PREPRINT, COPYRIGHT IEEE, TO APPEAR IN IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS–II: EXPRESS BRIEFS, SEE HTTPS://DOI.ORG/10.1109/TCSII.2018.28237801

Optimal Constant Multiplication using Integer Linear Programming

Martin Kumm, *Member, IEEE*

***Abstract*—Constant multiplication circuits can be realized multiplierless by using additions, subtractions and bit-shifts. The problem of finding a multiplication circuit with minimum adders and subtractors for a given constant (set of constants) is known as single (multiple) constant multiplication (SCM, MCM) problem. This work proposes a novel integer linear programming (ILP) formulation to optimally solve SCM and MCM problems. In contrast to previous ILP approaches, none of the possible intermediate constants have to be pre-computed as all additions are directly evaluated. This leads to fewer ILP variables and more compact models. Due to the flexibility of ILP, the proposed model can be extended to several other (secondary) objectives. To demonstrate this, an extension for minimal SCM and MCM circuits using 3-input adders as well as an extension for minimizing the circuits' glitch path count for low power applications are provided. The experimental results show that the formulation is useful for practically relevant problem sizes.**

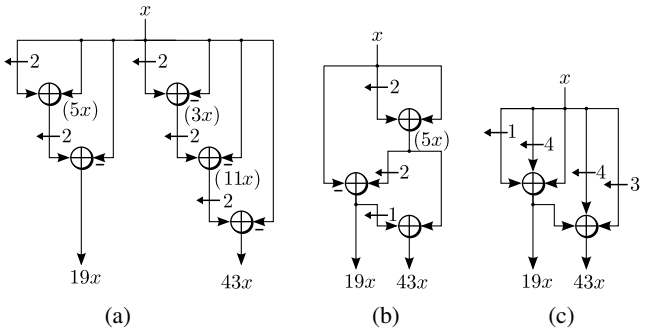

Fig. 1: Different adder circuits to realize a multiplication with constants 19 and 43 using (a) MSD, (b) minimal 2-input adders and (c) minimal 3-input adders

I. Introduction

The multiplication by a constant is an important operation that appears in many numerical algorithms. Commonly, the multiplication with a single constant is called single constant multiplication (SCM) while the multiplication with several constants is referred to as multiple constant multiplication (MCM). The MCM operation appears quite often in algorithms from digital signal processing, like finite impulse response (FIR) filters or discrete transforms. One common and efficient way for realizing SCM or MCM circuits is to transform them into several additions, subtractions and bit-shifts.

The conventional binary multiplication is based on the shifted addition of (digit-wise) partial products. When one operand is a constant, obviously, only the non-zero digits have to be considered. Using a signed digit representation of the constant [1] often reduces the number of non-zeros digits. However, it is well known that this typically does not lead to an optimal circuit, i.e., a circuit with minimal adders/subtractors. Fig. 1 shows examples of an MCM operation with the two constants 19 and 43. While the solution in Fig. 1(a) uses a minimum signed digit (MSD) representation, the solution in Fig. 1(b) shows the optimal circuit using common 2-input adders. Determining the optimal circuit for a given constant(s) is known as the SCM (MCM) problem. It was shown that even the simpler SCM problem belongs to the class of NP-complete optimization problems [2].

Optimal SCM circuits were exploited for up to four adders by the MAG algorithm of Dempster and Macleod [3]. Later, these results were extended by Gustafsson et al. to five adders [4]. Further extensions for up to six adders were considered in [5]. These works showed that using four, five and six adders are sufficient to multiply with 12 bit, 19 bit and 32 bit constants, respectively. However, to integrate results from these works into CAD tools, large look-up tables storing all enumerated solutions have to be used.

