}$: signed constant of adder from input i before adder a after the shift;
$\Phi_{a,i} \in \{0, 1\}$, $\forall a \in [\![1, N_A]\!]$, $i \in \{l, r\}$: sign of i input of adder a. 0 for + and 1 for −;
$c_{a,i,k} \in \{0, 1\}$, $\forall a \in [\![1, N_A]\!]$, $i \in \{l, r\}$, $k \in [\![0, N_A - 1]\!]$: 1 if input i of adder a is adder k, 0 otherwise;
$\phi_{a,i,s} \in \{0, 1\}$, $\forall a \in [\![1, N_A]\!]$, $i \in \{l, r\}$, $s \in [\![S_{\min}, S_{\max}]\!]$: 1 if shift on input i before adder a is equal to s, 0 otherwise;
$o_{a,j} \in \{0, 1\}$, $\forall a \in [\![1, N_A]\!]$, $j \in [\![1, N_O]\!]$: 1 if adder a is equal to the j-th target constant, 0 otherwise.

of ILP Formulation 1 are defined Table 1. Then, constraints follow:

$$c_0 = 1 \tag{C1.1}$$

$$c_a = c_{a,l}^{\text{sh,sg}} + c_{a,r}^{\text{sh,sg}} \qquad \forall a \in [\![1, N_A]\!] \tag{C1.2}$$

$$c_{a,i} = c_k \text{ if } c_{a,i,k} = 1 \qquad \forall a \in [\![1, N_A]\!], i \in \{l, r\}, \forall k \in [\![0, a-1]\!] \tag{C1.3}$$

$$\sum_{k=0}^{a-1} c_{a,i,k} = 1 \qquad \forall a \in [\![1, N_A]\!], i \in \{l, r\} \tag{C1.4}$$

$$c_{a,i}^{\text{sh}} = 2^s c_{a,i} \text{ if } \phi_{a,i,s} = 1 \qquad \forall a \in [\![1, N_A]\!], i \in \{l, r\}, s \in [\![S_{\min}, S_{\max}]\!] \tag{C1.5}$$

$$\sum_{s=S_{\min}}^{S_{\max}} \phi_{a,i,s} = 1 \qquad \forall a \in [\![1, N_A]\!], i \in \{l, r\} \tag{C1.6}$$

$$\phi_{a,l,s} = 0 \qquad \forall s \in [\![1, S_{\max}]\!], a \in [\![1, N_A]\!] \tag{C1.7}$$

$$\phi_{a,l,s} = \phi_{a,r,s} \qquad \forall s \in [\![S_{\min}, -1]\!], a \in [\![1, N_A]\!] \tag{C1.8}$$

$$c_{a,i}^{\text{sh,sg}} = -c_{a,i}^{\text{sh}} \text{ if } \Phi_{a,i} = 1 \qquad \forall a \in [\![1, N_A]\!], i \in \{l, r\} \tag{C1.9}$$

$$c_{a,i}^{\text{sh,sg}} = c_{a,i}^{\text{sh}} \text{ if } \Phi_{a,i} = 0 \qquad \forall a \in [\![1, N_A]\!], i \in \{l, r\} \tag{C1.10}$$

$$\Phi_{a,l} + \Phi_{a,r} \leq 1 \qquad \forall a \in [\![1, N_A]\!] \tag{C1.11}$$

$$c_a = C_j \text{ if } o_{a,j} = 1 \qquad \forall a \in [\![0, N_A]\!], j \in [\![1, N_O]\!] \tag{C1.12}$$

$$\sum_{a=0}^{N_A} o_{a,j} = 1 \qquad \forall j \in [\![1, N_O]\!] \tag{C1.13}$$

Constraint (C1.2) states that the value of an adder is equal to the addition of its inputs. Constraints (C1.3) and (C1.5) state that the unsigned i-th input of an adder a is equal to the s shifted constant of the adder k if the input i of adder a is adder k and is shifted by s. Constraints (C1.4) and (C1.6) state that for each input of each adder there is only one previous adder and it has been shifted only once – a 0-shift is allowed. Constraints (C1.7) and (C1.8) ensure that one input is positive shifted while the other shift is zero or that both have an identical negative shift, this is sufficient and reduces the exploration space [19, Th. 3]. Constraints (C1.9), (C1.10) and (C1.11) apply the sign for each input and ensure that only one input per adder is negative. This last constraint is redundant since $c_a \in [\![0, 2^d]\!]$ would be contradictory with $c_a = c_{a,l}^{\text{sh,sg}} + c_{a,r}^{\text{sh,sg}}$ if $c_{a,l}^{\text{sh,sg}}$ and $c_{a,r}^{\text{sh,sg}}$ are both negative. Finally (C1.12) fixes the value of adder a to the j-th constant and (C1.13) ensures that each constant has exactly one adder that computes it.

Note that this model is not linear: implications are not linear constraints and are present in constraints (C1.3), (C1.5), (C1.9), (C1.10) and (C1.12). However common mixed integer linear programming (MILP) solvers, as CPLEX or Gurobi, handle those constraints very well and this permits to avoid using big M [21]. Yet, the five constraints involving implications could have been replaced by two big M constraints each:

$$c_{a,i} \geq c_k + (c_{a,i,k} - 1) \times M \qquad \forall a \in [\![1, N_A]\!], i \in \{l, r\}, \forall k \in [\![0, a-1]\!] \qquad \text{(C1.3a)}$$

$$c_{a,i} \leq c_k + (1 - c_{a,i,k}) \times M \qquad \forall a \in [\![1, N_A]\!], i \in \{l, r\}, \forall k \in [\![0, a-1]\!] \qquad \text{(C1.3b)}$$

$$c_{a,i}^{\text{sh}} \geq 2^s c_{a,i} + (\phi_{a,i,s} - 1) \times M \qquad \forall a \in [\![1, N_A]\!], i \in \{l, r\}, s \in [\![S_{\min}, S_{\max}]\!] \qquad \text{(C1.5a)}$$

$$c_{a,i}^{\text{sh}} \leq 2^s c_{a,i} + (1 - \phi_{a,i,s}) \times M \qquad \forall a \in [\![1, N_A]\!], i \in \{l, r\}, s \in [\![S_{\min}, S_{\max}]\!] \qquad \text{(C1.5b)}$$

$$c_{a,i}^{\text{sh,sg}} \geq -c_{a,l}^{\text{sh}} + (\Phi_{a,i} - 1) \times M \qquad \forall a \in [\![1, N_A]\!], i \in \{l, r\} \qquad \text{(C1.9a)}$$

$$c_{a,i}^{\text{sh,sg}} \leq -c_{a,l}^{\text{sh}} + (1 - \Phi_{a,i}) \times M \qquad \forall a \in [\![1, N_A]\!], i \in \{l, r\} \qquad \text{(C1.9b)}$$

$$c_{a,i}^{\text{sh,sg}} \geq c_{a,l}^{\text{sh}} - (\Phi_{a,i}) \times M \qquad \forall a \in [\![1, N_A]\!], i \in \{l, r\} \qquad \text{(C1.10a)}$$

$$c_{a,i}^{\text{sh,sg}} \leq c_{a,l}^{\text{sh}} + (\Phi_{a,i}) \times M \qquad \forall a \in [\![1, N_A]\!], i \in \{l, r\} \qquad \text{(C1.10b)}$$

$$C_j \geq c_a + \left(o_{a,j} - 1\right) \times M \qquad \forall a \in [\![0, N_A]\!], j \in [\![1, N_O]\!] \qquad \text{(C1.12a)}$$

$$C_j \leq c_a + \left(1 - o_{a,j}\right) \times M \qquad \forall a \in [\![0, N_A]\!], j \in [\![1, N_O]\!] \qquad \text{(C1.12b)}$$

An equality constraint in an implication constraint has to be decomposed into two inequality constraints with big M. One can replace the equal symbol and add a big M which is activated or not depending of the binary variable used for the implication. Constraint (C1.3) derives into constraints (C1.3a) and (C1.3b). For constraint (C1.3a), if $c_{a,i,k} = 0$ then $c_{a,i} \geq c_k + (c_{a,i,k} - 1) \times M$ is inactivated because of the big M that, in a way, implies $c_{a,i} \geq -\infty$. However, since M is finite, the real implication of $c_{a,i,k} = 0$ is $c_{a,i}$ has to be greater than something smaller than its lower bound. The same reasoning holds for constraint (C1.3b). On the other hand, if $c_{a,i,k} = 1$ then $0 \times M$ is removed from both inequalities which becomes $c_{a,i} \geq c_k$ and $c_{a,i} \leq c_k$, thus $c_{a,i} = c_k$.

Kumm chose to use implications to avoid numerical instability in the solving process as it is suggested by Klotz and Newman [21, Section 3.4.]. Default integrality tolerance on Gurobi and CPLEX is equal to 10^{-5}. Hence, only $M > 10^5$ might lead to numerical instability. Since $M < 2^{2d}$ with d denoting the wordlength, default integrality tolerance is enough for wordlengths up to 8bit. Lowering integrality tolerance is possible and safe down to 10^{-9}. This allows wordlengths up to 14bit but with a downside: an increasing computing time. For any larger wordlengths, big M constraints might not be usable. We give a

more detailed analysis in Section 3.1.2 and a solution is proposed to bound big M by 2^d: with the drawback of creating more variables, this allows bigger wordlengths using big M constraints.

2.2.3. ILP Formulation 2

ILP Formulation 2 tackles $\mathrm{MCM}_{\mathrm{BAD}}$ and can be looked as a shortest-path in a hypergraph. Indeed, since the adder depth is bounded, the total number of possible values that can be obtained with shift, add and subtract operations is reduced. All those values can be computed: each value is a vertex and each way to compute that value a hyperedge. First, we will give a few mathematical tools that are necessary for the comprehension of that formulation. Hypergraphs can be constructed with these notions. Second, a shortest-path solution, taking the form of an ILP based model presented after, is applied.

An $\mathcal{A}$-operation is the operation that produces the fundamental of each node depending on its inputs $u, v \in \mathbb{N}$ and on an $\mathcal{A}$-configuration $q = (l_u, l_v, r, s)$. The integers $l_u, l_v \in \mathbb{N}$ correspond to the left shifts of u and v respectively, $r \in \mathbb{N}$ is the output right shift and $s \in \{0, 1\}$ stands for addition or subtraction. This can be synthesized into a compact definition:

$$\mathcal{A}_q(u, v) = \left|2^{l_u}u + (-1)^s\, 2^{l_v}v\right| 2^{-r}. \tag{23}$$

This definition can be related to the computations of nodes of adder graphs where $\mathcal{A}_q(u, v)$ is the fundamental computed in a node from input nodes u and v.

It is said that $u, v, w \in \mathbb{N}$ are in an $\mathcal{A}$-relation if $\exists q$ an $\mathcal{A}$-configuration such that $w = \mathcal{A}_q(u, v)$. The definition of $\mathcal{A}$-configuration is adapted to only allow odd fundamentals, which simplifies its formulation without any loss for MCM problem comprehension and modeling. The statement "q is an $\mathcal{A}$-configuration" will refer, somewhat imprecisely, to $q = (l_u, l_v, r, s)$ is a *valid* configuration such that $\mathcal{A}_q(u, v)$ is an odd integer smaller than a given $c_{\max}$. The statement "q is an $\mathcal{A}$-configuration" is imprecise since $\mathcal{A}$-configuration directly depends on the context (u, v and $c_{\max}$). In our MCM problem context with d a fixed wordlength we will always assume that $c_{\max} = 2^d$. This definition constrains r: For l_u, l_v and s fixed, exactly one r gives rise to a valid $\mathcal{A}$-configuration. This is direct to prove using a reductio ad absurdum argument. From $\mathcal{A}$-operation, $\mathcal{A}$-sets naturally derive. This notion can be applied either on integers or on sets of integers. The $\mathcal{A}_*(u, v)$-set, with $u, v \in \mathbb{N}$, contains all odd fundamentals which can be obtained from each $\mathcal{A}$-operation on (u, v). It corresponds to all the integers that are in an $\mathcal{A}$-relation with u and v:

$$\mathcal{A}_*(u, v) := \left\{\mathcal{A}_q(u, v) \mid q \text{ an } \mathcal{A}\text{-configuration}\right\}. \tag{24}$$

Remark. *Without a* $c_{\max}$, $\mathrm{card}(\mathcal{A}_*(u, v)) = \infty$, i. e. *there is an infinite number of odd fundamentals that can be computed from two fundamentals.*

The $\mathcal{A}_*(U, V)$-set, with $U, V \subset \mathbb{N}$, contains all odd fundamentals which can be obtained from every $\mathcal{A}$-operations on each pair $(u, v) \in U \times V$:

$$\mathcal{A}_*(U, V) := \bigcup_{(u,v)\in U\times V} \mathcal{A}_*(u, v)\,. \tag{25}$$

Let X be a set of integers, concisely $\mathcal{A}_*(X, X)$ will be written $\mathcal{A}_*(X)$, hence:

$$\mathcal{A}_*(X) = \left\{\mathcal{A}_q(u, v) \mid u, v \in X, q \text{ an } \mathcal{A}\text{-configuration}\right\}. \tag{26}$$

Example 3. *For* $c_{\max} = 2^4$ *we have:*

$$\begin{aligned}\mathcal{A}_*(\{1\}) &= \left\{\mathcal{A}_q(1, 1) \mid q \textit{ an } \mathcal{A}\textit{-configuration}\right\}\\ &= \{1, 3, 5, 7, 9, 15\}\,.\end{aligned}$$

Table 2: Constants (top) and variables (bottom) used for ILP Formulation 1

Constants/Variables
$a_w^s \in \{0,1\}$, $\forall s \in [\![1, S]\!]$, $w \in \bigcup \mathcal{S}$: 1 if node w is computed by an adder at depth s, 0 otherwise. a_w^s corresponds to adders;
$r_w^s \in \{0,1\}$, $\forall s \in [\![1, S]\!]$, $w \in \bigcup \mathcal{S}$: 1 if node w was computed at a depth strictly lower than s, 0 otherwise. r_w^s are called registers;
$x_{(u,v)}^s \in \{0,1\}$, $\forall s \in [\![1, S]\!]$, $u, v \in \bigcup \mathcal{S}$: 1 if both u and v are available in depth s, 0 otherwise.

