Approximation of Multiple Constant Multiplications Using Minimum Look-Up Tables on FPGA

Levent Aksoy[†], Paulo Flores[†‡] and José Monteiro[†‡]

[†] INESC-ID, [‡] Instituto Superior Técnico, Universidade de Lisboa

Lisbon, Portugal

Abstract— **In many digital signal processing (DSP) systems, computations can be carried out within a tolerable error range rather than finding the exact output, enabling significant reductions in area, delay, or power dissipation of the design. This paper addresses the problem of approximating the multiple constant multiplications (MCM) operation which frequently occurs in DSP applications. We consider the realization of constant multiplications using look-up tables (LUTs) on field programmable gate arrays (FPGA) and introduce an exact algorithm, called THETIS, that can find a minimum number of distinct LUTs required to realize the partial products of constant multiplications, satisfying an error constraint. Experimental results show that THETIS can achieve significant reductions in number of LUTs on MCM instances and its solutions lead to less complex filter designs on FPGA than those realized using original filter coefficients.**

I. INTRODUCTION

The MCM operation realizes the multiplication of multiple constants by an input variable and appears in many DSP systems such as fast Fourier transforms, finite impulse response (FIR) filters (Fig. 1), and linear DSP transforms. Since the implementation of a multiplier in hardware is expensive in terms of area, delay, and power dissipation, efficient multiplierless constant multiplication design architectures have been introduced [1], [2]. For the multiplierless design of a single constant multiplication (SCM) on FPGA, the LUT-based multipliers (LBMs), which take into account the basic block of an FPGA, was proposed in [2], [3]. In this technique, supposing n-bit LUTs are used, the input variable is split into n-bit segments, the result of a constant multiplication by an n-bit segment (partial product) is stored in n-bit LUTs, and the partial products are shifted and added to compute the result. Note that shifts by a constant value can be realized using only wires which represent no hardware cost. This technique has been used in Altera and Xilinx FPGAs [4], [5]. In [6], LBMs were used for the multiplierless realization of the MCM block, where common LUTs are shared among the partial products of constant multiplications. It was shown in [6] that FIR filters designed using LBMs include similar number of slices and have significantly less delay with respect to those designed under a shift-adds architecture [1], where constant multiplications are replaced with shifts and adders/subtractors.

Approximate computing refers to a class of methods that relax the requirement of exact equivalence between the specification and implementation of a computing system [7]. For example, in many image and video processing applications, due to the limited perceptual abilities of human beings, the accuracy of numerical outputs can be traded for reductions in area, delay, or power dissipation of the design [8].
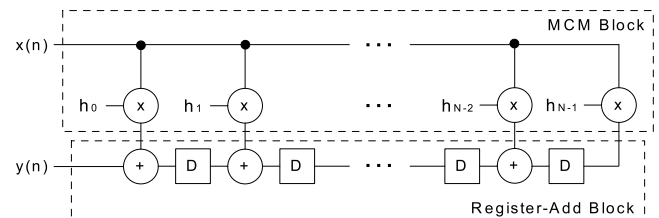

Fig. 1. Transposed form FIR filter.

The approximation of the MCM operation, which targets the shift-adds architecture and aims to reduce the number of adders/subtractors, was proposed in [9], [10]. In this paper, we focus on the approximation of the MCM operation realized using LBMs with a minimum number of LUTs on FPGA. In this problem, given an error constraint and a set of alternative constants determined for each constant multiplication, the aim is to select a constant from each set such that the chosen constants satisfy the error constraint and the chosen constant multiplications require a minimum number of distinct LUTs. To the best of our knowledge, there exists no algorithm proposed for this problem. To this end, we formulate this problem as a 0-1 integer linear programming (ILP) problem and introduce an exact algorithm THETIS. Moreover, we apply THETIS to the filter design optimization (FDO) problem under a tolerable error [11]. Experimental results show that THETIS can be applied to MCM instances which include a large number of coefficients with large bit-widths, finding a minimum solution in a reasonable time. Also, filters designed based on the optimized solutions of THETIS occupy significantly fewer slices than those implemented using original filter coefficients.