Solving the MCM problem in an optimal way received a lot of attention in the recent years. A first approach using integer linear programming (ILP) was proposed by Gustafsson [6]. The search space of the MCM problem was constructed as a directed hypergraph where each node corresponds to a possible factor and hyperarcs (=two input arcs) are used to represent ways to compute this factor from other factors. Then, the MCM problem is identical to the problem of finding a Steiner hypertree in that hypergraph. However, its computational complexity is only suitable for small MCM instances or to find lower bounds of the adder count by relaxing the model to a continuous LP problem. Similar ideas were used in [7], [8] to solve variants of the MCM problem, the pipelined MCM (PMCM) problem and MCM with minimum adder depth (AD). Another ILP-based approach was proposed by Aksoy et al. [9]. However, as it assumes a certain number representation, it is not globally optimal. The same group developed specialized breadth-first search (BFS) and depth-first search (DFS) algorithms for optimally solving the MCM problem [10], [11].

In all previous MCM ILP formulations [6]–[9], the search space has to be constructed in advance by tabulating all possible factors that may occur in an optimal solution. As the number of these factors grow exponentially with the coefficient word size, this quickly leads to very large and computationally intensive models. This work presents an alternative ILP formulation of the MCM problem that avoids any pre-computation of possible factors. This is achieved by optimizing the MCM

M. Kumm is with the Digital Technology Group, University of Kassel, 34121 Kassel, Germany (e-mail: kumm@uni-kassel.de).

circuit structure itself. Of course, this can not beat highly specialized branch-and-bound algorithms [10], [11] but is easy to re-implement and offers a framework for different main or secondary optimization objectives or additional constraints. Besides that, any progress in the performance of ILP solvers translates into the performance of solving the MCM problem. Examples for different objectives are minimizing power consumption [12]–[16], the combination with LUT-based multipliers [7] or additional constraints on fan-out [6], pipelining [8], 3-input adders [17], [18], carry-save adders [19] or reconfiguration [20]. Moreover, it might be integrated into related ILP formulations, e.g., for the optimization of digital filters [21]. To show the versatility of the proposed ILP model, an extension for 3-input adders as well as an extension for minimizing the GPC are proposed. In the following, the ILP formulation is introduced.

II. ILP Formulation

In the proposed ILP formulation, we assume that the number of adders N_A is known. Hence, to find the minimal number of adders, we have to start our search by solving the ILP for a lower bound of the adders $N_\mathrm{A} = N_\mathrm{A,lb}$ and, if infeasible, increase N_A by one until a feasible solution is found. As a consequence, the objective for the ILP solver is not used (set to a constant) but can be used for other secondary objectives. There are well-known lower bounds for the adder count available [22], [23].

We first start with a non-linear mathematical model of the SCM problem to highlight the main ideas and the definition of the main variables. These non-linear constraints are next linearized and then, the formulation is extended to MCM.

A. A First Non-Linear Model

SCM or MCM instances like the ones shown in Fig. 1 are represented as a so-called *adder graph*. Each node in an adder graph corresponds to an adder/subtractor and is assigned to a constant, which is a multiple of the input. Each edge corresponds to a connection between adders or input/output(s) and is assigned to a bit shift. In the following, each adder node is labeled with an index $a = 1 \dots N_\mathrm{A}$. In addition, we define an input node $a = 0$ and its corresponding constant is constrained to be

$$\text{C1:} \qquad c_0 = 1 \ .$$

The constant of an adder node is computed from two other nodes by

$$c_a = (-1)^{\phi_{a,\ell}} 2^{s_{a,\ell}} c_{a,\ell} + (-1)^{\phi_{a,\mathrm{r}}} 2^{s_{a,\mathrm{r}}} c_{a,\mathrm{r}} \ , \qquad (1)$$

where $\phi_{a,\ell/\mathrm{r}}$, $s_{a,\ell/\mathrm{r}}$ and $c_{a,\ell/\mathrm{r}}$ correspond to the sign, the bit shift and the source node connected at the left (ℓ) and right (r) input of the adder, respectively.

Now, the SCM problem can be defined by finding a feasible solution of source nodes, shifts and signs in (1) for the given adder nodes with minimal N_A, such that one coefficient c_a is identical to the target constant C we want to multiply with. As (1) is non-linear, it can not be used in standard ILP solvers. For that, an indicator constraint model is derived in the following.

