Truncated Multiple Constant Multiplication with Minimal Number of Full Adders

Rémi Garcia, Anastasia Volkova, Martin Kumm

Truncated Multiple Constant Multiplication with Minimal Number of Full Adders

Rémi Garcia and Anastasia Volkova
Nantes Université, Centrale Nantes, CNRS, LS2N
Nantes, France
Email: firstname.lastname@univ-nantes.fr

Martin Kumm
Fulda University of Applied Sciences,
Fulda, Germany
Email: martin.kumm@cs.hs-fulda.de

***Abstract*—Many algorithms from digital signal processing, including digital filters or discrete transforms, require the multiplications with several constants. These can be efficiently implemented multiplierless by using additions, subtractions, and bit-shifts. Finding a multiplierless solution with minimal cost is known as the multiple constant multiplication (MCM) problem. Usually, not the full precision is required at the output. The state-of-the-art approaches consist in finding an MCM solution first, and truncating it in a second step. In this work, we solve the MCM problem with minimal number of full adders for truncated outputs. By combining the two steps into a global optimization problem, modeled through mixed-integer linear programming, we are able to reduce the number of full adders by 60% in best cases and by 12% on average. Our method has shown its efficiency on more than 80 instances from literature and permits a fast improvement of state-of-the-art results in most of the cases.**

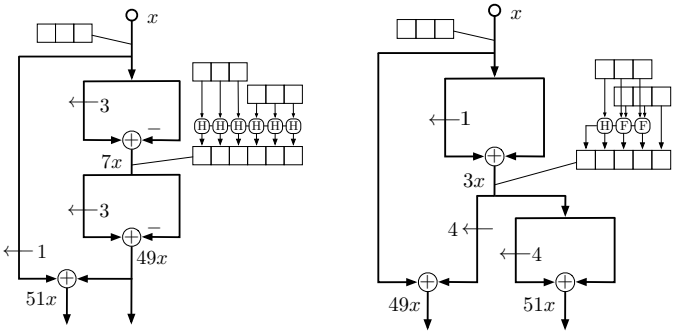
***Index Terms*—Multiple constant multiplication, digital arithmetic, circuit optimization, full adders**

I. Introduction

The multiple constant multiplication (MCM) is a frequently-used operation present in many numerical algorithms. It often appears in digital signal processing, in particular when evaluating digital filters, or in the parallel inference of deep neural networks, for instance as a part of dot-product evaluation. One common and efficient way of performing fixed-point (FxP) multiplications by constants is to transform them into additions/subtractions and bit-shifts. Moreover, internal data sharing is possible: for example, when computing both $49x$ and $51x$, the value $49x$ can be reused for both outputs (see Fig. 1a). We refer to the resulting MCM shift-and-add circuits, illustrated in Fig. 1, as adder graphs in the following.

To minimize the cost of the implemented hardware, most previous methods focused on searching for an adder graph that minimizes the number of adders [1]–[10]. The optimization problem behind is conjectured NP-hard [11] and has been tackled with heuristics [1], [5], [6], [12] and optimal approaches [7], [8], [13]. While giving a good idea on the actual resource consumption, counting adders is a high-level metric: different solutions having the same minimal number of adders often have different actual hardware cost. This can be refined by counting the total number of *full adders* (FA), or even *half adders* (HA) required for additions [14]–[19].

The number of full adders gives a much finer-grain low-level metric. For example, in Fig. 1 we see two solutions for multiplication by 49 and 51, both using 3 adders. However, the solution in Fig. 1a requires 23 FA and HAs, while the solution in Fig. 1b needs only 9 FA and/or HA, for a 3-bit input.



(a) Requires 23 FA/HA (b) Requires 9 FA/HA

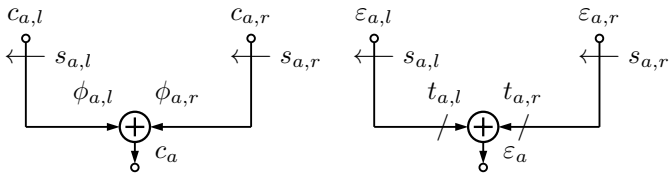
Fig. 1: MCM solutions computing $51x$ and $49x$ with 3 adders. Horizontal arrows indicate left shifts.