II. BACKGROUND

A. 0-1 ILP Problem

The 0-1 ILP problem is the optimization of a linear objective function subject to a set of linear constraints and is generally defined as follows[1]:

$$\text{minimize} \quad \mathbf{w}^T \cdot \mathbf{x} \tag{1}$$

$$\text{subject to} \quad \mathbf{A} \cdot \mathbf{x} \geq \mathbf{b}, \quad \mathbf{x} \in \{0,1\}^k \tag{2}$$

In the objective function of the 0-1 ILP problem given in Eq. 1, w_i in $\mathbf{w}$ is a weight value associated with each variable x_i, where $1 \leq i \leq k$ and $\mathbf{w} \in \mathbb{Z}^k$. In Eq. 2, $\mathbf{A} \cdot \mathbf{x} \geq \mathbf{b}$ denotes a set of j linear constraints, where $\mathbf{b} \in \mathbb{Z}^j$ and $\mathbf{A} \in \mathbb{Z}^j \times \mathbb{Z}^k$.

---

[1]The minimization objective can be converted to a maximization objective by negating the objective function. Less-than-or-equal and equality constraints are respectively accommodated by the equivalences, $\mathbf{A} \cdot \mathbf{x} \leq \mathbf{b} \Leftrightarrow -\mathbf{A} \cdot \mathbf{x} \geq -\mathbf{b}$ and $\mathbf{A} \cdot \mathbf{x} = \mathbf{b} \Leftrightarrow (\mathbf{A} \cdot \mathbf{x} \geq \mathbf{b}) \wedge (\mathbf{A} \cdot \mathbf{x} \leq \mathbf{b})$.
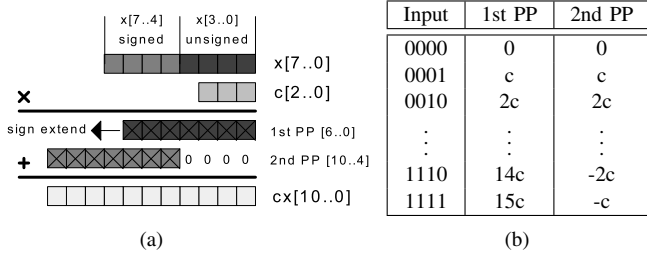
Fig. 2. (a) Realization of *cx* when *n* is 4; (b) the partial products of *cx*.

Fig. 3. The network generated for **c**=[11,25] when ε is 2 and *n* is 4.

B. LUT Based Multiplier

Suppose that a signed 8-bit input variable *x* is multiplied by a 3-bit constant *c* when the size of LUTs (*n*) is 4. Fig. 2a shows the way of realizing *cx* and Fig. 2b presents the all possible values of partial products (PPs). Note that for a signed input variable, only the most significant *n*-bit segment is treated as a signed input and for an unsigned input variable, every *n*-bit segment is an unsigned input.

In LBM, each output bit of a partial product is stored in a LUT, except the ones that consist of all 0s, all 1s, or that are equal to an input bit. Note that an output bit of a partial product may be equal to another output bit of the same partial product or as commonly observed, equal to an output bit of a partial product generated for a different constant multiplication [6]. In this case, this LUT can be shared among the partial products.

C. Problem Definitions

The problem of approximating the MCM operation can be defined as: given a vector of *N* original integer constants **c**=[$c_0, \cdots, c_{N-1}$], a tolerable error $\varepsilon > 0$, and a vector error norm, *e.g.*, 1-norm[2], find a vector of *N* optimized constants **c′**=[$c'_0, \cdots, c'_{N-1}$] such that their multiplications by a variable using LBMs require a minimum number of LUTs and the error constraint, *e.g.*, $||\mathbf{c} - \mathbf{c'}||_1 \le \varepsilon$, is satisfied.

The frequency response of an FIR filter is computed as:

$$H(w) = \sum_{i=0}^{N-1} h_i e^{-jwi}$$

where *N* is the filter length, **h**=[$h_0, \cdots, h_{N-1}$] $\in \mathbb{R}^N$ is a vector of floating-point filter coefficients, and $w \in \mathbb{R}$ is the frequency in radians. The absolute error in the frequency response is bounded by the 1-norm of the coefficient vector error regardless of the frequency [11], given as:

$$\left| |H(w)| - |H'(w)| \right| \le \left| \sum_{i=0}^{N-1} (h_i - h'_i) e^{-jwi} \right| \le \sum_{i=0}^{N-1} |h_i - h'_i| = ||\mathbf{h} - \mathbf{h'}||_1$$

where **h′** denotes a vector of optimized coefficients. Thus, the FDO problem under a tolerable error can be defined as: given the vector of original coefficients **h** and a tolerable error $\varepsilon > 0$, find the vector of optimized coefficients **h′** that yields a filter design (Fig. 1) whose MCM block is realized using LBMs with a minimum number of LUTs, satisfying $||\mathbf{h} - \mathbf{h'}||_1 \le \varepsilon$.