It should be noted that computing $\mathcal{A}_*(\{1\})$ with $c_{\max}$ equal to a power of 2 is recurrent in SCM and MCM problems. This is the first step to compute $\mathcal{A}_* \circ \mathcal{A}_* \circ \cdots \circ \mathcal{A}_*(\{1\})$.

Let $\mathcal{A}^s$ be the set of odd fundamentals that can be computed in depth s of the tree hypergraph, *e. g.* for an adder depth that is bounded by four, which is fine in practice, $\mathcal{A}^4 = \mathcal{A}_*(\mathcal{A}_*(\mathcal{A}_*(\mathcal{A}_*(\{1\}))))$ corresponds to the nodes of the hypergraph. $\mathcal{A}^s$ are computed using $\mathcal{A}$-sets:

$$\mathcal{A}^0 := \{1\} \tag{27}$$

$$\mathcal{A}^s := \mathcal{A}_*\left(\mathcal{A}^{s-1}\right) \qquad \forall s \in \mathbb{N}^* \tag{28}$$

The size $\mathcal{A}^s$ increases quite fast with the wordlength, until all representable odd fundamentals are produced. Kumm gave the size of $\mathcal{A}$-sets for different adder depth and wordlengths [3]. For wordlengths less to 16 an adder depth of four is sufficient to compute all the possible odd fundamentals. These results were obtained by explicitly compute $\mathcal{A}^s$ for different wordlengths: There is no known simple formula for $\text{card}(\mathcal{A}^s)$. The size of the generated hypergraph can, and should, be reduced before using the solver: $\mathcal{S}^s \subseteq \mathcal{A}^s$ denotes the odd fundamentals that may be useful to compute the odd target constants of the given instance, T_{odd}. The set $\mathcal{T}^s$ denotes the set of triplets (u, v, w) such that u, v and w are in an $\mathcal{A}$-relation, $u \leq v$ and $u, v \in \mathcal{A}^s$. The set $\mathcal{S}^s$ and $\mathcal{T}^s$ are both computed recursively with S the maximum possible adder depth:

$$\mathcal{S}^S := T_{\text{odd}} \tag{29}$$

$$\mathcal{T}^{s-1} := \left\{(u, v, w) \mid w = \mathcal{A}_q(u, v),\ u, v \in \mathcal{A}^{s-1},\ u \leq v,\ w \in \mathcal{S}^s\right\} \qquad \forall s \in \mathbb{N}^* \tag{30}$$

$$\mathcal{S}^{s-1} := \left\{u, v \mid (u, v, w) \in \mathcal{T}^{s-1}\right\} \qquad \forall s \in \mathbb{N}^* \tag{31}$$

ILP Formulation 2 can be derived from these constants. Its variables are presented Table 2.

The objective is to minimize the number of adder that are used, *i. e.* the objective is to minimize the number of variables a_w^s that are equal to one:

$$\min \sum_{s=1}^{S} \sum_{w \in \mathcal{S}^s} a_w^s \tag{32}$$

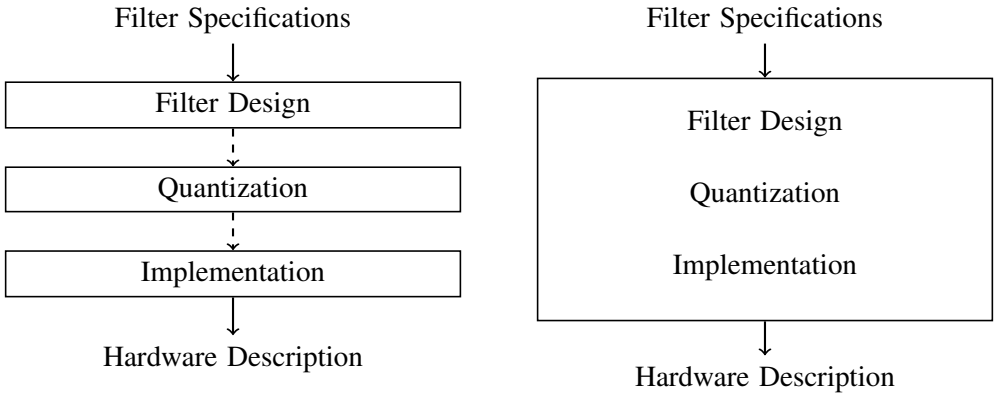


Figure 5: Filter design for hardware, classic flow (left) and proposed flow (right)

Constraints:

$$r_w^S + a_w^S = 1 \qquad \forall w \in T_{\text{odd}} \tag{C2.1}$$

$$r_w^s = 0 \qquad \forall s \in [\![1, S-1]\!], w \in \mathcal{S}^s \setminus \bigcup_{s'=0}^{s-1} \mathcal{S}^{s'} \tag{C2.2}$$

$$r_w^s - a_w^{s-1} - r_w^{s-1} \leq 0 \qquad \forall s \in [\![2, S]\!], w \in \mathcal{S}^s \tag{C2.3}$$

$$a_w^s - \sum_{\substack{u,v \in \mathcal{S}^{s-1} \\ (u,v,w) \in \mathcal{T}^s}} x_{(u,v)}^{s-1} \leq 0 \qquad \forall s \in [\![2, S]\!], w \in \mathcal{S}^s \tag{C2.4}$$

$$x_{(u,v)}^s - r_u^s - a_u^s \leq 0 \qquad \forall s \in [\![1, S-1]\!], u, v \in \mathcal{S}^{s-1}, u \leq v \tag{C2.5}$$

$$x_{(u,v)}^s - r_v^s - a_v^s \leq 0 \qquad \forall s \in [\![1, S-1]\!], u, v \in \mathcal{S}^{s-1}, u \leq v \tag{C2.6}$$

Constraint (C2.1) ensures that, for each target constant, an adder or a register computes it. Constraint (C2.2) initializes registers. Constraint (C2.3) avoids that a register takes a value at a stage if no register or adder already contained this value at the previous stage. In order for an adder to take a specific value w, then u and v such that $\mathcal{A}_*(u, v) = w$ have to be available at previous stage; to be available at a given stage, u and v have to be in a register or an adder: constraints (C2.4), (C2.5) and (C2.6) ensure that.

2.3. Design of filters

Design of filter is usually done in three steps [22]: First, the filter coefficients are designed using floating-point arithmetic; second, the quantization process is applied, roughly it corresponds to the rounding of computed coefficients; third, the implementation minimizing the hardware cost. Volkova *et al.* [1] proposed another approach that combined all the steps into a single process. Figure 5 illustrates this change of approach. For FIR filters it was recently shown that the whole block can be expressed through the form of a single ILP model. To do so, the filter design which corresponds to the filter specification constraints, is conveyed by the zero-phase frequency response [9]:

$$H_R(\omega) = \sum_{m=0}^{M-1} h_m c_m(\omega), \tag{33}$$

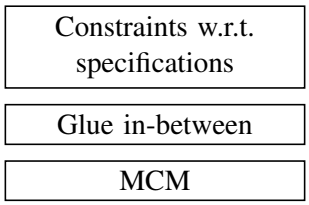


Figure 6: High-level overview of FIRopt ILP model

Table 3: Trigonometric functions c_m depend on the parity of the filter order and its symmetry. Both are deduced from the filter type.

Filter type	Filter order	Symmetry	c_m
I	Even	Symmetric	$c_m(\omega) = \begin{cases} 1 & \text{if } m = 0 \\ 2\cos(\omega m) & \text{if } m \geq 1 \end{cases}$
II	Odd	Symmetric	$c_m(\omega) = 2\cos\left(\omega\left(m + \frac{1}{2}\right)\right)$
III	Even	Asymmetric	$c_m(\omega) = 2\sin(\omega\,(m + 1))$
IV	Odd	Asymmetric	$c_m(\omega) = 2\sin\left(\omega\left(m + \frac{1}{2}\right)\right)$

with $\omega \in [0, \pi]$, M the number of independent coefficients and c_m trigonometric functions defined Table 3 that depend on filter order and its potential symmetry. The absolute value of the zero-phase frequency response is equal to the modulus of the transfer function on the unit circle, *i. e.*,

$$|H_R(\omega)| = \left|H\left(e^{i\omega}\right)\right|, \qquad \omega \in [0, \pi]\,. \tag{34}$$

Coefficients h_m's are target constants to optimize, this is done using MCM through its ILP formulations. For the MCM problem, target constants were input data. On the other hand, they are not *a priori* known in this one-block approach. To link together filter specifications and MCM problem, a few constraints are added. This is the glue in-between. In a way, Figure 6 represents this model. To design FIR filters using this approach an open-source tool, FIRopt, has been proposed [23]. We aim to propose a similar method and tool for IIR filters which, for now, are designed using the same usual three steps process [22].

3. Multiple Constant Multiplication

> It's still magic even if you know how it's done.
>
> Terry Pratchett

In this section, MCM problem is studied in several ways: ILP formulations, size of models, possibility of NLP models, etc. This allows a deeper understanding of the MCM problem and improvements of initial ILP formulations with original approaches.

3.1. Linear Models

3.1.1. Analysis of the two existing models

ILP Formulation 1 has no objective function, this has a strong impact on whether implications are efficient or not [21]. Typically, constraints with implications are dropped in the linear relaxation of the

model. This affects the quality of the linear relaxation, hence, in the branch and bound process, less nodes are bounded by the linear relaxation. Note that when there is no objective function the bound process is not relevant. As we will add objective functions to the ILP Formulation 1, experimental results will show this effect of implications or big M choice.

Solving MCM problem with an adder graph or an odd adder graph is equivalent [19]. ILP Formulation 1 uses this property by limiting possible shifts, (C1.7) and (C1.8), and ILP Formulation 2 relies on this, through $\mathcal{A}$-operation. We propose to use further this property in ILP Formulation 1 by removing negative shifts from inputs and by having no more than one input with a positive shift. Hence, we will suppose that left shifts are dropped. However, to match with [19, Theorem 3] a negative shift should be allowed after the addition (or subtraction): a negative shift inside the adder. To do so, sets of new variables are needed:

- $c_a^{\text{nsh}} \in [\![0, 2^d]\!]$, $\forall a \in [\![1, N_A]\!]$: constant obtained in adder a before an eventual shift;
- $c_a^{\text{odd}} \in \mathbb{N}$, $\forall \in [\![1, N_A]\!]$: allows forcing c_a to be odd;
- $\Psi_{a,s} \in \{0, 1\}$, $\forall a \in [\![1, N_A]\!]$, $s \in [\![S_{\min}, 0]\!]$: 1 if a s shift is applied to adder a, 0 otherwise.

Constraint (C1.2) is replaced by $c_a^{\text{nsh}} = c_{a,l}^{\text{sh,sg}} + c_{a,r}^{\text{sh,sg}}$, then constraints $c_a = 2c_a^{\text{odd}} + 1$ are added to force constants to be odd. Finally, a new set of constraints links c_a with c_a^{nsh} and $\Psi_{a,s}$:

$$c_a^{\text{nsh}} = 2^{-s} c_a \text{ if } \Psi_{a,s} = 1 \qquad \forall a \in [\![1, N_A]\!], s \in [\![S_{\min}, 0]\!] \tag{35}$$

$$\sum_{s=S_{\min}}^{0} \Psi_{a,s} = 1 \qquad \forall a \in [\![1, N_A]\!] \tag{36}$$

or, using big M constraints,

$$c_a^{\text{nsh}} \geq 2^{-s} c_a + (\Psi_{a,s} - 1) \times M \qquad \forall a \in [\![1, N_A]\!], s \in [\![S_{\min}, 0]\!] \tag{37}$$

$$c_a^{\text{nsh}} \leq 2^{-s} c_a + (1 - \Psi_{a,s}) \times M \qquad \forall a \in [\![1, N_A]\!], s \in [\![S_{\min}, 0]\!] \tag{38}$$

$$\sum_{s=S_{\min}}^{0} \Psi_{a,s} = 1 \qquad \forall a \in [\![1, N_A]\!] \tag{39}$$

This big M is bounded by 2^{2d} and can be adjusted to 2^d adding more variables and constraints, as detailed in Section 3.1.2.

It can be noticed that a shift inside an adder only makes sense if the sum of the inputs is even. Yet, the left input is odd since left shifts are dropped. Thus, the sum of the inputs is even if and only if the right input is odd. Hence, the sum of the inputs is even if and only if there is no right shift. From that, a constraint can be deduced to speed up the resolution:

$$\phi_{a,r,0} = \sum_{s=S_{\min}}^{-1} \Psi_{a,s} \qquad \forall a \in [\![1, N_A]\!]. \tag{40}$$

ILP Formulation 1 modified with previous variables and constraints will be referred to as ILP Formulation 1 with odd fundamentals only, or abbreviated ILP Formulation 1odd.

Example 4. *The target set* $\{7, 19, 31\}$ *can be obtained with three adders. First,* 7 *and* 31 *are computed as a power of two (a left shift of one) minus one. Second,* 19 *corresponds to* $7 + 31$ *with a negative shift inside the adder. The variable* c_a^{nsh} *would be equal to* 38 *and the inside shift* $\Psi_{a,s}$ *would be equal to one for* $s = -1$*, thus* c_a *would be equal to* $38 << -1 = 19$*.*

Despite the combinatorial dimension, ILP Formulation 1 (and ILP Formulation 1odd) remains relatively small: the number of variables and constraints is quadratic on the number of adders, the number of outputs and the wordlength. On the other hand, ILP Formulation 2 quickly becomes huge. In a way, ILP Formulation 2 uses the whole search space of ILP Formulation 1 as its input data. Hence, the complexity of the model is passed from the search space to the size of the model. Kumm reported the size of $\mathcal{A}^s$ for a few s and increasing wordlengths [3, Section 5.5]. For $s = 4$, $\mathcal{A}$-sets become quickly too big and long to be computed and, for wordlengths greater than 22, $\mathcal{A}^3$ already has more than a million elements. This rapidly constrains the adder depth and/or the wordlength of this formulation. However, in most practical uses a relatively small wordlength and an adder depth of three or four is enough or a material constraint.