Moreover, in practical applications, the output is often truncated to a certain word length (WL), *e. g.*, in digital filter evaluation. Since there is no sense in computing unnecessary bits, further resource optimizations can be achieved by introducing truncations in intermediate operations. Naturally, the rounding errors due to truncations are propagated and combined to the output and should be constrained to the user-given "error budget". And again, some adder graph topologies are better suited for truncations than others. Hence, the truncations and the corresponding error quantities should be incorporated into the overall MCM model.

Previous works solve this problem only partially: [7], [8], [13] only considers MCM problem but solves optimally; the work in [19] optimizes truncations optimally but only for a fixed adder graph; the authors of [14]–[16] consider MCM with FA metric but solve only heuristically.

In this paper we optimally solve the general truncated MCM problem including FA metric in one step: given a set of constants and an *a priori* accuracy of the result, we produce an adder graph describing the shifts, the adders and truncations that result in minimal number of FAs. We present a Mixed-Integer Linear Programming (MILP) model that can be efficiently solved using any generic solver.

Fig. 2 illustrates a graphical representation of the variables of the MILP model. Our first contribution is the in-depth



(a) Adder in classic MCM [13] (b) Adder in presence of errors ε and truncations t

Fig. 2: Classic (left) and proposed (right) adder models.

modification of the classic MCM model (left) to handle the FA metric. A second contribution is the extension of our model to handle truncations and error propagation (right) in order to gain more FAs by allowing for topology modification as we search for a solution. Our new approach yields adder graphs that reduce the FA count by 12%, on average, and up to 60% in the best case, w. r. t. the state-of-the-art MCM design. The complete MILP model is available with the open-source implementation[1].

II. A New MILP Model Minimizing Full Adders

A. Problem Formulation

The inputs of this MCM problem with an FA metric (MCM_{FA}) are: the target constants, the maximum acceptable error of the outputs (which is usually derived from the output format) and the input format in terms of the most and the least significant bits (MSB and LSB), msb_{in} and lsb_{in}, respectively. The maximum WL of the internal constants can be given as an input too or inferred from the output coefficients. For simplicity in MILP modeling, in the following we consider that the LSB of the input is equal to 0, such that all quantities related to FxP formats are positive.

At output, the MCM_{FA} produces an adder graph with the minimal number of FAs. Each constant multiplication is defined as a sequence of shifts and adds, where each adder also bears the information on the truncation of its inputs.

We limit ourself to the minimization of full adders and do not distinguish between specialized half adders or even single gates as done in some previous work [14]–[16]. This distinction is trivial in the model but would make the presentation and comparison more complex. Instead, we indicate where the changes should be done in the model.

The key ingredients for the MCM_{FA} model are:

- Constraints for the adder graph topology;
- Modeling the count of FAs, for each adder;
- Modeling the truncations in intermediate adders as the FA gain and the impact of the induced truncation error.

In the following we give a general overview of these three key ingredients.

B. Constraints for General MCM

Kumm proposed an ILP model [13], centered on finding the values of intermediate adders and possible shifts as illustrated in Fig. 2a. These adders are constrained by the equation

$$c_a = (-1)^{\phi_{a,l}}\, 2^{s_{a,l}} c_{a,l} + (-1)^{\phi_{a,r}}\, 2^{s_{a,r}} c_{a,r} \tag{1}$$

[1]https://gitlab.com/filteropt/mcmfa

that links an adder, a, with its inputs in the adder graph, where ϕ, s and c correspond to the sign, the bit shift and the adder value at the left (l) and (r) inputs, respectively. Equation (1) can be linearized, either using indicator or big M constraints in order to be used as part of an ILP model. Indicator constraints are special constraints of the form

$$x = y \quad \text{if } z = 1, \tag{2}$$

in which a binary variable z controls whether or not a specified linear constraint is active. It can be rewritten in linear form by using the so-called big M constraint

$$y - (1-z)M \leq x \leq y + (1-z)M\ , \tag{3}$$

for a sufficiently large M. Indicator constraints are directly supported by many MILP solvers and are numerically more robust compared to so-called big M constraints but are typically slower due to weaker relaxations [20].

The linearization of (1) is done by introducing as many variables as there are possible shifts $\varphi_{a,i,s}$, signs $\phi_{a,i}$ and input sources $c_{a,i,k}$ for all possible combinations of target adders a and source adders k, of the inputs $i \in \{l, r\}$ and shifts s.

This linearization is described in detail in [13] and serves as a base for our MILP model.