TABLE I: Used constants (top) and variables (bottom)

Constant/Variable	Meaning
$N_\mathrm{A} \in \mathbb{N}$	Number of adders
$N_\mathrm{O} \in \mathbb{N}$	Number of outputs (for MCM)
C, $C_j \in \mathbb{N}$	Target SCM constant, target MCM constant j
S_min, $S_\mathrm{max} \in \mathbb{Z}$	Minimum and maximum shift
$c_a \in \mathbb{N}$	Constant computed in adder a
$c_{a,i} \in \mathbb{N}$	Constant of input $i \in \{\ell, \mathrm{r}\}$ of adder a
$c^\mathrm{sh}_{a,i} \in \mathbb{N}$	Shifted constant of input $i \in \{\ell, \mathrm{r}\}$ of adder a
$c^\mathrm{sh,sg}_{a,i} \in \mathbb{N}$	Shifted, sign corrected constant of input $i \in \{\ell, \mathrm{r}\}$ of adder a
$\phi_{a,i} \in \{0,1\}$	Sign of input $i \in \{\ell, \mathrm{r}\}$ of adder a (0:'+', 1:'−')
$c_{a,i,k} \in \{0,1\}$	true, if input i of adder a is connected to adder k
$\varphi_{a,i,s} \in \{0,1\}$	true, if input i of adder a is shifted by s bits
$o_{a,j} \in \{0,1\}$	true, if output j is connected to adder a

B. A Model using Indicator Constraints

Indicator constraints are special constraints in which a binary variable controls whether or not a specified linear constraint is active. They are directly supported by many ILP solvers and are numerically more robust compared to so-called big-M constraints but are typically slower due to weaker relaxations [24]. We start with a formulation containing indicator constraints. First, the non-linear relation of (1) is split into several additional variables as follows

$$c_a = \underbrace{(-1)^{\phi_{a,\ell}} \underbrace{2^{s_{a,\ell}} c_{a,\ell}}_{=c^\mathrm{sh}_{a,\ell}}}_{=c^\mathrm{sh,sg}_{a,\ell}} + \underbrace{(-1)^{\phi_{a,\mathrm{r}}} \underbrace{2^{s_{a,\mathrm{r}}} c_{a,\mathrm{r}}}_{=c^\mathrm{sh}_{a,\mathrm{r}}}}_{=c^\mathrm{sh,sg}_{a,\mathrm{r}}} \ . \qquad (2)$$

The constant after the bit shift of the left (ℓ) and right (r) input, respectively, is represented by $c^\mathrm{sh}_{a,\ell/\mathrm{r}}$, while $c^\mathrm{sh,sg}_{a,\ell/\mathrm{r}}$ also include the sign. Now, the adder result can be linearly represented by

$$\text{C2:} \quad c_a = c^\mathrm{sh,sg}_{a,\ell} + c^\mathrm{sh,sg}_{a,\mathrm{r}} \quad \forall a = 1 \dots N_\mathrm{A} \ .$$

Next, several binary decision variables are used to select the sources, shifts and signs of each node. An overview of the constants and variables is given in Table I. All variables (including the ones explained next) describing the configuration of the adder computing $19x$ in Fig. 1(b) are shown in Fig. 2.

1) Adder Source: The binary variable $c_{a,i,k}$ is used to select the source. It is defined to be true, when input $i \in \{\ell, \mathrm{r}\}$ of adder a is connected to adder k. This can be represented by the following constraints:

$$\text{C3a:} \quad c_{a,i} = c_k \text{ if } c_{a,i,k} = 1 \quad \forall a = 1 \dots N_\mathrm{A}, i \in \{\ell, \mathrm{r}\}, k = 0 \dots a-1$$

$$\text{C3b:} \quad \sum_{k=1}^{a-1} c_{a,i,k} = 1 \quad \forall a = 1 \dots N_\mathrm{A}, i \in \{\ell, \mathrm{r}\}$$

While C3a is the indicator constraint that assigns the value c_k in case $c_{a,i,k}$ is set, C3b makes sure that exactly one source is selected for each adder input.