As mentioned before, this formulation is a shortest-path in a particular finite and relatively small hypergraph while ILP Formulation 1 has a search space that is a much bigger hypergraph and which is not precomputed. Thus, for most instances, ILP Formulation 2 is supposed to perform better and we confirm this with experimental results.

3.1.2. Fine-tuning of big M *constraints*

Given a big M constraint, the better big M is the smallest one. The purpose of a big M is to inactivate a constraint, if it is too small it might not inactivate the constraint and rather interfere with values that some variables could have taken. Let $a_i \in \left[\underline{a}, \overline{a}\right]$, $b_i \in \left[\underline{b}, \overline{b}\right]$ and $c_i \in \{0, 1\}$ be three sets of variables. Roughly, for a constraint $a_i \leq b_i + c_i M$, we have,

$$M = \overline{a} - \underline{b}. \tag{41}$$

We will see that, in ILP Formulation 1, a few big M's need to take values up to 2^{2d}. This has a direct impact on for which wordlengths numerical instability could occur. By fine-tuning the maximum value of big M's by modifying and adding constraints we show that it allows a wider range of possible wordlengths to be used. For this analysis we will restrain shifts to positives only, $s \geq 0$. This analysis naturally extends for negative shifts. First, it is straightforward that $M = 2^d$ is big enough for constraints (C1.3a), (C1.3b), (C1.9a), (C1.9b), (C1.10a), (C1.10b), (C1.12a) and (C1.12b). However, for constraints (C1.5a) and (C1.5b) different M's are needed. For the latter $M = 2^d - 2^s$ is big enough. *Per contra*, for (C1.5a) $M = 2^{d+s}$ is needed. Thus, for $s = d$, M have to take values up to 2^{2d}. Default integrality tolerance is 10^{-5} and can be safely adjusted down to 10^{-9} for CPLEX and Gurobi. This parameter will be denoted *IntTol* and used to deduce which maximum value the wordlength can take depending on this tolerance. Hence, for:

$$\phi_{a,i,s} \in [1 - \mathit{IntTol}, 1 + \mathit{IntTol}], \tag{42}$$

we have:

$$c_{a,i}^{\text{sh}} \in \left[2^s c_{a,i} - \mathit{IntTol} \times 2^{2d}, 2^s c_{a,i} + \mathit{IntTol} \times 2^{2d}\right]. \tag{43}$$

It is clear that a problem can occur only if $\mathit{IntTol} \times 2^{2d} \geq 1$. Thus, it is imperative that $\mathit{IntTol}^{-1} > 2^{2d}$. Hence, it is direct that $d = 8$ and $d = 14$ are the longest possible wordlengths for, respectively, default integrality tolerance, 10^{-5}, and smallest integrality tolerance, 10^{-9}. Those wordlengths are smalls and it would be interesting to modify big M constraints in order to increase possible wordlength values.

Switching between indicators and big M constraints in a same model when M becomes too big could be a solution to avoid numerical instabilities. Nevertheless, a fully linear programming model can be obtained with lower big M. To this end, new sets of variables are needed:

- $c_{a,i,s} \in [\![0, 2^d]\!]$, $a \in [\![1, N_A]\!]$, $i \in \{l, r\}$, $s \in [\![0, S_{\max}]\!]$: 0 if $2^s c_a > 2^d$, constant c_a shifted by s otherwise;
- $\psi_{a,i,s} \in \{0, 1\}$, $a \in [\![1, N_A]\!]$, $i \in \{l, r\}$, $s \in [\![1, S_{\max}]\!]$: 1 if $2^s c_a > 2^d$.

Constraints (C1.5a) and (C1.5b) can be replaced by

$$c_{a,i}^{\text{sh}} \geq c_{a,i,s} + (\phi_{a,i,s} - 1) \times M \text{ and } c_{a,i}^{\text{sh}} \leq c_{a,i,s} + (1 - \phi_{a,i,s}) \times M. \tag{44}$$

In those constraints, $M = 2^d$ is enough. To link $c_{a,i,s}$ and $\psi_{a,i,s}$ with the whole model, a few constraints are added:

$$2 \times c_{a,i,s-1} \leq 2^d + \psi_{a,i,s} \times 2^d \qquad \forall a \in [\![1, N_A]\!], i \in \{l, r\}, s \in [\![1, S_{\max}]\!] \tag{C1.5-1}$$

$$c_{a,i,0} = c_{a,i} \qquad \forall a \in [\![1, N_A]\!], i \in \{l, r\} \tag{C1.5-2}$$

$$c_{a,i,s} \leq 2 \times c_{a,i,s-1} \qquad \forall a \in [\![1, N_A]\!], i \in \{l, r\}, s \in [\![1, S_{\max}]\!] \tag{C1.5-3}$$

$$c_{a,i,s} \geq 2 \times c_{a,i,s-1} - \psi_{a,i,s} \times 2^d \qquad \forall a \in [\![1, N_A]\!], i \in \{l, r\}, s \in [\![1, S_{\max}]\!] \tag{C1.5-4}$$

$$c_{a,i,s} \leq (1 - \psi_{a,i,s}) \times 2^d \qquad \forall a \in [\![1, N_A]\!], i \in \{l, r\}, s \in [\![1, S_{\max}]\!] \tag{C1.5-5}$$

Constraint (C1.5-1) forces $\psi_{a,i,s} = 1$ if $2^s c_{a,i} > 2^d$. Constraint (C1.5-5) ensures that $c_{a,i,s} = 0$ if $\psi_{a,i,s} = 1$. Constraints (C1.5-2), (C1.5-3) and (C1.5-4) imply that $c_{a,i,s} = 2^s c_{a,i}$ if $\psi_{a,i,s} = 0$. Note that the variable $c_{a,i,s}$ could be equal to 0 even if $2^s c_{a,i} \leq 2^d$, yet this has no impact on the solution. This modification has a positive impact on possible wordlengths. Maximum wordlength becomes 16 for default integrality tolerance and 29 for an integrality tolerance of 10^{-9}. The greater the integrality tolerance is, the faster is the solver. However, this modelization takes more variables and constraints, that might offset the gain. Hence, we use this modelization only when the wordlength is too big for the initial model.

3.1.3. Objective functions

As stated before, ILP Formulation 1 has no objective function. In order to minimize the number of adders, ILP Formulation 1 is solved multiple times incrementally increasing N_A until a solution is found. This implies to try to solve multiple infeasible linear models, consequently to prove that for multiple values of N_A the model is infeasible. This problem could be solved the other way around: use a greedy algorithm or an heuristic to find a feasible N_A and solving the model by decrementing NA until a model is infeasible. Fixing N_A to its upper bound and minimizing the number of *used adders* is a third possible approach. To do so, for each adder, a binary variable, that is activated when its adder is used, can be added. Then the objective function is to minimize the sum of those binary variables. Formally, that means adding $u_a \in \{0, 1\}$, $\forall a \in [\![1, N_A]\!]$, to the model and the following set of constraints,

$$c_a = 1 \text{ if } u_a = 0 \qquad \forall a \in [\![1, N_A]\!], \tag{45}$$

or, using big M,

$$c_a \leq u_a \times M + 1 \qquad \forall a \in [\![1, N_A]\!], \tag{46}$$

where $M = 2^d$ can be used. This constraint ensures that c_a brings no added value to the solution if $u_a = 0$, it could be replaced by c_0. Yet, this can be enhanced with a few constraints to break symmetries:

$$u_a \leq u_{a-1} \qquad \forall a \in [\![2, N_A]\!] \tag{47}$$

$$\sum_{k=1}^{a-1} c_{a,i,k} \leq 2u_a \qquad \forall a \in [\![1, N_A]\!], i \in \{l, r\} \tag{48}$$

The first constraint forces adder $a-1$ to be used if adder a is. Thus, used adders are first indices. The second constraint implies that inputs of an unused adder are from the initial input. This modification of the model

Table 4: Constants (top) and variables (bottom) for the first NLP model

Constants/Variables
$N_A \in \mathbb{N}$: number of adders;
$N_O \in \mathbb{N}$: number of outputs;
$C \in \mathbb{N}^{N_O}$: target constants;
$S_{\min}, S_{\max} \in \mathbb{Z}$: minimum and maximum shift;
$d \in \mathbb{N}$: wordlength.
$c_a \in [\![1, 2^d]\!]$, $\forall a \in [\![0, N_A]\!]$: constant of adder a;
$\sigma_a \in \{0, 1\}$, $\forall a \in [\![1, N_A]\!]$: equivalent to s in (23), 0 if the adder a corresponds to an addition, 1 if it corresponds to a subtraction;
$c_{a,i,k} \in \{0, 1\}$, $\forall a \in [\![1, N_A]\!]$, $i \in \{l, r\}$, $k \in [\![0, N_A - 1]\!]$: 1 if input i of adder a is adder k, 0 otherwise;
$s_a^+ \in [\![0, S_{\max}]\!]$, $\forall a \in [\![1, N_A]\!]$: shift of left input of adder a;
$s_a^- \in [\![S_{\min}, 0]\!]$, $\forall a \in [\![1, N_A]\!]$: negative shift of adder a;
$o_{a,j} \in \{0, 1\}$, $\forall a \in [\![1, N_A]\!]$, $j \in [\![1, N_O]\!]$: 1 if adder a is equal to the j-th target constant, 0 otherwise.

permits to give to the solver a first solution using heuristics or bounds found with a greedy algorithm. This can speed up the solver allowing it to use its full potential and not only its satisfiability part. However, this modeling requires more variables and its linear relaxation might be far from a known lower bound $\underline{N}_A$. To help the solver, u_a should be fixed to 1 for all $a \leq \underline{N}_A$. MCM problem with this objective will be referred to as MCM_{OBJ}.

As already presented, a second objective which might be interesting to minimize is the adder depth. For each adder, a new integer variable, ad_a, representing the adder depth can be added to the model. Then we just introduce an integer variable max_ad, the constraints $max\_ad \geq ad_a$, $\forall a \in [\![1, N_A]\!]$, and we minimize max_ad. In order to represent the adder depth, first $ad_0 = 0$ is fixed. Then a set of constraints is added:

$$ad_a \geq ad_k + 1 - (1 - c_{a,i,k}) \times N_A, \qquad \forall a \in [\![1, N_A]\!], i \in \{l, r\}, k \in [\![0, a - 1]\!]. \tag{49}$$

Despite that N_A can be seen as a big M here, N_A remains relatively small and it would be ineffective to write this constraint using an implication. Note that for a given a, ad_a might be greater than the actual adder depth of its associated adder, however it will be low enough so that the graph adder depth is actually minimized and its real adder depth could be computed from the solution. Adding this objective function permits to tackle MCM_{MAD}. In order to handle MCM_{BAD} it is enough to add a unique constraint, $max\_ad \leq AD$, where AD is the wanted bound.

Combining both previous objectives is possible with a priority on minimizing the number of adders. It is straightforward that minimizing $N_A \times \sum u_a + max\_ad$ tackles this aim which allows handling the whole problem solving a single model. This corresponds to a weighted sum that ensures the first objective, which is to minimize the number of used adders, is optimized before the second objective, which is to minimize the adder depth.

3.2. Nonlinear model

No analysis of NLP models are given in the literature, however this seems to be a reasonable approach because of the nature of $\mathcal{A}$-sets. We propose two models to evaluate the potential of NLP models. The first one checks for satisfiability while the second minimizes the number of used adders. Constants and variables that are used in the first model are presented Table 4.

Constraints for first NLP model:

$$c_0 = 1 \tag{C3.1}$$

$$2^{s_a^-} c_a = \left| 2^{s_a^+} \sum_{k=0}^{a-1} (c_k c_{a,l,k}) + (1 - 2\sigma_a) \sum_{k=0}^{a-1} (c_k c_{a,r,k}) \right| \qquad \forall a \in [\![1, N_A]\!] \tag{C3.2}$$

$$\sum_{k=0}^{a-1} c_{a,i,k} = 1 \qquad \forall a \in [\![1, N_A]\!], i \in \{l, r\} \tag{C3.3}$$

$$s_a^- s_a^+ = 0 \qquad \forall a \in [\![1, N_A]\!] \tag{C3.4}$$

$$s_a^+ - s_a^- \geq 1 \qquad \forall a \in [\![1, N_A]\!] \tag{C3.5}$$

$$\sum_{a=1}^{N_A} c_a o_{a,j} = C_j \qquad \forall j \in [\![1, N_O]\!] \tag{C3.6}$$

$$\sum_{a=1}^{N_A} o_{a,j} = 1 \qquad \forall j \in [\![1, N_O]\!] \tag{C3.7}$$

Constraints (C3.2) is equivalent to the $\mathcal{A}$-operation (23) where a shift has been removed, where for a unique k, $u = c_k c_{a,l,k}$ and for a unique k', $v = c_{k'} c_{a,r,k'}$. For a given pair of (a, i), only one k permits $c_k c_{a,i,k} \neq 0$ because of constraint (C3.3). Furthermore, $(-1)^s$ has been replaced with $1 - 2s$ because we think, *a priori*, that NLP solvers might handle the latter better. Constraint (C3.4) and (C3.5) ensure that either the input shift or the adder shift is equal to zero and one is not zero. Last constraints, (C3.6) and (C3.7), permit to associate adder values to output values and ensure that each output as an associated adder.

This first decision NLP model can be extended to obtain a minimization model using the same method as for the ILP approach. To do so, we start by adding variables $u_a \in \{0, 1\}$, $\forall a \in [\![1, N_A]\!]$ and an objective to minimize: $\min \sum_{a=1}^{N_A} u_a$. Then we add a few constraints that ensure that an unused adder cannot be used as an input for another adder nor associated to an output. Furthermore, the value of an unused adder is fixed to one and adders are sorted: unused adders at the tail. Formally, this is done with four constraints:

$$(1 - u_a) \times (c_a - 1) = 0 \qquad \forall a \in [\![1, N_A]\!] \tag{50}$$

$$u_a \leq u_{a-1} \qquad \forall a \in [\![2, N_A]\!] \tag{51}$$

$$\sum_{j=1}^{N_O} o_{a,j} \leq u_a \qquad \forall a \in [\![1, N_A]\!] \tag{52}$$

$$c_{a,i,k} \leq u_k \qquad \forall a \in [\![1, N_A]\!], k \in [\![1, a-1]\!], i \in \{l, r\} \tag{53}$$

This gives two NLP models to compare with. Although results are not expected to be promising, MCM problem can be combined with filter design which naturally comes with nonlinear constraints and having a NLP formulation for MCM can become convenient.