C. Defining FA Metric

It should be first emphasized that FA metric is not equivalent to WL optimization. The number of FA necessary to perform an addition of two FxP numbers depends not only on the WLs but also MSB positions of the operands. In a general case, a result of addition of two n-bit numbers requires n FAs and a potential carry can increase the output WL. For that reason, computing the MSB of every adder is necessary. The number of full adders for each adder relies on multiple variables:

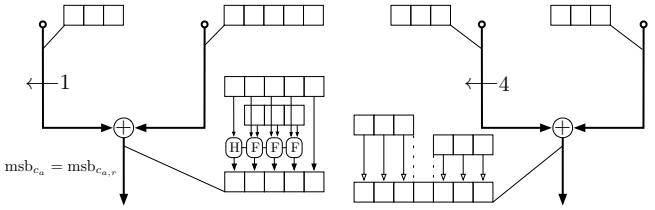
- $\text{FA}_a \in \mathbb{N}$ corresponds to the number of FA of adder a;
- $\text{msb}_a \in \mathbb{N}$ is the MSB position of the output of adder a, it corresponds to the maximum MSB of the inputs;
- $g_a \in \mathbb{N}$ corresponds to the number of full adders that can be saved with respect to the full-precision of adder a;
- $\psi_a \in \{0, 1\}$ corresponds to the presence of a necessary full adder for computing the result's MSB. For subtractions, the MSB of the adder is obtained from the last full adder for which the carry is known to be zero. In general, for additions, the value stored at the MSB is obtained with the carry of the last full adder. However, in some cases, a last full adder, for which the carry is known to be equal to zero, is necessary, as illustrated in Fig. 3a.

These variables are linked together as follows:

$$\text{FA}_a = \text{msb}_a - g_a + \psi_a, \qquad \forall \text{ adder } a, \tag{4}$$

where g_a is equal to the left shift if the right input is nonnegative, 0 otherwise. The value of g_a can be increased when an output error is allowed and truncations are involved. This potential gain is discussed in the next section.

As shown in Fig. 3b, in some rare cases it could be possible to totally remove full adders. It would require a small data path



(a) An additional half adder H is needed to compute the MSB.

(b) The adder inputs do not overlap and no FA is used.

Fig. 3: Corner cases when counting full adders

compared to the constant's WL. This is only the case for very low WLs, thus we left that outside of the scope of our model.

In our overall model we want to minimize the total number of full adders. The absence of objective in the initial MCM model permits to include the objective without effort:

$$\min \sum_a \text{FA}_a. \tag{5}$$

Finally, it is important to note that, during the solving process, the smallest MSB will be preferred over an unnecessarily large MSB as the model minimizes FA_a and, thus, msb_a. For that reason it is sufficient to ensure that the MSB of each adder is large enough to hold the result, which can be done with a single constraint

$$\text{msb}_a \geq \log_2 \left(2^{\text{msb}_{\text{in}}} c_a\right). \tag{6}$$

This constraint has to be linearized to avoid logarithm, to do so, we first exponentiate and obtain

$$2^{\text{msb}_a} \geq 2^{\text{msb}_{\text{in}}} c_a, \tag{7}$$

where $2^{\text{msb}_{\text{in}}}$ is a precomputed constant and c_a is the unknown coefficient corresponding to the adder in question. The expression 2^{msb_a} can be linearized in a similar way as explained in Section II-B. The challenge here is that the range for msb_a can be large. To speed up the resolution, we propose to relax the constraint for the intermediate integer variable designating 2^{msb_a} and use a continuous variable instead.

D. Modeling Truncation Error

In Fig. 2b, the position of truncations, $t_{a,i}$, and the flow of the error is represented for a given adder a. Truncations permits to decrease the number of full adders. This gain is modeled through the use of variables g_a in (4) and its value is defined similarly to [19] as:

$$g_a = \begin{cases} \max\left(t_{a,l} + s_{a,l}, t_{a,r}\right) & \text{if } s_{a,l} \geq 0 \wedge \phi_{a,l} = \phi_{a,r} \\ t_{a,l} + s_{a,l} & \text{if } s_{a,l} \geq 0 \wedge \phi_{a,l} = 1 \\ t_{a,r} & \text{if } s_{a,l} \geq 0 \wedge \phi_{a,r} = 1 \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

where variables $\phi_{a,i}$ handle the sign of each input as exposed in (1) and $t_{a,i}$ correspond to the truncations. These variables are closely linked to the error propagation constraints. Because of the maximum and the conditionals, (8) can be linearized using indicator or big M constraints. Note that (8) in combination with (4) can be similar defined to account for the number of half adders or gate level metrics. Then, their cost can be considered by trivially extending the objective (5) by weighting their area contribution.