Instead of allowing arbitrary connections from any adder to any other adder, the adders are ordered and only connections from previous adders with a lower id (i.e., $k < a$) are allowed. This has two reasons. First, loops are prevented, i.e., factors which are computed from succeeding adders forming

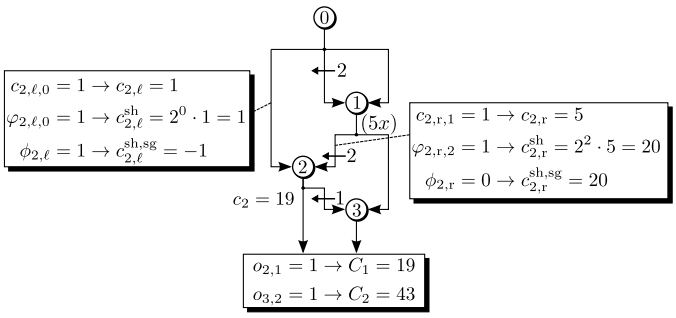

Fig. 2: Variables of adder 2 for the example in Fig. 1(b)

an invalid loop [6]. Second, the search space is reduced as many combinations having the same quality are excluded without loosing the optimal solution. This can be proven by the fact that nodes of an optimal adder graph can be always be reordered such that the node a with $a > k$ has a higher or equal depth than node k.

2) Shift of Input: The binary variable $\varphi_{a,i,s}$ is used to select the bit shift. It is defined to be true, when input $i \in \{\ell, \text{r}\}$ of adder a is shifted by s bits. The shifted constant is obtained by

$$\text{C4a:} \quad c^{\text{sh}}_{a,i} = 2^s c_{a,i} \text{ if } \varphi_{a,i,s} = 1 \quad \forall a = 1 \dots N_\text{A}, i \in \{\ell, \text{r}\}, \; s = S_\text{min} \dots S_\text{max}$$

$$\text{C4b:} \quad \sum_{s=S_\text{min}}^{S_\text{max}-1} \varphi_{a,i,s} = 1 \quad \forall a = 1 \dots N_\text{A}, i \in \{\ell, \text{r}\} \; .$$

Again, the shift is applied in indicator constraint C4a while C4b ensures that exactly one shift is selected. However, not all possible shift combinations are necessary. In general, it is sufficient that one input has a positive (left) shift while the other is zero or both inputs have an identical negative (right) shift [3]. As the inputs can be arbitrary connected in constraints C3a/b, we can further tighten the formulation by allowing only the right input shift to be non-zero positive and requiring both shifts to be identical for negative shifts

$$\text{C4c:} \quad \varphi_{a,\text{l},s} = 0 \quad \forall s > 0$$
$$\text{C4d:} \quad \varphi_{a,\ell,s} = \varphi_{a,\text{r},s} \quad \forall s < 0 \; .$$

3) Sign of Input: Another binary variable $\phi_{a,i}$ is defined for each adder input that determines the sign. The shifted and sign-corrected input of adder node a is obtained by

$$\text{C5a:} \quad c^{\text{sh,sg}}_{a,i} = -c^{\text{sh}}_{a,i} \text{ if } \phi_{a,i} = 1 \quad \forall a = 1 \dots N_\text{A}, i \in \{\ell, \text{r}\}$$
$$\text{C5b:} \quad c^{\text{sh,sg}}_{a,i} = c^{\text{sh}}_{a,i} \text{ if } \phi_{a,i} = 0 \quad \forall a = 1 \dots N_\text{A}, i \in \{\ell, \text{r}\}$$
$$\text{C5c:} \quad \phi_{a,\ell} + \phi_{a,\text{r}} \leq 1 \quad \forall a = 1 \dots N_\text{A} \; .$$

While C5a and C5b are obvious, constraints C5c ensure that only one input of the adder is subtracted (as subtracting both inputs is typically much more hardware demanding).