3.3. Experimental results

In this section we will compare solving times for different models. Two modeling languages were considered. First, ScaLP [8] is the modeling language used by Kumm and it was natural to use the same since many models he proposed are open source. However this language does not handle yet nonlinear terms. This limitation was a real issue since part of our objective was to compare the efficiency of nonlinear models with linear models. The other modeling language considered, which is the one we used, is JuMP [7].

Table 5: Instances used for the experiments

Name	wordlength	Unique odd coefficients
gaussian 3x3 8bit	8	3, 21, 159
gaussian 5x5 12bit	12	1, 23, 343, 1267
highpass 5x5 8bit	8	1, 3, 5, 7, 121
highpass 9x9 10bit	10	1, 3, 5, 7, 11, 125
highpass 15x15 12bit	12	1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 507
laplacian 3x3 8bit	8	5, 21, 107
lowpass 5x5 8bit	8	11, 33, 35, 53, 103
lowpass 9x9 10bit	10	1, 5, 7, 25, 31, 63, 65, 67, 73, 97, 117, 165, 303
lowpass 15x15 12bit	12	1, 5, 7, 13, 17, 19, 21, 27, 41, 43, 45, 53, 61, 79, 93, 101, 103, 113, 133, 137, 199, 331, 333, 613, 1097, 1197
unsharp 3x3 8bit	8	3, 11, 63
unsharp 3x3 12bit	12	43, 171, 1109

Table 6: Runtime comparison in seconds of ILP Formulation 2 (ILP2) and different formulations derived from ILP Formulation 1 (ILP1). Odd corresponds to ILP Formulation 1odd, MAD is the minimization of adder depth modification, BAD_3 the bounded adder depth modification for a bound of three and MIN is the minimization problem. TO is for Time Out (3600s) and OOM stands for Out Of Memory. Experiments were made using CPLEX (top) and Gurobi (bottom).

Names		Using indicator constraints					Using big M constraints				
	ILP2 [3]	ILP1 [2]	Odd	MAD	BAD_3	MIN	ILP1 [2]	Odd	MAD	BAD_3	MIN
gaussian 3x3 8bit	< 1	< 1	< 1	< 1	< 1	< 1	< 1	< 1	< 1	< 1	< 1
gaussian 5x5 12bit	*TO*	< 1	18	431	1285	*TO*	9	84	88	484	1745
highpass 5x5 8bit	< 1	< 1	< 1	< 1	< 1	< 1	< 1	< 1	< 1	< 1	< 1
highpass 9x9 10bit	< 1	< 1	< 1	< 1	< 1	< 1	< 1	< 1	< 1	< 1	< 1
highpass 15x15 12bit	< 1	53	*TO*	1251	13	47	154	< 1	771	6	123
laplacian 3x3 8bit	< 1	< 1	< 1	< 1	< 1	< 1	< 1	< 1	< 1	< 1	< 1
lowpass 5x5 8bit	< 1	13	7	3	12	*TO*	2	4	14	2	51
lowpass 9x9 10bit	< 1	*TO*	*TO*	*TO*	*TO*	*TO*	40	18	*OOM*	473	*OOM*
lowpass 15x15 12bit	*TO*	*TO*	*TO*	*TO*	*TO*	*TO*	*TO*	*TO*	*OOM*	*TO*	*OOM*
unsharp 3x3 8bit	< 1	< 1	< 1	< 1	< 1	< 1	< 1	< 1	< 1	< 1	< 1
unsharp 3x3 12bit	1301	10	5	60	21	*TO*	2	2	9	3	145
gaussian 3x3 8bit	< 1	< 1	< 1	< 1	< 1	< 1	< 1	< 1	< 1	< 1	< 1
gaussian 5x5 12bit	1758	244	54	756	2706	*TO*	4	2	368	701	44
highpass 5x5 8bit	< 1	< 1	< 1	< 1	< 1	< 1	< 1	< 1	< 1	< 1	< 1
highpass 9x9 10bit	< 1	< 1	< 1	< 1	< 1	26	< 1	< 1	< 1	< 1	< 1
highpass 15x15 12bit	< 1	< 1	*TO*	*TO*	*TO*	*TO*	< 1	< 1	53	2	2
laplacian 3x3 8bit	< 1	< 1	< 1	< 1	< 1	< 1	< 1	< 1	< 1	< 1	< 1
lowpass 5x5 8bit	< 1	6	11	156	70	7	2	2	85	2	4
lowpass 9x9 10bit	< 1	*TO*	*TO*	*TO*	*TO*	*TO*	19	74	*TO*	1169	*TO*
lowpass 15x15 12bit	*TO*	*TO*	*TO*	*TO*	*TO*	*TO*	*TO*	*TO*	*TO*	*TO*	*TO*
unsharp 3x3 8bit	< 1	< 1	< 1	< 1	< 1	< 1	< 1	< 1	< 1	< 1	< 1
unsharp 3x3 12bit	849	56	157	345	105	*TO*	3	< 1	21	2	6

Table 7: Runtime comparison in seconds of ILP Formulation 2 (ILP2), ILP Formulation 1 (ILP1), 1odd (Odd) with big M constraints and the NLP models, NLP corresponds to the decision one when NLP_Min corresponds to the minimization form. TO is for Time Out (3600s). Experiments were made using SCIP.

	ILP2 [3]	ILP1 [2]	Odd	NLP	NLP_Min
gaussian 3x3 8bit	< 1	< 1	< 1	3	2
gaussian 5x5 12bit	TO	299	1042	TO	TO
highpass 5x5 8bit	< 1	< 1	< 1	< 1	2
highpass 9x9 10bit	< 1	< 1	< 1	< 1	155
highpass 15x15 12bit	< 1	170	TO	TO	TO
laplacian 3x3 8bit	< 1	< 1	< 1	< 1	2
lowpass 5x5 8bit	< 1	4	14	151	TO
lowpass 9x9 10bit	< 1	2486	TO	TO	TO
lowpass 15x15 12bit	TO	TO	TO	TO	TO
unsharp 3x3 8bit	< 1	< 1	< 1	2	3
unsharp 3x3 12bit	TO	179	38	235	TO

He has both linear and nonlinear modeling possibilities and can be plugged-in with Gurobi [5], CPLEX [4] and SCIP [6] which are solvers used in previous works.

Several models are tested, both with indicator constraints and big M constraints. When big M constraints were used, integrality tolerance has been adjusted to avoid numerical instability. Models are implemented in Julia using JuMP and solved with a time limit of one hour, 3600 seconds. CPLEX 12.9 and Gurobi 9 were used for ILP formulations and SCIP 6.0.2 for the NLP models. The computer on which the experimentation was done is on Linux 64bit (Ubuntu 18.04.4 LTS), it has 16Go of DDR3 RAM and an Intel® Core™ i5-4570S, quadcore (2.90GHz). All the experiments were performed on instances from Kumm's thesis, the detail of those instances is given Table 5. No assumptions on possible shifts were made, hence the shift's range is from minus the wordlength to the wordlength.

Table 6 gives the results in seconds for ILP Formulation 1 and its variants, ILP Formulation 2 results corresponds to the ones reported by Kumm where runtimes greater than our time limit are considered timed out [3, Table 5.4]. Some instances using big M constraints raised an Out Of Memory error with CPLEX. Since models using indicator constraints and big M constraints are of the same order of magnitude this surely comes from the way the solver optimizes its branch and bound. Models with indicator constraints timed-out more often than models with big M constraints and often took more time to be solved. Globally, models with an objective function took more time than decision models. Despite its theoretical interest, the MIN model does not improve the solving and can probably be left aside in future work, unless the use of better upper bounds on the number of adders drastically increases its efficiency. MAD model, on the other hand, has showed to be efficient enough to be used when minimizing the adder depth is important. In that case, the use of big M constraints permitted faster solving time with the drawback of great memory usage. Note that in both models using an objective function, an optimal solution could have been found fast and most of the time could have been use to confirm its optimality. We left that verification aside as it is a whole experiment in itself and it is not of immediate interest for us. BAD_3 model worked just fine compared to ILP Formulation 1 and other variants. This variant is essential since it can be fairly compared to ILP Formulation 2 and reassures that the precomputation process is of great interest in solving MCM_{BAD} problem. As expected ILP Formulation 2 seems to behave at least as good as ILP Formulation 1. Surprisingly, the gaussian 5x5 12bit and the unsharp 3x3 12bit instances have been handled better by ILP Formulation 1 BAD_3. ILP Formulation 1odd performed as good as ILP Formulation 1 and other variants.

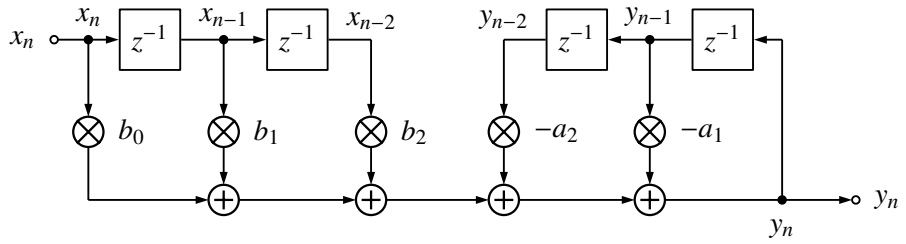


Figure 7: Direct form realization of a second-order IIR filter.

Except for the instance lowpass 15x15 12bit, which was left unsolved by almost every model and solver, ILP Formulation 1odd solved every instance in a reasonable time.

Both NLP models obtained time results are reported Table 7. ILP Formulation 2 corresponds to results reported by Kumm where runtimes greater than our time limit are considered timed out [3, Table 5.4]. On a few instances that most models handled well, NLP models shown to work too. However, NLP models timed out on instances that were solved relatively fast by ILP formulations. We conclude that ILP formulations should probably be preferred over NLP formulations. No clear result emerge from the comparison between variants and indicator/big M constraints. An experimentation on more instances and an analysis of these instances using Data Science methods seem useful to help understanding when and why a variant prevails on another. When a formulation for MCM problem will be needed in the following, we will prefer ILP Formulation 1odd with big M constraints as it seems more robust: It gave the best results on most of the instances and not too far from the best on the others, regardless of the solver.

4. Design of second order IIR filters

> Study hard what interests you the most in the most undisciplined, irreverent and original manner possible.
>
> Richard Feynman

We extended the design process of FIR filters to second order IIR filters. Those filters have a strongly different behavior and all the coefficients cannot be included into a single MCM problem. Indeed, in Figure 7 it can be seen that a's and b's have different inputs. Second order IIR filters are an essential block of IIR filter design since many methods rely on second order sections (SOS) to design greater order filters [24, 25]. The transfer function of second order filters is given below:

$$H(z) = \frac{b(z)}{a(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} \qquad \text{with } a_1, a_2, b_0, b_1, b_2 \in \mathbb{R}, z \in \mathbb{C}. \tag{54}$$

In order to have the desirable design in the meaningful frequency domain, $|H(z)|$ has to be constrained on the unit circle, *i. e.* $z = e^{i\omega}$:

$$\underline{D}(\omega) \leq \left|H\left(e^{i\omega}\right)\right| \leq \overline{D}(\omega)\,, \qquad \forall \omega \in \Omega. \tag{55}$$

We obtained an explicit expression with respect to the filter parameters by expanding $\left|H\left(e^{i\omega}\right)\right|^2$:

$$\left|H\left(e^{i\omega}\right)\right|^2 = \frac{{b_0}^2 + {b_1}^2 + {b_2}^2 + 2b_0b_1\cos(\omega) + 2b_0b_2\cos(2\omega) + 2b_1b_2\cos(\omega)}{1 + {a_1}^2 + {a_2}^2 + 2a_1\cos(\omega) + 2a_2\cos(2\omega) + 2a_1a_2\cos(\omega)}. \tag{56}$$

Details are given in Appendix A. This expansion involves quadratic terms that do not fit for an ILP model. Furthermore, unlike FIR filters, IIR filters have stability issues. Those issues are addressed in the next sections and solved. It should be noted that,

$$\left|b\left(e^{i\omega}\right)\right|^2 = {b_0}^2 + {b_1}^2 + {b_2}^2 + 2b_0b_1\cos(\omega) + 2b_0b_2\cos(2\omega) + 2b_1b_2\cos(\omega)\,, \tag{57}$$

has two symmetries: The values of the variables b_0 and b_2 could be exchanged and the values (b_0, b_1, b_2) could be sign reversed to $(-b_0, -b_1, -b_2)$.