The error computed after the addition corresponds to the sum of input errors which are either determined by previous errors in the circuits or by truncations. Error propagation constraints and truncations can be summarized as

$$\begin{aligned} \varepsilon_a = \max\left(2^{s_{a,l}}\varepsilon_{a,l}, 2^{t_{a,l}+s_{a,l}} - 1\right) \\ + \max\left(2^{s_{a,r}}\varepsilon_{a,r}, 2^{t_{a,r}+s_{a,r}} - 1\right) \end{aligned} \tag{9}$$

where $s_{a,l}$, $s_{a,r}$, $t_{a,l}$ and $t_{a,r}$ are variables for the bit shift and truncations, and require a linearization. The variables ε_a correspond to the error at adder a and, $\varepsilon_{a,l}$ and $\varepsilon_{a,r}$ are the error of the left and right inputs, respectively. These last variables are closely linked to the adder graph topology and they are linked to the right value using the constraint

$$\varepsilon_{a,i} = \varepsilon_k \quad \text{if } c_{a,i,k} = 1 \qquad \forall a, i \text{ and } k\ . \tag{10}$$

In a similar way, variables $s_{a,i}$ from (9) can be fixed to the right values with a constraint that uses $\varphi_{a,i,s}$.

Our goal is to gain as many bits as possible, for example by increasing the error as much as possible. To avoid an error too large, user given constants $\overline{\varepsilon}_j$ are necessary to bound the output errors. In the classical MCM, binary variables $o_{a,j}$ are equal to 1 if adder a is used for producing output j and 0 otherwise. Reusing this variable permits to bound the error:

$$\varepsilon_a \leq \overline{\varepsilon}_j \quad \text{if } o_{a,j} = 1 \qquad \forall a \text{ and } j. \tag{11}$$

Outside of the error propagation, the only constraint on errors is an upper bound. For that reason, the error propagation constraints have to ensure that the error is not underestimated. Thus, equality on (9) can be replaced by a greater or equal relation. This allows a straightforward linearization of the $\max$ function using intermediate variables. The linearization of the powers of two is done in the same way as for the classic MCM constraint involving shifts.

III. Experimental Results

We applied our model to benchmarks from image processing (11 instances) that have been used previously for MCM [9], [21] and to benchmarks from the FIRsuite project (60 instances) [22]. Models were implemented using the modeling language JuMP [23] and solved with Gurobi [24], with a time limit of 30 minutes and 4 threads from 1.1 GHz CPUs.

Our implementation allows to directly solve the model for a given set of target constants or to give an initial solution to the solver to optimize from (which is called *warm start* in the following). The initial solution relies on the fast RPAG [12] heuristic and an MILP model that counts the number of full adders. We consider the time for using heuristics and counting the number of full adders as negligible.

Because of big M constraints, there exists a risk of invalid solutions, due to numerical issues in the solver. For instances using WL strictly greater than 12 the solver might return

TABLE I: Results and comparison of the proposed MCM with FA metric. Inputs are 8-bits, and truncated outputs are 8-bits.

				#FA, no truncations			#FA, with truncations		
Benchmark	**WL**	**#coeff.**	**#adders**	**MCM [13]**	**Proposed**	**% gain**	**MCM [13] + trunc. [19]**	**Proposed**	**%gain**
GAUSSIAN_3	8	3	4	47	40	14.89 %	33	24	27.27 %
GAUSSIAN_5	11	3	5	57	57*	0.00 %	27	27*	0.00 %
HIGHPASS_5	7	4	4	46	39	15.22 %	34	20	41.18 %
HIGHPASS_9	7	5	5	54	47	12.96 %	35	25	28.57 %
HIGHPASS_15	9	12	12	142	105*	26.06 %	106	42*	60.38 %
LAPLACIAN_3	7	3	3	34	31	8.82 %	22	19	13.64 %
LOWPASS_5	7	5	6	73	60*	17.81 %	60	42*	30.00 %
LOWPASS_9	9	12	13*	152*	128*	15.79 %	91*	84*	7.69 %
LOWPASS_15	11	25	27*	306*	250*	18.30 %	158*	131*	17.09 %
UNSHARP_3	7	3	4	37	32	13.51 %	25	20	20.00 %
UNSHARP_3	11	3	5	67	49*	26.87 %	45	26*	42.22 %

*heuristic solution; no optimality could be proven within timeout

invalid adder graphs. Our tool checks the result a posteriori and ensures that only valid solutions are returned.