4) Output Constraints: For the SCM problem, the last adder has to be constrained to the target coefficient C

$$\text{C6:} \quad c_{N_\text{A}} = C \; .$$

To consider multiple outputs, we have to add new binary decision variables $o_{a,j}$ which determine the adder result a for each output j. This leads to the following constraints which replace C6 in the MCM case:

$$\text{C6a:} \quad C_j = c_a \text{ if } o_{a,j} = 1 \quad \forall a = 0 \dots N_\text{A}, j = 1 \dots N_\text{O}$$
$$\text{C6b:} \quad \sum_{j=1}^{N_\text{O}} o_{a,j} = 1 \quad \forall a = 0 \dots N_\text{A}$$

C. Fully Linear ILP Model

An indicator constraint of the form

$$x = y \text{ if } z = 1 \tag{3}$$

can be rewritten in linear form by using a big-M constraint

$$y - M + Mz \leq x \leq y + M - Mz \; . \tag{4}$$

For $z = 1$, (4) becomes $x = y$. For $z = 0$, (4) becomes

$$y - M \leq x \leq y + M \; . \tag{5}$$

Hence, if constant M is set to a sufficiently large value, the constraint is disabled.

Using this relation, the indicator constraints C3a, C4a, C5a, C5b and C6a can be linearized as follows:

$$\text{C3a':} \quad c_k - M + Mc_{a,i,k} \leq c_{a,i} \leq c_k + M - Mc_{a,i,k}$$
$$\text{C4a':} \quad 2^s c_{a,i} - M + M\varphi_{a,i,s} \leq c^{\text{sh}}_{a,i} \leq 2^s c_{a,i} + M - M\varphi_{a,i,s}$$
$$\text{C5a':} \quad c^{\text{sh}}_{a,i} - M\phi_{a,i} \leq c^{\text{sh,sg}}_{a,i} \leq c^{\text{sh}}_{a,i} + M\phi_{a,i}$$
$$\text{C5b':} \quad -c^{\text{sh}}_{a,i} - M + M\phi_{a,i} \leq c^{\text{sh,sg}}_{a,i} \leq -c^{\text{sh}}_{a,i} + M - M\phi_{a,i}$$
$$\text{C6a':} \quad c_a - M + Mo_{a,j} \leq C_j \leq c_a + M - Mo_{a,j}$$

For all Big-M constraints above, M must be larger than that maximum coefficient value. Note that the integer variables from Table I are computed from integer constants and booleans, so they are actually integers and can be relaxed to real numbers to speed up the optimization. However, for large constants, this may yield to numerical problems within the solver which have to be considered.

As we now have a framework for SCM and MCM, we will now look at two specific extensions for 3-input adders and for minimum GPC to show the versatility of ILP.

III. Ternary Adder Extension

Modern FPGAs provide the possibility to implement 3-input adders, commonly referred to as ternary adders. These are capable of adding three independent inputs with the same logic resources than used for common two-input adders [25], [26]. The use of ternary adders also showed significant resource reductions for SCM [18] and MCM [17] operations. To give an example, Fig. 1(c) shows the optimal solution of the previous MCM example when using ternary adders. So far, no method for optimally solving MCM with ternary adders exists but the proposed ILP formulation is easily extended as follows.

Constraints C2 and C5c have to be replaced by

$$\text{C2}^*\text{:} \quad c_a = c^{\text{sh,sg}}_{a,\ell} + c^{\text{sh,sg}}_{a,\text{m}} + c^{\text{sh,sg}}_{a,\text{r}} \quad \forall a = 1 \dots N_\text{A}$$
$$\text{C5c}^*\text{:} \quad \phi_{a,\ell} + \phi_{a,\text{m}} + \phi_{a,\text{r}} \leq 2 \quad \forall a = 1 \dots N_\text{A} \; .$$

where the third input is represented as additional (middle) input $i = \text{m}$. While the extension in C2* is obvious, C5c*

limits up to two of the three inputs to being negative, which is realizable on modern FPGAs [17]. Constraints C3a/b, C4a/b and C5a/b have to be extended to include the case $i = \mathrm{m}$, i. e., $i \in \{\ell, \mathrm{m}, \mathrm{r}\}$. Constraints C1, C4c/d as well as C6 (SCM) or C6a/b (MCM) remain identical.