4.1. Linearize quadratic terms

In order to avoid quadratic terms as $z = xy$, with $x, y, z \in \mathbb{N}$, a method was presented in 2008 by Billionnet *et al.* [26]. Let $d \in \mathbb{N}$ be a positive integer, under the assumption of $0 \leq x \leq 2^d$ and $0 \leq y \leq 2^d$, it suffices to introduce binary variables $t_{y,i}$ for $i \in [\![0, d]\!]$ and the constraint $y = \sum_{i=0}^{d} 2^i t_{y,i}$. Then xy can be rewrite as d multiplications between a binary variable and an integer variable which can be linearized easily. Hence $z = xy$ will become the following set of constraints:

$$y = \sum_{i=0}^{d} 2^i t_{y,i} \tag{58}$$

$$u_{x,y,i} \leq M \times t_{y,i} \qquad \forall i \in [\![0, d]\!] \tag{59}$$

$$u_{x,y,i} \leq x \qquad \forall i \in [\![0, d]\!] \tag{60}$$

$$u_{x,y,i} \geq x - M\left(1 - t_{y,i}\right) \qquad \forall i \in [\![0, d]\!] \tag{61}$$

$$z = \sum_{i=0}^{d} 2^i u_{x,y,i}, \tag{62}$$

with $u_{x,y,i} \in \mathbb{N}$ for $i \in [\![0, d]\!]$ and with M equals to the upper bound of x. In Section 4.3 we will show how to get bounds a_1, a_2, b_0, b_1, b_2, therefore $M = 2^d$ meets the objective after normalization (10) and this ensures that the assumptions $x \leq 2^d$ and $y \leq 2^d$ are verified in our design method. However, there is no certitude on the sign of a_1, a_2, b_0, b_1, b_2, thus a sign variable is added for each coefficient: $a_i^{\text{sg}} \in \{0, 1\}$ and $b_i^{\text{sg}} \in \{0, 1\}$. Given a variable x, the constraints $x \geq \underline{x}(1 - x^{\text{sg}})$ and $x \leq \overline{x}x^{\text{sg}}$ ensure that $x^{\text{sg}} = 0$ if $x < 0$ and $x^{\text{sg}} = 1$ if $x > 0$. Then variables $a_i^+ = |a_i|$ and $b_i^+ = |b_i|$ can be introduced using a_i^{sg} and b_i^{sg}. Consequently, quadratic products are linearized using positives a_i^+ and b_i^+, then the sign is adjusted with respect to a_i^{sg} and b_i^{sg}. This means replacing x and y with, respectively, x^+ and y^+ in previous constraints and replacing (62) with the following set of inequalities where x^{sg} and y^{sg} are binary variables equal to 1 if $x > 0$ and $y > 0$ respectively,

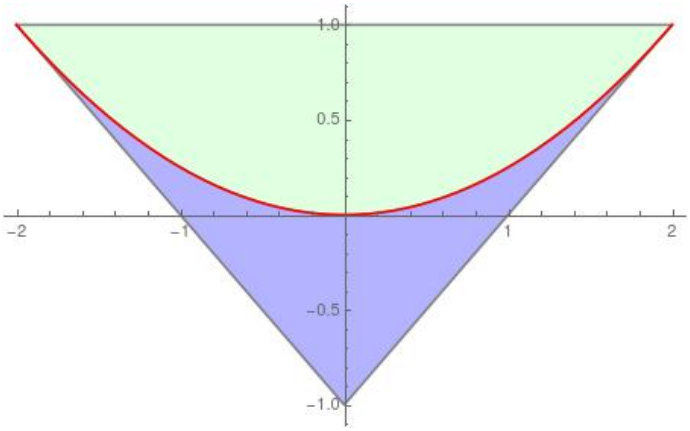


Figure 8: a's that ensure stability. a_1 is the abscissa and a_2 the ordinate. In blue (bottom), poles are both real valued, in green (top), poles are complex conjugates, in red a single double pole.

and equal to 0 if $x < 0$ and $y < 0$ respectively:

$$z \geq \sum_{i=0}^{d} 2^i u_{x,y,i} - 2 \times M\,(x^{\text{sg}} + y^{\text{sg}}) \tag{63}$$

$$z \geq \sum_{i=0}^{d} 2^i u_{x,y,i} - 2 \times M\,(2 - x^{\text{sg}} - y^{\text{sg}}) \tag{64}$$

$$z \leq \sum_{i=0}^{d} 2^i u_{x,y,i} \tag{65}$$

$$z \leq -\sum_{i=0}^{d} 2^i u_{x,y,i} + 2 \times M\,(1 + x^{\text{sg}} - y^{\text{sg}}) \tag{66}$$

$$z \leq -\sum_{i=0}^{d} 2^i u_{x,y,i} + 2 \times M\,(1 - x^{\text{sg}} + y^{\text{sg}}) \tag{67}$$

$$z \geq -\sum_{i=0}^{d} 2^i u_{x,y,i} \tag{68}$$

Here, M should be replaced by the upper bound of x^+ times the upper bound of y^+. $M = 2^{2d}$ is good enough.

4.2. Stability

Stability is an issue for IIR filters. An IIR filter is stable if and only if its poles are located inside the unit circle. For a second order IIR filter there is two poles to consider, either both real, complex conjugate or one double root. To do so, three cases are considered: $\Delta < 0$, $\Delta > 0$ and $\Delta = 0$, with $\Delta = {a_1}^2 - 4a_2$. From those three cases, we deduced constraints on a_1 and a_2 that ensure stability.

Theorem 1 (Stability)**.** *Given F a second order IIR filter, F is stable if, and only if, $a_1 \in \,]-2, 2[$ and $a_2 \in \,]|a_1| - 1, 1[$.*

Proof. We recall that stability (9) is equivalent to, for all λ, poles of the filter, $|\lambda| < 1$. For second order filters poles are the roots of the quadratic equation:

$$A(z) = z^2 + a_1 z + a_2. \tag{69}$$

We have three cases to consider that depend on the sign of Δ with:

$$\Delta = {a_1}^2 - 4a_2. \tag{70}$$

First, we suppose that $\Delta < 0$, thus that the roots of the polynomial are complex conjugates. The roots, z_1 and z_2, of the polynomial A are equal to:

$$z_1 = \overline{z_2} = \frac{1}{2}\left(-a_1 + i\sqrt{-\Delta}\right). \tag{71}$$

We make the hypothesis:

$$\begin{cases} \Delta < 0 \\ |z_1| < 1 \wedge |z_2| < 1 \end{cases}, \tag{72}$$

which is equivalent to:

$$\begin{cases} \frac{{a_1}^2}{4} < a_2 \\ \sqrt{{a_1}^2 + \sqrt{-{a_1}^2 + 4{a_2}^2}^2} < 2 \Leftrightarrow 0 < {a_1}^2 + -{a_1}^2 + 4a_2 < 4 \Leftrightarrow 0 < a_2 < 1 \end{cases}. \tag{73}$$

We have obtained an equivalent and simpler set of constraints to (72): $a_2 \in \left]\frac{{a_1}^2}{4}, 1\right[$. In order to guaranty the existence of a valid value for a_2 we obtain bounds on a_1 too: $a_1 \in]-2, 2[$.

Second, we suppose that $\Delta > 0$, thus that the roots of the polynomial are real valued. The roots, z_1 and z_2, of the polynomial A are equal to:

$$z_1 = \frac{1}{2}\left(-a_1 + \sqrt{\Delta}\right) \text{ and } z_2 = \frac{1}{2}\left(-a_1 - \sqrt{\Delta}\right). \tag{74}$$

We make the hypothesis:

$$\begin{cases} \Delta < 0 \\ |z_1| < 1 \wedge |z_2| < 1 \end{cases}. \tag{75}$$

By noting that for $u \in \mathbb{R}$ and $v, w \in \mathbb{R}_+$, $|u \pm v| < w$ is equivalent to $|u| + v < w$ we deduce that:

$$|z_1| < 1 \wedge |z_2| < 1 \Leftrightarrow \frac{1}{2}\left(|a_1| + \sqrt{\Delta}\right) < 1. \tag{76}$$

Then, System (75) is equivalent to:

$$\begin{cases} a_2 < \frac{{a_1}^2}{4} \\ \sqrt{{a_1}^2 - 4a_2} < 2 - |a_1| \Leftrightarrow 0 < {a_1}^2 - 4a_2 < {a_1}^2 - 4\,|a_1| + 4 \Leftrightarrow |a_1| - 1 < a_2 < \frac{{a_1}^2}{4} \end{cases}. \tag{77}$$

We have obtained an equivalent and simpler set of constraints to (75): $a_2 \in \left]|a_1| - 1, \frac{{a_1}^2}{4}\right[$. Furthermore, in system (77) we had $0 < \sqrt{\Delta} < 2 - |a_1|$, hence $a_1 \in]-2, 2[$.

Third case, we suppose $\Delta = 0$, thus that we have a single root, z_1, equal to:

$$z_1 = -\frac{1}{2}a_1. \tag{78}$$

Constraining its absolute value leads to:

$$\begin{cases} \Delta = 0 \\ |z_1| < 1 \end{cases} \Leftrightarrow \begin{cases} {a_1}^2 - 4a_2 = 0 \\ |a_1| < 2 \end{cases} \Leftrightarrow \begin{cases} a_1 \in]-2, 2[\\ a_2 = \frac{{a_1}^2}{4} \end{cases}. \tag{79}$$

Finally, we reunify the three cases into one set of constraints:

$$\text{Second-order filter stability} \Leftrightarrow \left\{ \begin{array}{l} \left\langle \begin{array}{l} a_2 \in \left]|a_1| - 1, \frac{{a_1}^2}{4}\right[\\ a_2 = \frac{{a_1}^2}{4} \\ a_2 \in \left]\frac{{a_1}^2}{4}, 1\right[\end{array} \right\rangle \\ a_1 \in]-2, 2[\end{array} \right\} \Leftrightarrow a_2 \in]|a_1| - 1, 1[\quad . \tag{80}$$

□

Possible values for a_1 and a_2 are represented Figure 8 depending on poles. Theorem 1 gives bounds on a's and a single constraint, $a_2 > |a_1| - 1$, that ensure stability for a second order IIR filter. In fixed-point arithmetic the absolute value can be linearize without much effort, actually to linearize the bilinear terms, $|a_1|$ must be computed anyway.

4.3. Bounds on coefficients

As explained in Section 4.1, bounds on coefficients a's and b's are necessary for the linearization part. From Theorem 1, bounds on a_1 and a_2 are directly deduced: $a_1 \in]-2, 2[$ and $a_2 \in]-1, 1[$. Bounds on b's are left to be founded. Coefficients a's and b's satisfy the following constraint:

$$\underline{D}(\omega)^2 \leq \left|H\left(e^{i\omega}\right)\right|^2 = \frac{\left|b\left(e^{i\omega}\right)\right|^2}{\left|a\left(e^{i\omega}\right)\right|^2} \leq \overline{D}(\omega)^2\,, \qquad \forall \omega \in \Omega. \tag{81}$$

This links $\left|b\left(e^{i\omega}\right)\right|^2$ and $\left|a\left(e^{i\omega}\right)\right|^2$ together, where,

$$\left|b\left(e^{i\omega}\right)\right|^2 = {b_0}^2 + {b_1}^2 + {b_2}^2 + 2b_0b_1\cos(\omega) + 2b_0b_2\cos(2\omega) + 2b_1b_2\cos(\omega)\,, \qquad \forall \omega \in \Omega, \tag{82}$$

$$\left|a\left(e^{i\omega}\right)\right|^2 = 1 + {a_1}^2 + {a_2}^2 + 2a_1\cos(\omega) + 2a_2\cos(2\omega) + 2a_1a_2\cos(\omega)\,, \qquad \forall \omega \in \Omega. \tag{83}$$

We use the bounds on a's to find bounds on $\left|a\left(e^{i\omega}\right)\right|^2$ using (83) then we get bounds on $\left|b\left(e^{i\omega}\right)\right|^2$ using (81). As a square, 0 is a direct lower bound for $\left|a\left(e^{i\omega}\right)\right|^2$. Furthermore, we found that this bound cannot be increased:

$$\lim_{\substack{a_1 \to 0 \\ a_2 \to 1 \\ \omega \to \frac{\pi}{2}}} \left|a\left(e^{i\omega}\right)\right|^2 = 0. \tag{84}$$

Studying the monotony of terms of $\left|a\left(e^{i\omega}\right)\right|^2$ it is straightforward that $\left|a\left(e^{i\omega}\right)\right|^2 \leq 16$. This bound of 16 cannot be lowered since:

$$\lim_{\substack{a_1\to 2\\ a_2\to 1\\ \omega\to 0}} \left|a\left(e^{i\omega}\right)\right|^2 = 16. \tag{85}$$

From these bounds it can be deduced bounds on $\left|b\left(e^{i\omega}\right)\right|^2$:

$$\underline{D}(\omega)^2 \leq \frac{\left|b\left(e^{i\omega}\right)\right|^2}{\left|a\left(e^{i\omega}\right)\right|^2} \leq \overline{D}(\omega)^2, \qquad \forall\omega\in\Omega, \tag{86}$$

$$\Rightarrow \left|a\left(e^{i\omega}\right)\right|^2 \underline{D}(\omega)^2 \leq \left|b\left(e^{i\omega}\right)\right|^2 \leq \left|a\left(e^{i\omega}\right)\right|^2 \overline{D}(\omega)^2, \qquad \forall\omega\in\Omega, \tag{87}$$

$$\Rightarrow 0 \leq \left|b\left(e^{i\omega}\right)\right|^2 \leq 16\overline{D}(\omega)^2, \qquad \forall\omega\in\Omega. \tag{88}$$

Since $\overline{D}(\omega)$ is not bounded *a priori*, it is not possible to derive bounds on b's independent of filter specifications. However, those bounds are mandatory to linearize quadratic terms or to situate the most significant bit (MSB) for fixed-point arithmetic. Some MILP solvers (as CPLEX or Gurobi) handle quadratic terms in case those are convex problems. Problems,

$$\min / \max b_i \qquad \forall i\in\{0,1,2\} \tag{89}$$

$$\text{s.t.} \quad \left|b\left(e^{i\omega}\right)\right|^2 \leq 16\overline{D}(\omega)^2 \qquad \forall\omega\in\Omega_d \tag{90}$$

$$b_0, b_1, b_2 \in \mathbb{R},$$

are convex, thus we can use CPLEX or Gurobi to find bounds on each b_i before implementing the whole model. The following models have to be solved in order to obtain lower and upper bounds on b_i's denoted $\underline{b}_i$ and $\overline{b}_i$ respectively:

$$\min / \max b_i \qquad \forall i\in\{0,1,2\} \tag{91}$$

$$\text{s.t.} \quad {b_0}^2 + {b_1}^2 + {b_2}^2 + 2b_0b_1\cos(\omega) + 2b_0b_2\cos(2\omega) + 2b_1b_2\cos(\omega) \leq 16\overline{D}(\omega)^2 \qquad \forall\omega\in\Omega_d \tag{92}$$

$$b_0, b_1, b_2 \in \mathbb{R}.$$

Knowing these bounds allows obtaining a linear model for the design of second-order IIR filters: The whole design process is summarized in the next section.