Due to the lack of space, we only report the results from the image processing benchmark [13] in TABLE I. The complete results, including the full FIRsuite benchmark, are available on the project's git.

A. MCM at Full Precision with FA Metric

In the first experiment, we evaluate our method using the full output precision without truncations (see "no truncations" column in TABLE I). As reference, we used the optimal ILP-based MCM [13]. When [13] could not provide an optimal solution within the timeout, we compared to the RPAG heuristic [12]. Finally, we computed the number of FAs in the reference solutions similarly to [19].

When using a warm start, in mere seconds our tool improves the starting state-of-the-art solution, in most cases. The rest of the time until the timeout, improvements were more rare or absent. For example, it took less than 5 seconds to obtain an objective value of 105, when solving `HIGHPASS_15`, and all the remaining time has been spent trying to improve it or to prove its optimality.

In many cases, using a warm start was necessary to simply the finding of a solution within a few minutes. In extremely rare cases, if the warm start solution is not of a good quality, e.g., the number of adders is large and non optimal, the design-space is too large and the solution after timeout could even be slightly worse than the state-of-the-art. In general, though, when not improving the reference result we usually prove that it is already optimal. We demonstrated, that in 9 cases out of 10 optimizing for the number of adders is non optimal and we could improve the FA count, on average, by 15% for the image processing benchmark and 12% for the whole benchmark including FIRsuite. In the best cases, we could achieve an improvement of more than 26%.

B. Truncated MCM vs. 2-step optimization

A second experiment consisted in comparing our approach to the state-of-the-art in presence of truncations, as long as a faithful rounding to the output format is guaranteed. The reference 2-step optimization is based on the optimal MCM [13] (or RPAG [12]) followed by optimal truncations [19]. For the image processing benchmark, we consider that the input WL matches the output WL, and is equal to 8 bits. In other words, the output accuracy requirement is that all 8 bits of the output are correct, except perhaps for the last bit.

It has been already observed in literature that allowing for truncations greatly decreases the number of FAs, compared to computing a full precision output. Indeed, TABLE I demonstrates that applying [13] + [19] on the image processing benchmark results in a 34% reduction in number of FAs, on average. With our approach, the improvements are pushed even further and an average FA reduction is around 43%.

When comparing our approach to [13] + [19], we observe a substantial improvement in the FA count: in the best cases we decrease the number of FAs by 60% and on average by 26%, and by 13% for the whole benchmark, including FIRsuite. Indeed, searching directly for an adder graph that allows for the best truncations, similarly to the example in Fig. 1, gives an optimal, or at least a better, result when compared to first fixing the adder graph topology. In our observations, the reduction degree really depends on the coefficient values, and not on the filter length or WL. Similarly to the experiment without truncations, good improvements were mostly obtained in the first few seconds using a warm start.

IV. Conclusion

With this work we proposed for the first time an optimal approach for the solution of the truncated MCM problem with the minimum number of FAs. In contrast to previous works, we consider both the FA metric and the truncations as part of the MCM optimization. Our work experimentally confirmed that large reductions are achievable, compared to a two-step approach consisting of the optimally solving the MCM [13] first and then optimally truncating [19]. Tests on the FIRSuite benchmark confirm the efficiency of our approach even on large instances. Even though some of the instances could not be solved optimally within 30 minutes, they nevertheless usually showed improvement in FA count.

The future work will focus on the digital filter design problems, for which the optimal MCM has been already used as a basic block. In particular, we plan to incorporate our new truncated MCM into the FIR filter design problem, extending the work in [25].

REFERENCES

[1] R. Bernstein, "Multiplication by integer constants," *Softw. - Pract. Exp.*, vol. 16, no. 7, pp. 641–652, Jul. 1986.

[2] D. R. Bull and D. H. Horrocks, "Primitive operator digital filters," *IEE Proc. G - Circuits, Devices and Syst.*, vol. 138, no. 3, pp. 401–412, Jun. 1991.