IV. Glitch Path Count Extension

There exist several metrics which give an indirect measure of the power consumption of shift-and-add based constant multiplication circuits. The simplest word level power metric is the adder depth (AD) (also called logic depth) of an MCM circuit. It is defined as the maximum number of cascaded adders on each path. Several methods exist which optimize the adder depth as a secondary objective [8], [13]–[15]. The glitch path (GP) count (GPC) was introduced as a more accurate word level power estimation [12] which was used in several succeeding works [8], [14], [15], [27]. The main idea is based on the observation that each adder produces glitches due to the internal carry propagation. Each glitch may now produce further glitches in succeeding adders. For circuits having the same adder cost, there may be substantial differences in the glitch propagation. As more glitches directly lead to more dynamic power, glitch-reduced adder graphs are preferred. The error of the GPC estimation compared to the gate-level power simulation Prime Power was shown to be in the order of 20% [16]. However, it was shown in several works that an MCM circuit with lower GPC typically provides a lower power consumption even with identical AD [12], [14], [15].

A. Definition of Glitch Path Count

The GP is defined for each adder output as

$$\mathrm{GP}_{\text{output}} = \mathrm{GP}_{\text{left input}} + \mathrm{GP}_{\text{right input}} + 1 \ , \tag{6}$$

where the GP of the adder graph input is defined to be zero. The sum of all GP's leads to the GPC of the adder circuit.

B. ILP Extension to Minimizing Glitch Path Count

Let GP_a be the GP of adder a. Per definition, the GP of the input ($a = 0$) is constrained to be zero

$$\text{C7a:} \qquad \mathrm{GP}_0 = 0 \ .$$

Using the already defined source decision variable $c_{a,i,k}$, the GP of the input is constrained to as

$$\text{C7b:} \quad \mathrm{GP}_{a,i} = \mathrm{GP}_k \text{ if } c_{a,i,k} = 1 \quad \forall a = 1 \dots N_\mathrm{A}, i \in \{\ell, \mathrm{r}\}, \ k = 0 \dots a-1$$

using indicator constraints or by

$$\text{C7b':} \quad \mathrm{GP}_k - M + M c_{a,i,k} \leq \mathrm{GP}_{a,i} \leq \mathrm{GP}_k + M - M c_{a,i,k}$$

using a linear model. The GP of adder a is now

$$\text{C7c:} \qquad \mathrm{GP}_a = \mathrm{GP}_{a,\ell} + \mathrm{GP}_{a,\mathrm{r}} + 1 \quad \forall a = 1 \dots N_\mathrm{A}$$

and the total GPC is the sum over all nodes, which is used as the objective for the ILP

$$\text{minimize} \ \sum_{a=1}^{N_\mathrm{A}} \mathrm{GP}_a \ .$$

As we have an outer loop testing for the minimal N_A, the GPC is in fact the secondary objective. This is desirable not only for the reason of the reduced area but also for power as the GPC does not consider the static power contribution of additional adders. However, if static logic is low compared to dynamic power, one could check larger N_A for lower GPC. The extension of the GPC objective to ternary adders is also straight forward.

V. Experimental Results

The implementation of the proposed formulation was done with the help of the Scalable Linear Programming library ScaLP [28] and is available as the `omcm` tool within the open-source framework PAGSuite [29]. Several experiments were performed to show the practical use of the ILP formulation. In all experiments, gurobi 7.0.2 [30] was used as ILP solver running on four cores of an Intel Xeon E5-2650 2.3 GHz CPU.

To get a first impression of the complexity, we consider the 15th order linear-phase lowpass FIR filter used in [6] with the coefficients $\{-8, 4, 28, 5, -67, -44, 175, 422\}/1024$. It was reported in [6] that only the linear relaxation of the problem was solved in which all integral constraints on the variables are relaxed to continuous variables which significantly reduces the problem complexity. The linear relaxation objective was found to be 6.12, which tells that the lower bound is seven adders and proves the optimality of solutions of several previous heuristics. The proposed method is able to prove that six adders is not sufficient after 55 seconds while the optimal solution using seven adders was found in less than a second.

To test the ternary adder extension, the same filter benchmark as used in [17] was optimized. For the smaller designs with $N = 6$, $N = 10$ and $N = 13$ coefficients, the heuristic results obtained in [17] could be proven to be optimal. For the larger designs no solution could be found within 24 hours. This shows the efficiency but also the necessity of the heuristic.