4.4. ILP Model

In fixed-point context, for this first ILP model, it is easier to suppose that variables a's and b's are in $[-1, 1]$ and use the whole interval so that their integer counterpart is in $[\![-2^d, 2^d]\!]$ where d is the wordlength. This was already mentioned in Section 2.1 with (10). Going forward, integer coefficients will be used more, hence real coefficients a's and b's will be denoted a''s and b''s in the following. Bounds, $\underline{a}'_i$, $\overline{a}'_i$, $\underline{b}'_i$, $\overline{b}'_i$, on a''s and b''s can be formally obtained or precomputed. Normalized integer coefficients a's and b's are deduced from a''s, b''s and their bounds:

$$a_i = a'_i \times \frac{2^d}{2^{\left\lceil \log_2 \max\left\{-\underline{a}'_1, -\underline{a}'_2, \overline{a}'_1, \overline{a}'_2\right\}\right\rceil}} \qquad \forall i\in\{1,2\} \tag{93}$$

$$b_i = b'_i \times \frac{2^d}{2^{\left\lceil \log_2 \max\left\{-\underline{b}'_0, -\underline{b}'_1, -\underline{b}'_2, \overline{b}'_0, \overline{b}'_1, \overline{b}'_2\right\}\right\rceil}} \qquad \forall i\in\{0,1,2\} \tag{94}$$

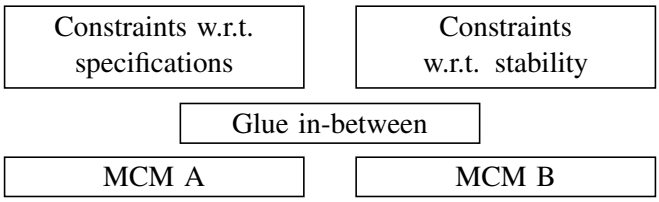


Figure 9: High-level overview of second order IIR filter ILP model

Table 8: Instances used for the IIR experiments

Name	wordlength	passband/π	stopband/π	δ
lowpass 8bit	8	$[0.0, 0.3]$	$[0.7, 1.0]$	0.0636
highpass 8bit	8	$[0.7, 1.0]$	$[0.0, 0.3]$	0.0636
lowpass 10bit	10	$[0.0, 0.5]$	$[0.9, 1.0]$	0.1

This normalization feeds through to $\underline{D}$ and $\overline{D}$, to avoid an overlayer of complexity, just as a's and b's, $\underline{D}$ and $\overline{D}$ are rewritten:

$$\underline{D} \leftarrow \underline{D} \times \frac{2^{\lceil \log_2 \max\{-\underline{a}'_1, -\underline{a}'_2, \overline{a}'_1, \overline{a}'_2\} \rceil}}{2^{\lceil \log_2 \max\left\{-\underline{b}'_0, -\underline{b}'_1, -\underline{b}'_2, \overline{b}'_0, \overline{b}'_1, \overline{b}'_2\right\} \rceil}} \tag{95}$$

$$\overline{D} \leftarrow \overline{D} \times \frac{2^{\lceil \log_2 \max\{-\underline{a}'_1, -\underline{a}'_2, \overline{a}'_1, \overline{a}'_2\} \rceil}}{2^{\lceil \log_2 \max\left\{-\underline{b}'_0, -\underline{b}'_1, -\underline{b}'_2, \overline{b}'_0, \overline{b}'_1, \overline{b}'_2\right\} \rceil}} \tag{96}$$

The model for second order IIR filter is available in Appendix B. First, there is the filter specifications constraints and the stability constraints. Second, constraints for the linearization of quadratic terms containing b's then containing a's are added. The constraints from ILP Formulation 1odd for MCM problems follow. Finally, this model has constraints that link MCM problems, one for a's and one for b's, with filter coefficients. This whole process is presented in a high-level representation Figure 9. Variables and their domain are detailed at the end of the model.

Glue constraints, detailed in Figure B.16 in Appendix B, state that the input is both available in MCM problems for a's and b's coefficients. Then a few constraints ensure that variable $samemcm_{k,k'}$ is equal to 1 if and only if adders k and k' are both available in MCM problem for a's coefficients or both available in MCM problem for b's coefficients. A constraint prevents adder k to be an input of adder a if both adders are not available for the same coefficients (a's or b's). Finally, a few constraints link outputs of MCM with filter coefficients. As already pointed out, the function $\left|b\left(e^{i\omega}\right)\right|^2$ as two symmetries we can break. The constraint $b_0 \geq |b_2|$ allows this and is easy to implement since $|b_2|$ is already computed in the linearization process.

As Volkova *et al.* [1] did for FIR filter coefficients, we first solve the problem without MCM constraints with the objectives min / max a_i and min / max b_i in order to find integer bounds on coefficients. This allows reducing the coefficient range, hence the search space. Furthermore, we integrate in the final solving process an objective function which indicates to maximize the number of coefficients with a value of zero.

4.5. Experimental results

Comparison of our method with existing ones is not possible in a classical way since there are no benchmarks for second order IIR filters. Although second order IIR filters are an essential block in many methods [25, 27, 28], they are part of algorithms that cannot be applied in our single block approach.

Table 9: Runtimes in seconds for the instances used to test our second order IIR filter design method. The first few columns corresponds to the name of the instance, the total time of the solving process (Total), the part of total time used by the precomputation (Pre) and the solving time without any precomputation (NoPre). Total_a and Pre_a correspond to the solving times with precomputation of coefficient a's then b's, Total_b and Pre_b correspond to the opposite. The number of adders used is denoted by NA. Then coefficient values b's and a's are given followed by adder values subscripted with a b or an a to indicate if it is used for the b or the a coefficients respectively. The symmetry breaking constraint was used (top) and inactivated (bottom). Experiments were conducted using CPLEX.

Name	Total_a	Pre_a	Total_b	Pre_b	NoPre	NA	$[b_0, b_1, b_2]$	$[a_1, a_2]$	adder values
lowpass 8bit	15	3	19	7	23	3	$[56, 88, 56]/2^8$	$2 \times [-64, 36]/2^8$	$[9_a, 7_b, 11_b]$
highpass 8bit	9	3	12	6	68	3	$[56, -84, 56]/2^8$	$2 \times [64, 40]/2^8$	$[7_b, 5_a, 21_b]$
lowpass 10bit	31	31	124	124	4	1	$4 \times [128, 144, 32]/2^{10}$	$2 \times [0, 128]/2^{10}$	$[9_b]$
lowpass 8bit	27	9	29	11	157	3	$[-64, -84, -48]/2^8$	$2 \times [-64, 40]/2^8$	$[3_b, 21_b, 5_a]$
highpass 8bit	20	7	27	13	158	3	$[-64, 76, -40]/2^8$	$2 \times [80, 40]/2^8$	$[5_b, 19_b, 5_a]$
lowpass 10bit	728	726	206	204	4	1	$4 \times [32, 144, 128]/2^{10}$	$2 \times [0, 128]/2^{10}$	$[9_b]$

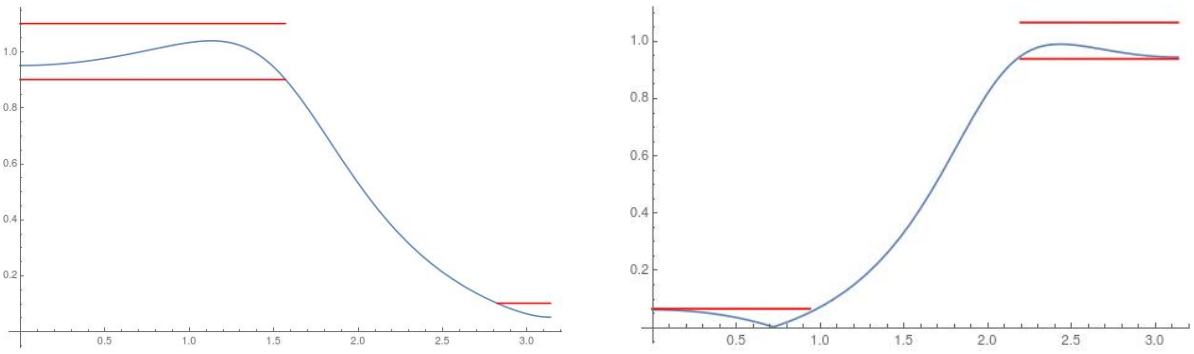
However, lowpass and highpass filters that can be handle with second order IIR filters are proposed Table 8. Their solving times are given Table 9. In this last table we also give the number of adder used for their implementation, the value of these adders and the coefficient values we found. This experiment was done on the same computer as for MCM problems: Linux 64bit (Ubuntu 18.04.4 LTS), with 16Go of DDR3 RAM and an Intel® Core™ i5-4570S, quadcore (2.90GHz).

The instances used for those experiments are initially FIR filter instances that we manually simplified to be handled with a second order IIR filter. The obtained results are found in a reasonable time and are certified optimal for multiplierless second order IIR filters. The transfer functions we found with our method for lowpass 10bit and highpass 8bit are represented Figure 10. Our experiments show that for a filter with five coefficients we are able to obtain hardware implementations that only involve a few adders, down to a single one. Furthermore a filter coefficient was even fixed to zero for lowpass 10bit. Surprisingly the precomputation of coefficient bounds is essential for highpass 8bit while it should be avoided for lowpass 10bit. The extreme difference of utility of the precomputation does not permit to conclude anything on its usage and more research on this aspect is needed. The use of the symmetry breaking constraint allows speeding up computation, with and without precomputations. Finally, it has been shown that precomputing a coefficient ranges or b coefficient ranges first has an impact on the precomputation time. However, it is not clear that a choice prevails on the other: For lowpass 8bit and highpass 8bit a's before b's was faster with and without the symmetry breaking constraint, however for lowpass 10bit without the symmetry breaking constraint the solving time was divided by more than three. As the symmetry breaking constraint seems to give better results and should be kept in the final method we find that precomputing a coefficient ranges first is more efficient.

5. Conclusion and perspectives

> Reserve your right to think, for even to think wrongly is better than not to think at all.
>
> Hypatia of Alexandra

We proposed a method to design optimal second order IIR filters with respect to the number of adders used in their hardware implementation. Our method is based on ILP modeling but not only as it makes use



(a) Lowpass 10bit

(b) Highpass 8bit

Figure 10: Examples of obtained second order IIR filters

of the quadratic capabilities of MILP solvers. Results are promising: For an instance it has been shown that only one adder suffices to implement in hardware a reasonable second order IIR filter. We tackled the stability issue by obtaining constraints that allows ensuring stability when designing second order IIR filter coefficients in fixed-point arithmetic. We limited the search space by finding bounds on coefficients and we linked the filter specifications with the hardware implementation optimization using two MCM models.

The MCM problem is an important basic block of the design of second order IIR filters. We proposed a NLP formulation as an alternative to ILP formulations and we proposed efficient variants to known ILP formulations. Our variants have shown to be competitive with the literature and future work on this would be to understand when and why a variant behave greater than the others. We left aside the complexity analysis of MCM problem, yet this should probably be tackled since the conjectured complexity might no be applicable for practical variants. Thus, it is not clear that faster solving algorithms are not possible. In either case, we think that a dedicated branch and bound for this problem as a great potential and could be of great use particularly for harder instances.

We extended a design process to second order IIR filters. Despite this is an essential step, being restricted to second order is limiting and our future work on this would be to propose an algorithm that decomposes a greater order filter into second order sections. Such algorithms already exist for the classical design of filters and we are confident that we can do the same in the context of our variant.

References

[1] M. Kumm, A. Volkova, S.-I. Filip, Design of Optimal Multiplierless FIR Filters (2019).

[2] M. Kumm, Optimal Constant Multiplication Using Integer Linear Programming, IEEE Transactions on Circuits and Systems II: Express Briefs 65 (5) (2018) 567–571. doi:10.1109/TCSII.2018.2823780.

[3] M. Kumm, Multiple Constant Multiplication Optimizations for Field Programmable Gate Arrays, Springer Fachmedien Wiesbaden, Wiesbaden, 2016. doi:10.1007/978-3-658-13323-8_1.

[4] CPLEX, CPLEX User's Manual (2020).
URL https://www.ibm.com/analytics/cplex-optimizer

[5] Gurobi Optimization LLC, Gurobi Optimizer Reference Manual (2020).
URL https://www.gurobi.com/

[6] A. Gleixner, M. Bastubbe, L. Eifler, T. Gally, G. Gamrath, R. L. Gottwald, G. Hendel, C. Hojny, T. Koch, M. E. Lübbecke, S. J. Maher, M. Miltenberger, B. Müller, M. E. Pfetsch, C. Puchert, D. Rehfeldt, F. Schlösser, C. Schubert, F. Serrano, Y. Shinano, J. M. Viernickel, M. Walter, F. Wegscheider, J. T. Witt, J. Witzig, The SCIP Optimization Suite 6.0, software, Optimization Online (2018).
URL https://scip.zib.de/

[7] I. Dunning, J. Huchette, M. Lubin, JuMP: A Modeling Language for Mathematical Optimization, SIAM Review 59 (2) (2017) 295–320. doi:10.1137/15M1020575.

[8] P. Sittel, T. Schönwälder, M. Kumm, 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.
URL http://www.uni-kassel.de/go/scalp

[9] A. Antoniou, Digital Filters: Analysis, Design, and Signal Processing Applications, McGraw-Hill Education, New York, 2018.
URL https://www.mheducation.com/highered/product/digital-filters-analysis-design-signal-processing-applicati
9780071846035.html

[10] P. Prandoni, M. Vetterli, Signal Processing for Communications, EPFL Press, 2008.
URL https://www.sp4comm.org/

[11] D. Shi, 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 58 (1) (2011) 126–136. doi:10.1109/TCSI.2010.2055290.

[12] F. de Dinechin, J. Detrey, O. Cret, R. Tudoran, When FPGAs Are Better at Floating-Point than Microprocessors, in: Proceedings of the 16th International ACM/SIGDA Symposium on Field Programmable Gate Arrays, FPGA '08, Association for Computing Machinery, New York, NY, USA, 2008, p. 260. doi:10.1145/1344671.1344717.

[13] R. Bernstein, Multiplication by integer constants, Software: Practice and Experience 16 (7) (1986) 641–652. doi:10.1002/spe.4380160704.

[14] R. M. Hewlitt, E. S. Swartzlantler, Canonical signed digit representation for FIR digital filters, in: 2000 IEEE Workshop on SiGNAL PROCESSING SYSTEMS. SiPS 2000. Design and Implementation (Cat. No.00TH8528), 2000, pp. 416–426. doi:10.1109/SIPS.2000.886740.