[3] A. G. Dempster and M. D. Macleod, "Constant integer multiplication using minimum adders," *IEE Proc. - Circuits, Devices and Syst.*, vol. 141, no. 5, pp. 407–413, Oct. 1994.

[4] V. Lefèvre, "Multiplication by an Integer Constant," INRIA, Research Report RR-4192, May 2001, Accessed: Jan., 2022. [Online]. Available: https://hal.inria.fr/inria-00072430

[5] Y. Voronenko and M. Püschel, "Multiplierless multiple constant multiplication," *ACM Trans. on Algorithms*, vol. 3, no. 2, p. 11, May 2007.

[6] O. Gustafsson, "A difference based adder graph heuristic for multiple constant multiplication problems." IEEE Int. Symp. Circuits Syst., 2007, pp. 1097 – 1100.

[7] ——, "Towards optimal multiple constant multiplication: A hypergraph approach," in *Asilomar Conf. Signals, Syst. Computers*, Oct. 2008, pp. 1805–1809.

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

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

[10] O. Gustafsson, "Lower Bounds for Constant Multiplication Problems," *IEEE Trans. Circuits Syst., II, Exp. Briefs*, vol. 54, no. 11, pp. 974–978, Nov. 2007.

[11] J. Thong and N. Nicolici, "An Optimal and Practical Approach to Single Constant Multiplication," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 30, no. 9, pp. 1373–1386, 2011.

[12] M. Kumm, K. Möller, M. Hardieck, and P. Zipf, "PAGSuite project website," 2021, Accessed: Jan., 2022. [Online]. Available: http://www.uni-kassel.de/go/pagsuite

[13] M. Kumm, "Optimal Constant Multiplication Using Integer Linear Programming," *IEEE Trans. Circuits Syst., II, Exp Briefs*, vol. 65, no. 5, pp. 567–571, May 2018.

[14] K. Johansson, O. Gustafsson, and L. Wanhammar, "A detailed complexity model for multiple constant multiplication and an algorithm to minimize the complexity," in *IEEE Proc. 2005 European Conf. Circuit Theory Design*, vol. 3, pp. 465–468.

[15] ——, "Bit-Level Optimization of Shift-and-Add Based FIR Filters," in *2007 IEEE Int. Conf. Electron., Circuits Syst.*, pp. 713–716.

[16] L. Aksoy, E. Costa, P. Flores, and J. Monteir, "Optimization of area in digital FIR filters using gate-level metrics," in *Proc. 44th Annu. Des. Autom. Conf.*, Jun. 2007, pp. 420–423.

[17] R. Guo, L. Wang, and L. S. DeBrunner, "A novel FIR filter implementation using truncated MCM technique," in *2009 Conf. Rec. 43th Asilomar Conf. Signals, Syst. Computers*, pp. 718–722.

[18] R. Guo, L. S. DeBrunner, and K. Johansson, "Truncated MCM using pattern modification for FIR filter implementation," in *Proc. 2010 IEEE Int. Symp. Circuits Syst.*, pp. 3881–3884.

[19] F. de Dinechin, S.-I. Filip, M. Kumm, and L. Forget, "Table-Based versus Shift-And-Add Constant Multipliers for FPGAs," in *2019 IEEE 26th Symp. Computer Arithmetic*, pp. 151–158.

[20] E. Klotz and A. M. Newman, "Practical guidelines for solving difficult mixed integer linear programs," *Surv. Oper. Res. Manag. Sci.*, vol. 18, no. 1, pp. 18–32, Oct. 2013.

[21] M. Kumm, D. Fanghänel, K. Möller, P. Zipf, and U. Meyer-Baese, "FIR filter optimization for video processing on FPGAs," *EURASIP J. Adv. Signal Process.*, vol. 2013, no. 1, May 2013.

[22] FIRsuite, "Suite of Constant Coefficient FIR Filters," 2021, Accessed: Jan., 2022. [Online]. Available: http://www.firsuite.net

[23] I. Dunning, J. Huchette, and M. Lubin, "JuMP: A Modeling Language for Mathematical Optimization," *SIAM Rev.*, vol. 59, no. 2, pp. 295–320, May 2017.

[24] Gurobi Optimization LLC, "Gurobi Optimizer Reference Manual," 2020, Accessed: Jan., 2022. [Online]. Available: https://www.gurobi.com/

[25] M. Kumm, A. Volkova, and S.-I. Filip, "Design of Optimal Multiplierless FIR Filters," 2019, *arXiv:1912.04210*.