Additional experiments were performed with an FIR filter benchmark from image processing that was previously used in [7], [8]. As a baseline, one of the most advanced MCM heuristic H_{cub} [31], the optimal MCM method DFS [11] as well as the minimum adder depth MCM heuristic mode of RPAG [8] were used. Table II shows for each filter instance, the size of the constant matrix, the bit width of the coefficients B_c, the number of unique odd coefficients (excluding 0 and 1) N_{uq} and, for each method, the resulting adder cost (#add), the average adder depth (AD) and the GPC. In addition, the runtime is given for the proposed ILP method, for all other methods, the runtime was always less than a second. Regarding the adder cost, the H_{cub} heuristic performs remarkably good as only one instance (gaussian 5×5) requires one adder more than the optimum. Of course, the specialized DFS search is able to find optimal results for every instance but is not simply extendable towards minimum GPC. The proposed approach was able to find a minimum adder solution except the lowpass filter 15×15, for which no solution could be found within 24 hours. Regarding the GPC, it can be observed that solutions with lower GPC can be found having the same optimal adder cost. Due to the increased complexity, for two

TABLE II: Optimization results for minimum adder cost (#add) and minimum GPC using different methods (best marked bold)

				H_{cub} [31]			DFS [11]			RPAG min AD [8]			proposed			
Filter	Size	B_c	N_{uq}	#add	AD (avg.)	GPC	#add	AD (avg.)	GPC	#add	AD (avg.)	GPC	#add	AD (avg.)	GPC	runtime [sec]
gaussian	3×3	8	3	**4**	**1.5**	7	**4**	**1.5**	7	**4**	**1.5**	**6**	**4**	**1.5**	**6**	0.7
gaussian	5×5	12	3	6	**2**	14	**5**	2.2	**17**	6	**2**	14	**5**	2.2	**17**	12.8
laplacian	3×3	8	3	**3**	1.67	**6**	**3**	1.67	**6**	4	**1.5**	7	**3**	1.67	**6**	0.02
unsharp	3×3	8	3	**4**	**1.5**	**6**	**4**	**1.5**	**6**	**4**	**1.5**	**6**	**4**	**1.5**	**6**	0.24
unsharp	3×3	12	3	**5**	2.2	16	**5**	2	15	6	**1.83**	15	**5**	2.2	**14**	14.5
lowpass	5×5	8	5	**6**	1.67	16	**6**	1.67	**13**	7	**1.57**	13	**6**	1.67	**13**	150.8
lowpass	9×9	10	12	**12**	1.58	**24**	**12**	1.58	28	13	**1.54**	23	**12**	–	–	12859
lowpass	15×15	12	25	**25**	**2.04**	**67**	**25**	**2.04**	69	27	**1.96**	70	–	–	–	–
highpass	5×5	8	4	**4**	**1.25**	**5**	**4**	**1.25**	**5**	**4**	**1.25**	**5**	**4**	**1.25**	**5**	0.13
highpass	9×9	10	5	**5**	**1.4**	**7**	**5**	**1.4**	**7**	**5**	**1.4**	**7**	**5**	**1.4**	**7**	0.45
highpass	15×15	12	12	**12**	**1.5**	**18**	**12**	**1.5**	19	**12**	**1.5**	**18**	**12**	–	–	305.7

more instances, the solver failed in finding a solution within 24 hours. One interesting observation is that the minimum adder depth solutions obtained by RPAG typically not lead to the minimum GPC solutions (in special, every time when the adder count was increased).

VI. Conclusion

A completely different way of formulating the SCM and MCM problems using ILP was proposed in this work. No precomputation of possible constants is required which yields to very compact models with only few variables. It is easily extendable towards different objectives or additional constraints. It was shown that problem sizes of practical interest can be solved. Due to the NP-complete nature of the considered problem and ILP itself, not every SCM or MCM problem can be solved. However, any improvement on future ILP solvers will help to get closer to achieving the goal of optimizing MCM related problems in a closed mathematical framework.

References

[1] A. Avizienis, "Signed-Digit Number Representations for Fast Parallel Arithmetic," *IEEE Trans. on Elec. Comput.*, vol. EC-10, no. 3, pp. 389–400, 1961.