[15] E. Backenius, E. Säll, Two's Complement Conversion to Minimal Signed Digit Code (2005).
URL https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.483.6316

[16] O. Gustafsson, Lower Bounds for Constant Multiplication Problems, IEEE Transactions on Circuits and Systems II: Express Briefs 54 (11) (2007) 974–978. doi:10.1109/TCSII.2007.903212.

[17] J. Thong, N. Nicolici, An Optimal and Practical Approach to Single Constant Multiplication, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 30 (9) (2011) 1373–1386. doi:10.1109/TCAD.2011.2153853.

[18] V. Lefèvre, Multiplication by an Integer Constant, Research Report RR-4192, INRIA (2001).
URL https://hal.inria.fr/inria-00072430

[19] A. G. Dempster, M. D. Macleod, Constant integer multiplication using minimum adders, IEE Proceedings - Circuits, Devices and Systems 141 (5) (1994) 407–413. doi:10.1049/ip-cds:19941191.

[20] N. J. A. Sloane, S. Plouffe, The Encyclopedia of Integer Sequences, Academic Press, 1995.
URL https://oeis.org/A000265

[21] E. Klotz, A. M. Newman, Practical guidelines for solving difficult mixed integer linear programs, Surveys in Operations Research and Management Science 18 (1) (2013) 18–32. doi:https://doi.org/10.1016/j.sorms.2012.12.001.

[22] H. Leich, Toolbox for the design of IIR digital filters, in: Proceedings of 13th International Conference on Digital Signal Processing, Vol. 2, 1997, pp. 621–624. doi:10.1109/ICDSP.1997.628426.

[23] M. Kumm, A. Volkova, S.-I. Filip, FIRopt (2020).
URL https://gitlab.com/filteropt/firopt

[24] G. Vanuytsel, P. Boets, L. Van Biesen, 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 (IEEE Cat. No.00CH37276), Vol. 1, 2002, pp. 793–797. doi:10.1109/IMTC.2002.1006943.

[25] R. Hourani, H. Alassaly, W. Alexander, Hardware implementation of IIR digital filters for programmable devices, in: 2013 IEEE 20th International Conference on Electronics, Circuits, and Systems (ICECS), 2013, pp. 783–786. doi:10.1109/ICECS.2013.6815531.

[26] A. Billionnet, S. Elloumi, A. Lambert, Linear Reformulations of Integer Quadratic Programs, in: H. A. Le Thi, P. Bouvry, T. Pham Dinh (Eds.), Modelling, Computation and Optimization in Information Systems and Management Sciences, Springer Berlin Heidelberg, Berlin, Heidelberg, 2008, pp. 43–51. doi:10.1007/978-3-540-87477-5_5.

[27] Z. Smékal, 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 (Cat. No.99EX357), Vol. 1, 1999, pp. 145–148. doi:10.1109/ICECS.1999.812244.

[28] L. D. Milić, M. D. Lutovac, Design of Multiplierless Elliptic IIR Filters with a Small Quantization Error, IEEE Transactions on Signal Processing 47 (2) (1999) 469–479. doi:10.1109/78.740130.

Appendix A. Expanding $\left|H\left(e^{i\omega}\right)\right|^2$

$$\left|H\left(e^{i\omega}\right)\right| = \left|\frac{b_0\,(\cos(\omega)+i\sin(\omega))^2 + b_1\,(\cos(\omega)+i\sin(\omega)) + b_2}{(\cos(\omega)+i\sin(\omega))^2 + a_1\,(\cos(\omega)+i\sin(\omega)) + a_2}\right| \tag{A.1}$$

$$\Leftrightarrow \left|H\left(e^{i\omega}\right)\right| = \frac{\left|b_0\left(\cos^2(\omega)-\sin^2(\omega)\right) + 2b_0 i\cos(\omega)\sin(\omega) + b_1\cos(\omega) + b_1 i\sin(\omega) + b_2\right|}{\left|\left(\cos^2(\omega)-\sin^2(\omega)\right) + 2i\cos(\omega)\sin(\omega) + a_1\cos(\omega) + a_1 i\sin(\omega) + a_2\right|} \tag{A.2}$$

$$\overset{(P1)}{\Leftrightarrow} \left|H\left(e^{i\omega}\right)\right|^2 = \frac{|b_0\cos(2\omega) + 2b_0 i\cos(\omega)\sin(\omega) + b_1\cos(\omega) + b_1 i\sin(\omega) + b_2|^2}{|\cos(2\omega) + 2i\cos(\omega)\sin(\omega) + a_1\cos(\omega) + a_1 i\sin(\omega) + a_2|^2} \tag{A.3}$$

$$\Leftrightarrow \left|H\left(e^{i\omega}\right)\right|^2 = \frac{(b_0\cos(2\omega) + b_1\cos(\omega) + b_2)^2 + (2b_0\cos(\omega)\sin(\omega) + b_1\sin(\omega))^2}{(\cos(2\omega) + a_1\cos(\omega) + a_2)^2 + (2\cos(\omega)\sin(\omega) + a_1\sin(\omega))^2} \tag{A.4}$$

$$\Leftrightarrow \left|H\left(e^{i\omega}\right)\right|^2 = \frac{b_0{}^2\cos^2(2\omega) + b_1{}^2\cos^2(\omega) + b_2{}^2 + 2b_0b_1\cos(2\omega)\cos(\omega) + 2b_0b_2\cos(2\omega) + 2b_1b_2\cos(\omega)}{\cos^2(2\omega) + a_1{}^2\cos^2(\omega) + a_2{}^2 + 2a_1\cos(2\omega)\cos(\omega) + 2a_2\cos(2\omega) + 2a_1a_2\cos(\omega)} +$$
$$\frac{b_0{}^2\,(2\cos(\omega)\sin(\omega))^2 + b_1{}^2\sin^2(\omega) + 4b_0b_1\cos(\omega)\sin^2(\omega)}{(2\cos(\omega)\sin(\omega))^2 + a_1{}^2\sin^2(\omega) + 4a_1\cos(\omega)\sin^2(\omega)} \tag{A.5}$$

$$\overset{(P2)}{\Leftrightarrow} \left|H\left(e^{i\omega}\right)\right|^2 = \frac{b_0{}^2\left(\cos^2(2\omega)+\sin^2(2\omega)\right) + b_1{}^2\left(\cos^2(\omega)+\sin^2(\omega)\right) + b_2{}^2}{\cos^2(2\omega)+\sin^2(2\omega) + a_1{}^2\left(\cos^2(\omega)+\sin^2(\omega)\right) + a_2{}^2} +$$
$$\frac{2b_0b_1\left(\cos(2\omega)\cos(\omega) + 2\cos(\omega)\sin^2(\omega)\right) + 2b_0b_2\cos(2\omega) + 2b_1b_2\cos(\omega)}{2a_1\left(\cos(2\omega)\cos(\omega) + 2\cos(\omega)\sin^2(\omega)\right) + 2a_2\cos(2\omega) + 2a_1a_2\cos(\omega)} \tag{A.6}$$

$$\overset{(P3)}{\Leftrightarrow} \left|H\left(e^{i\omega}\right)\right|^2 = \frac{b_0{}^2 + b_1{}^2 + b_2{}^2 + 2b_0b_1\cos(\omega) + 2b_0b_2\cos(2\omega) + 2b_1b_2\cos(\omega)}{1 + a_1{}^2 + a_2{}^2 + 2a_1\cos(\omega) + 2a_2\cos(2\omega) + 2a_1a_2\cos(\omega)} \tag{A.7}$$

Using the following properties:

(P1) $\cos^2(\omega) - \sin^2(\omega) = \cos(2\omega)$ is used to obtain (A.3);

(P2) $2\cos(\omega)\sin(\omega) = \sin(2\omega)$ is used to obtain (A.6);

(P3) $\cos^2(\omega) + \sin^2(\omega) = 1$ and $\cos(2\omega) = 1 - 2\sin^2(\omega)$ are used to obtain (A.7).

Appendix B. Second order IIR filter ILP model

$$b_{0,0} + b_{1,1} + b_{2,2} + 2b_{0,2}\cos 2\omega + 2b_{1,2}\cos\omega + 2b_{0,1}\cos\omega \geq \underline{D}(\omega)^2 \times \left(2^{2d-2} + a_{1,1} + a_{2,2} + 2^d a_2 \cos 2\omega + 2a_{1,2}\cos\omega + 2^d a_1 \cos\omega\right) \quad \forall\omega \in \Omega_d \tag{C4.1}$$

$$b_{0,0} + b_{1,1} + b_{2,2} + 2b_{0,2}\cos 2\omega + 2b_{1,2}\cos\omega + 2b_{0,1}\cos\omega \leq \overline{D}(\omega)^2 \times \left(2^{2d-2} + a_{1,1} + a_{2,2} + 2^d a_2 \cos 2\omega + 2a_{1,2}\cos\omega + 2^d a_1 \cos\omega\right) \quad \forall\omega \in \Omega_d \tag{C4.2}$$

$$b_0 \geq b_2^+ \tag{C4.3}$$

Figure B.11: Constraints with respect to specifications and symmetric breaking constraint

$$a_1 \geq -2^d + 1 \tag{C4.4}$$

$$a_1 \leq 2^d - 1 \tag{C4.5}$$

$$a_2 \geq -2^{d-1} + 1 \tag{C4.6}$$

$$a_2 \geq 2^{d-1} + 1 \tag{C4.7}$$

Figure B.12: Constraints with respect to stability

$$b_i \geq -2^d \left(1 - b_i^{\text{sg}}\right) \qquad \forall i \in [\![0,2]\!] \qquad \text{(C4.8)}$$

$$b_i \leq 2^d b_i^{\text{sg}} \qquad \forall i \in [\![0,2]\!] \qquad \text{(C4.9)}$$

$$b_i^+ \geq b_i \qquad \forall i \in [\![0,2]\!] \qquad \text{(C4.10)}$$

$$b_i^+ \leq b_i + 2 \times 2^d \left(1 - b_i^{\text{sg}}\right) \qquad \forall i \in [\![0,2]\!] \qquad \text{(C4.11)}$$

$$b_i^+ \geq -b_i \qquad \forall i \in [\![0,2]\!] \qquad \text{(C4.12)}$$

$$b_i^+ \leq -b_i + 2 \times 2^d b_i^{\text{sg}} \qquad \forall i \in [\![0,2]\!] \qquad \text{(C4.13)}$$

$$b_i^+ = \sum_{j=0}^{d} 2^j t_{b_i,j} \qquad \forall i \in [\![0,2]\!] \qquad \text{(C4.14)}$$

$$u_{b_i,b_{i'},j} \leq 2^d t_{b_{i'},j} \qquad \forall i \in [\![0,2]\!], i' \in [\![i,2]\!], j \in [\![0,d]\!] \qquad \text{(C4.15)}$$

$$u_{b_i,b_{i'},j} \leq b_i^+ \qquad \forall i \in [\![0,2]\!], i' \in [\![i,2]\!], j \in [\![0,d]\!] \qquad \text{(C4.16)}$$

$$u_{b_i,b_{i'},j} \geq b_i^+ - 2^d \left(1 - t_{b_{i'},j}\right) \qquad \forall i \in [\![0,2]\!], i' \in [\![i,2]\!], j \in [\![0,d]\!] \qquad \text{(C4.17)}$$

$$b_{i,i} = \sum_{j=0}^{d} 2^j u_{b_i,b_i,j} \qquad \forall i \in [\![0,2]\!] \qquad \text{(C4.18)}$$

$$b_{i,i'} \geq \sum_{j=0}^{d} 2^j u_{b_i,b_{i'},j} - 2 \times 2^{2d} \left(b_i^{\text{sg}} + b_{i'}^{\text{sg}}\right) \qquad \forall i \in [\![0,1]\!], i' \in [\![i+1,2]\!] \qquad \text{(C4.19)}$$

$$b_{i,i'} \geq \sum_{j=0}^{d} 2^j u_{b_i,b_{i'},j} - 2 \times 2^{2d} \left(2 - b_i^{\text{sg}} - b_{i'}^{\text{sg}}\right) \qquad \forall i \in [\![0,1]\!], i' \in [\![i+1,2]\!] \qquad \text{(C4.20)}$$

$$b_{i,i'} \leq \sum_{j=0}^{d} 2^j u_{b_i,b_{i'},j} \qquad \forall i \in [\![0,1]\!], i' \in [\![i+1,2]\!] \qquad \text{(C4.21)}$$

$$b_{i,i'} \geq -\sum_{j=0}^{d} 2^j u_{b_i,b_{i'},j} \qquad \forall i \in [\![0,1]\!], i' \in [\![i+1,2]\!] \qquad \text{(C4.22)}$$

$$b_{i,i'} \leq -\sum_{j=0}^{d} 2^j u_{b_i,b_{i'},j} + 2 \times 2^{2d} \left(1 - b_i^{\text{sg}} + b_{i'}^{\text{sg}}\right) \qquad \forall i \in [\![0,1]\!], i' \in [\![i+1,2]\!] \qquad \text{(C4.23)}$$

$$b_{i,i'} \leq -\sum_{j=0}^{d} 2^j u_{b_i,b_{i'},j} + 2 \times 2^{2d} \left(1 + b_i^{\text{sg}} - b_{i'}^{\text{sg}}\right) \qquad \forall i \in [\![0,1]\!], i' \in [\![i+1,2]\!] \qquad \text{(C4.24)}$$