[2] P. Cappello and K. Steiglitz, "Some Complexity Issues in Digital Signal Processing," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 32, no. 5, pp. 1037–1041, Oct. 1984.

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

[4] O. Gustafsson, A. Dempster, K. Johansson, M. Macleod, and L. Wanhammar, "Simplified Design of Constant Coefficient Multipliers," *Circuits, Systems, and Signal Processing*, vol. 25, no. 2, pp. 225–251, 2006.

[5] 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, Sep. 2011.

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

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

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

[9] L. Aksoy, E. da Costa, P. Flores, and J. Monteiro, "Exact and Approximate Algorithms for the Optimization of Area and Delay in Multiple Constant Multiplications," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 27, no. 6, pp. 1013–1026, 2008.

[10] L. Aksoy, E. Gunes, and P. Flores, "An Exact Breadth-First Search Algorithm for the Multiple Constant Multiplications Problem," *NORCHIP*, pp. 41–46, 2008.

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

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

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

[14] K. Johansson, "Low Power and Low Complexity Shift-and-Add Based Computations," Ph.D. dissertation, Linköping University, 2008.

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

[16] W. B. Ye and Y. J. Yu, "Switching activity analysis and power estimation for multiple constant multiplier block of FIR filters," *IEEE Int. Symp. on Circuits and Systems (ISCAS)*, pp. 145–148, 2011.

[17] M. Kumm, M. Hardieck, J. Willkomm, P. Zipf, and U. Meyer-Baese, "Multiple Constant Multiplication with Ternary Adders," in *IEEE Int. Conf. on Field Program. Logic and Appl. (FPL)*, 2013, pp. 1–8.

[18] M. Kumm, O. Gustafsson, M. Garrido, and P. Zipf, "Optimal Single Constant Multiplication using Ternary Adders," *IEEE Trans. Circuits Syst. II*, vol. accepted for publication, 2016.

[19] O. Gustafsson and L. Wanhammar, "Low-Complexity and High-Speed Constant Multiplications for Digital Filters Using Carry-Save Arithmetic," in *Digital Filters*. InTech, Apr. 2011.

[20] K. Möller, M. Kumm, M. Kleinlein, and P. Zipf, "Reconfigurable Constant Multiplication for FPGAs," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 36, no. 6, pp. 927–937, 2017.

[21] D. Shi and Y. J. Yu, "Design of Linear Phase FIR Filters With High Probability of Achieving Minimum Number of Adders," *IEEE Trans. Circuits Syst. I*, vol. 58, no. 1, pp. 126–136, 2011.

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

[23] M. G. C. Jimenez, U. M. Baese, and G. J. Dolecek, "Theoretical Lower Bounds for Parallel Pipelined Shift-and-Add Constant Multiplications with N-Input Arithmetic Operators," *EURASIP Journal on Advances in Signal Processing*, vol. 2017, no. 1, pp. 1–13, May 2017.

[24] E. Klotz and A. M. Newman, "Practical guidelines for solving difficult mixed integer linear programs," *Surveys in Operations Research and Management Science*, vol. 18, no. 1-2, pp. 18–32, Oct. 2013.

[25] J. M. Simkins and B. D. Philofsky, "Structures and Methods for Implementing Ternary Adders/Subtractors in Programmable Logic Devices," *US Patent No 7274211, Xilinx Inc.*, Sep. 2007.

[26] G. Baeckler, M. Langhammer, J. Schleicher, and R. Yuan, "Logic Cell Supporting Addition of Three Binary Words," *US Patent No 7565388, Altera Coop.*, 2009.

[27] M. Faust, "Design Methodologies for Complexity Reduction of FIR Filters," Ph.D. dissertation, NTU Singapore, 2014.

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

[29] M. Kumm, K. Möller, M. Hardieck, and P. Zipf. (2018) PAGSuite project website. [Online]. Available: http://www.uni-kassel.de/go/pagsuite

[30] Gurobi Inc., "Gurobi Optimizer Reference Manual," Tech. Rep., 2016.

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