Figure B.13: Constraints to linearize quadratic terms containing b's

$$a_i \geq -2^d \left(1 - a_i^{\text{sg}}\right) \qquad \forall i \in [\![1,2]\!] \qquad \text{(C4.25)}$$

$$a_i \leq 2^d a_i^{\text{sg}} \qquad \forall i \in [\![1,2]\!] \qquad \text{(C4.26)}$$

$$a_i^+ \geq a_i \qquad \forall i \in [\![1,2]\!] \qquad \text{(C4.27)}$$

$$a_i^+ \leq a_i + 2 \times 2^d \left(1 - a_i^{\text{sg}}\right) \qquad \forall i \in [\![1,2]\!] \qquad \text{(C4.28)}$$

$$a_i^+ \geq -a_i \qquad \forall i \in [\![1,2]\!] \qquad \text{(C4.29)}$$

$$a_i^+ \leq -a_i + 2 \times 2^d a_i^{\text{sg}} \qquad \forall i \in [\![1,2]\!] \qquad \text{(C4.30)}$$

$$a_i^+ = \sum_{j=0}^{d} 2^j t_{a_i,j} \qquad \forall i \in [\![1,2]\!] \qquad \text{(C4.31)}$$

$$u_{a_i,a_{i'},j} \leq 2^d t_{a_{i'},j} \qquad \forall i \in [\![1,2]\!], i' \in [\![i,2]\!], j \in [\![0,d]\!] \qquad \text{(C4.32)}$$

$$u_{a_i,a_{i'},j} \leq a_i^+ \qquad \forall i \in [\![1,2]\!], i' \in [\![i,2]\!], j \in [\![0,d]\!] \qquad \text{(C4.33)}$$

$$u_{a_i,a_{i'},j} \geq a_i^+ - 2^d \left(1 - t_{a_{i'},j}\right) \qquad \forall i \in [\![1,2]\!], i' \in [\![i,2]\!], j \in [\![0,d]\!] \qquad \text{(C4.34)}$$

$$a_{i,i} = \sum_{j=0}^{d} 2^j u_{a_i,a_i,j} \qquad \forall i \in [\![1,2]\!] \qquad \text{(C4.35)}$$

$$a_{i,i'} \geq \sum_{j=0}^{d} 2^j u_{a_i,a_{i'},j} - 2 \times 2^{2d} \left(a_i^{\text{sg}} + a_{i'}^{\text{sg}}\right) \qquad \forall i \in [\![1,1]\!], i' \in [\![i+1,2]\!] \qquad \text{(C4.36)}$$

$$a_{i,i'} \geq \sum_{j=0}^{d} 2^j u_{a_i,a_{i'},j} - 2 \times 2^{2d} \left(2 - a_i^{\text{sg}} - a_{i'}^{\text{sg}}\right) \qquad \forall i \in [\![1,1]\!], i' \in [\![i+1,2]\!] \qquad \text{(C4.37)}$$

$$a_{i,i'} \leq \sum_{j=0}^{d} 2^j u_{a_i,a_{i'},j} \qquad \forall i \in [\![1,1]\!], i' \in [\![i+1,2]\!] \qquad \text{(C4.38)}$$

$$a_{i,i'} \geq -\sum_{j=0}^{d} 2^j u_{a_i,a_{i'},j} \qquad \forall i \in [\![1,1]\!], i' \in [\![i+1,2]\!] \qquad \text{(C4.39)}$$

$$a_{i,i'} \leq -\sum_{j=0}^{d} 2^j u_{a_i,a_{i'},j} + 2 \times 2^{2d} \left(1 - a_i^{\text{sg}} + a_{i'}^{\text{sg}}\right) \qquad \forall i \in [\![1,1]\!], i' \in [\![i+1,2]\!] \qquad \text{(C4.40)}$$

$$a_{i,i'} \leq -\sum_{j=0}^{d} 2^j u_{a_i,a_{i'},j} + 2 \times 2^{2d} \left(1 + a_i^{\text{sg}} - a_{i'}^{\text{sg}}\right) \qquad \forall i \in [\![1,1]\!], i' \in [\![i+1,2]\!] \qquad \text{(C4.41)}$$

Figure B.14: Constraints to linearize quadratic terms containing a's

$$c_0 = 1 \tag{C4.42}$$

$$c_a^{\text{nsh}} = c_{a,l}^{\text{sh,sg}} + c_{a,r}^{\text{sh,sg}} \qquad \forall a \in [\![1, N_A]\!] \tag{C4.43}$$

$$c_a^{\text{nsh}} \geq 2^{-s} c_a + (\Psi_{a,s} - 1) \times M \qquad \forall a \in [\![1, N_A]\!], s \in [\![S_{\min}, 0]\!] \tag{C4.44}$$

$$c_a^{\text{nsh}} \leq 2^{-s} c_a + (1 - \Psi_{a,s}) \times M \qquad \forall a \in [\![1, N_A]\!], s \in [\![S_{\min}, 0]\!] \tag{C4.45}$$

$$c_a = 2c_a^{\text{odd}} + 1 \qquad \forall a \in [\![1, N_A]\!] \tag{C4.46}$$

$$\sum_{s=S_{\min}}^{0} \Psi_{a,s} = 1 \qquad \forall a \in [\![1, N_A]\!] \tag{C4.47}$$

$$\phi_{a,r,0} \leq \sum_{s=S_{\min}}^{-1} \Psi_{a,s} \qquad \forall a \in [\![1, N_A]\!] \tag{C4.48}$$

$$c_{a,i} \geq c_k + (c_{a,i,k} - 1) \times M \qquad \forall a \in [\![1, N_A]\!], i \in \{l, r\}, \forall k \in [\![0, a-1]\!] \tag{C4.49}$$

$$c_{a,i} \leq c_k + (1 - c_{a,i,k}) \times M \qquad \forall a \in [\![1, N_A]\!], i \in \{l, r\}, \forall k \in [\![0, a-1]\!] \tag{C4.50}$$

$$\sum_{k=0}^{a-1} c_{a,r,k} = 1 \qquad \forall a \in [\![1, N_A]\!] \tag{C4.51}$$

$$c_{a,r}^{\text{sh}} \geq 2^s c_{a,r} + (\phi_{a,r,s} - 1) \times M \qquad \forall a \in [\![1, N_A]\!], s \in [\![0, S_{\max}]\!] \tag{C4.52}$$

$$c_{a,r}^{\text{sh}} \leq 2^s c_{a,r} + (1 - \phi_{a,r,s}) \times M \qquad \forall a \in [\![1, N_A]\!], s \in [\![0, S_{\max}]\!] \tag{C4.53}$$

$$c_{a,l}^{\text{sh}} = c_{a,l} \qquad \forall a \in [\![1, N_A]\!] \tag{C4.54}$$

$$\sum_{s=0}^{S_{\max}} \phi_{a,r,s} = 1 \qquad \forall a \in [\![1, N_A]\!] \tag{C4.55}$$

$$c_{a,i}^{\text{sh,sg}} \geq -c_{a,i}^{\text{sh}} + (\Phi_{a,i} - 1) \times M \qquad \forall a \in [\![1, N_A]\!], i \in \{l, r\} \tag{C4.56}$$

$$c_{a,i}^{\text{sh,sg}} \leq -c_{a,i}^{\text{sh}} + (1 - \Phi_{a,i}) \times M \qquad \forall a \in [\![1, N_A]\!], i \in \{l, r\} \tag{C4.57}$$

$$c_{a,i}^{\text{sh,sg}} \geq c_{a,i}^{\text{sh}} - (\Phi_{a,i}) \times M \qquad \forall a \in [\![1, N_A]\!], i \in \{l, r\} \tag{C4.58}$$

$$c_{a,i}^{\text{sh,sg}} \leq c_{a,i}^{\text{sh}} + (\Phi_{a,i}) \times M \qquad \forall a \in [\![1, N_A]\!], i \in \{l, r\} \tag{C4.59}$$

$$\Phi_{a,l} + \Phi_{a,r} \leq 1 \qquad \forall a \in [\![1, N_A]\!] \tag{C4.60}$$

(C4.61)

Figure B.15: MCM 1 and MCM 2

$$mcm^a{}_0 = 1 \tag{C4.62}$$

$$mcm^b{}_0 = 1 \tag{C4.63}$$

$$samemcm_{0,k} = 1 \qquad \forall k \in [\![1, NA]\!] \tag{C4.64}$$

$$mcm^a{}_k + mcm^b{}_k = 1 \qquad \forall k \in [\![1, NA]\!] \tag{C4.65}$$

$$mcm^a{}_k + mcm^a{}_{k'} + 1 = 2mcm_{k,k'} + samemcm_{k,k'} \qquad \forall k \in [\![1, NA]\!], k' \in [\![k+1, NA]\!] \tag{C4.66}$$

$$c_{a,i,k} \leq samemcm_{k,a} \qquad \forall a \in [\![1, NA]\!], k \in [\![0, a-1]\!] \tag{C4.67}$$

$$a_j \geq (-1)^\Phi 2^s c_a + \left(oa_{a,j,s,\Phi} - 1\right) \times M \qquad \forall a \in [\![0, N_A]\!], j \in [\![1,2]\!], s \in [\![0, S_{\max}]\!], \Phi \in \{0,1\} \tag{C4.68}$$

$$a_j \leq (-1)^\Phi 2^s c_a + \left(1 - oa_{a,j,s,\Phi}\right) \times M \qquad \forall a \in [\![0, N_A]\!], j \in [\![1,2]\!], s \in [\![0, S_{\max}]\!], \Phi \in \{0,1\} \tag{C4.69}$$

$$\sum_{k=0}^{N_A} \sum_{s=0}^{S_{\max}} \sum_{\Phi=0}^{1} oa_{k,j,s,\Phi} = 1 \qquad \forall j \in [\![1,2]\!] \tag{C4.70}$$

$$oa_{k,j,s,\Phi} \leq mcm^a{}_k \qquad \forall k \in [\![1, NA]\!], j \in [\![1,2]\!], s \in [\![1, S_{\max}]\!], \Phi \in \{0,1\} \tag{C4.71}$$

$$b_j \geq (-1)^\Phi 2^s c_a + \left(ob_{k,j,s,\Phi} - 1\right) \times M \qquad \forall k \in [\![0, N_A]\!], j \in [\![0,2]\!], s \in [\![0, S_{\max}]\!], \Phi \in \{0,1\} \tag{C4.72}$$

$$b_j \leq (-1)^\Phi 2^s c_a + \left(1 - ob_{k,j,s,\Phi}\right) \times M \qquad \forall k \in [\![0, N_A]\!], j \in [\![0,2]\!], s \in [\![0, S_{\max}]\!], \Phi \in \{0,1\} \tag{C4.73}$$

$$\sum_{k=0}^{N_A} \sum_{s=0}^{S_{\max}} \sum_{\Phi=0}^{1} ob_{k,j,s,\Phi} = 1 \qquad \forall j \in [\![0,2]\!] \tag{C4.74}$$

$$ob_{a,j,s,\Phi} \leq mcm^b{}_k \qquad \forall k \in [\![1, NA]\!], j \in [\![1,2]\!], s \in [\![1, S_{\max}]\!], \Phi \in \{0,1\} \tag{C4.75}$$

Figure B.16: Glue in-between filter constraints and MCM problems

$a_i \in [\![-2^d, 2^d]\!]$	$\forall i \in [\![1,2]\!]$
$a_i^+ \in [\![0, 2^d]\!]$	$\forall i \in [\![1,2]\!]$
$a_i^{\text{sg}} \in \{0,1\}$	$\forall i \in [\![1,2]\!]$
$t_{a_i,j} \in \{0,1\}$	$\forall i \in [\![1,2]\!], j \in [\![0,d]\!]$
$u_{a_i,a_{i'},j} \in \{0,1\}$	$\forall i \in [\![1,2]\!], i' \in [\![i,2]\!], j \in [\![0,d]\!]$
$a_{i,i'} \in [\![-2^{2d}, 2^{2d}]\!]$	$\forall i \in [\![1,2]\!], i' \in [\![1,2]\!]$
$b_i \in [\![-2^d, 2^d]\!]$	$\forall i \in [\![0,2]\!]$
$b_i^+ \in [\![0, 2^d]\!]$	$\forall i \in [\![0,2]\!]$
$b_i^{\text{sg}} \in \{0,1\}$	$\forall i \in [\![0,2]\!]$
$t_{b_i,j} \in \{0,1\}$	$\forall i \in [\![0,2]\!], j \in [\![0,d]\!]$
$u_{b_i,b_{i'},j} \in \{0,1\}$	$\forall i \in [\![0,2]\!], i' \in [\![i,2]\!], j \in [\![0,d]\!]$
$b_{i,i'} \in [\![-2^{2d}, 2^{2d}]\!]$	$\forall i \in [\![0,2]\!], i' \in [\![0,2]\!]$
$c_a \in [\![1, 2^d]\!]$	$\forall a \in [\![0, NA]\!]$
$c_a^{\text{nsh}} \in [\![1, 2^d]\!]$	$\forall a \in [\![1, NA]\!]$
$c_a^{\text{odd}} \in [\![0, 2^{d-1}]\!]$	$\forall a \in [\![1, NA]\!]$
$c_{a,i} \in [\![1, 2^d]\!]$	$\forall a \in [\![0, NA]\!], i \in \{l, r\}$
$c_{a,i}^{\text{sh}} \in [\![1, 2^d]\!]$	$\forall a \in [\![0, NA]\!], i \in \{l, r\}$
$c_{a,i}^{\text{sh,sg}} \in [\![-2^d, 2^d]\!]$	$\forall a \in [\![0, NA]\!], i \in \{l, r\}$
$\Phi_{a,i} \in \{0,1\}$	$\forall a \in [\![1, NA]\!], i \in \{l, r\}$
$c_{a,i,k} \in \{0,1\}$	$\forall a \in [\![1, NA]\!], i \in \{l, r\}, k \in [\![0, a-1]\!]$
$\phi_{a,r,s} \in \{0,1\}$	$\forall a \in [\![1, NA]\!], s \in [\![0, S_{\max}]\!]$
$\Psi_{a,s} \in \{0,1\}$	$\forall a \in [\![1, NA]\!], s \in [\![S_{\min}, 0]\!]$
$oa_{a,j,s,\Phi} \in \{0,1\}$	$\forall a \in [\![1, NA]\!], j \in [\![1,2]\!], s \in [\![1, S_{\max}]\!], \Phi \in \{0,1\}$
$ob_{a,j,s,\Phi} \in \{0,1\}$	$\forall a \in [\![1, NA]\!], j \in [\![0,2]\!], s \in [\![1, S_{\max}]\!], \Phi \in \{0,1\}$
$mcm^a{}_k \in \{0,1\}$	$\forall a \in [\![0, NA]\!]$
$mcm^b{}_k \in \{0,1\}$	$\forall k \in [\![0, NA]\!]$
$samemcm_{k,k'} \in \{0,1\}$	$\forall k \in [\![0, NA]\!], k' \in [\![k+1, NA]\!]$
$mcm_{k,k'} \in \{0,1\}$	$\forall k \in [\![1, NA]\!], k' \in [\![k+1, NA]\!]$

Figure B.17: Variables