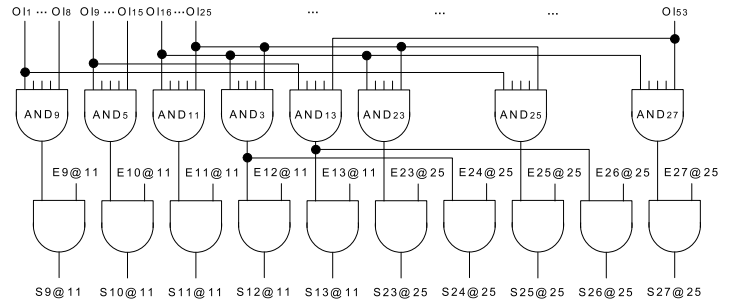[2]1-norm of a vector **y** is computed as $||\mathbf{y}||_1 = \sum_{i=0}^{N-1} |y_i|$.

III. THETIS: AN EXACT ALGORITHM

This section presents THETIS assuming that the variable, by which the constants are multiplied, is a signed input (Section II-B) and the 1-norm is used in the error constraint (Section II-C). THETIS has four main parts which are described through an example, where **c**=[11,25], ε is 2, and *n* is 4.

In a preprocessing phase, each constant of **c**, whose absolute value is not 0 and a power of 2, is moved to the target set *T*, since these constant multiplications need LBMs to be realized.

For our example, the target set *T* consists of 11 and 25.

In the first part, initially, for each target constant $t_i \in T$, we generate a set R^{t_i} consisting of constants between $t_i - \varepsilon$ and $t_i + \varepsilon$ which are alternative constants for the realization of $t_i x$ satisfying the error constraint. Then, for each constant in each set R^{t_i}, $r_j^{t_i}$, we convert it to a positive and odd constant and if it is not 0 or 1, we add it to the set *P* without repetition. The set *P* includes all distinct constants required to realize the constant multiplications in the set R^{t_i} of each target constant. Note also that the multiplication of a negative constant can be obtained by negating the related output of the MCM block. For each constant in *P*, we determine the LUTs required for the implementation of its partial products. For each LUT, which is not equal to an input bit or to an array with all 0s or 1s, we check if it is already included in a set of $2^n \times 1$ arrays called *L*. If not, it is added to *L*. The set *L* includes all distinct LUTs required to realize the constant multiplications in the set R^{t_i} of each target constant, satisfying the error constraint.

For our example, $R^{11}=\{9, 10, 11, 12, 13\}$, $R^{25}=\{23, 24, 25, 26, 27\}$, $P=\{3, 5, 9, 11, 13, 23, 25, 27\}$, and *L* consists of 53 distinct LUTs. For the partial products of each constant in *P*, 6, 7, 8, 10, 9, 12, 10, and 12 LUTs are required, respectively. We note that 21 LUTs out of 53 appear more than once in the partial products of constants in *P*, which will be shared.

In the second part, the realizations of constant multiplications are represented in a Boolean network. Initially, each array of the set *L* is represented as an optimization variable Ol_m. Then, for each constant in *P*, p_k, an AND gate AND_{p_k}, which combines all the LUTs required for its partial products, is generated. For each target constant t_i and for each constant in R^{t_i}, $r_j^{t_i}$, where $|r_j^{t_i}|$ is different from 0 and a power of 2, we generate two variables $Er_j^{t_i}@t_i$ and $Sr_j^{t_i}@t_i$, which respectively denote the *error* and *selection* variables, and generate a 2-input AND gate whose output is $Sr_j^{t_i}@t_i$, first input is $Er_j^{t_i}@t_i$, and second input is the output of AND_{p_k}, where p_k is the positive
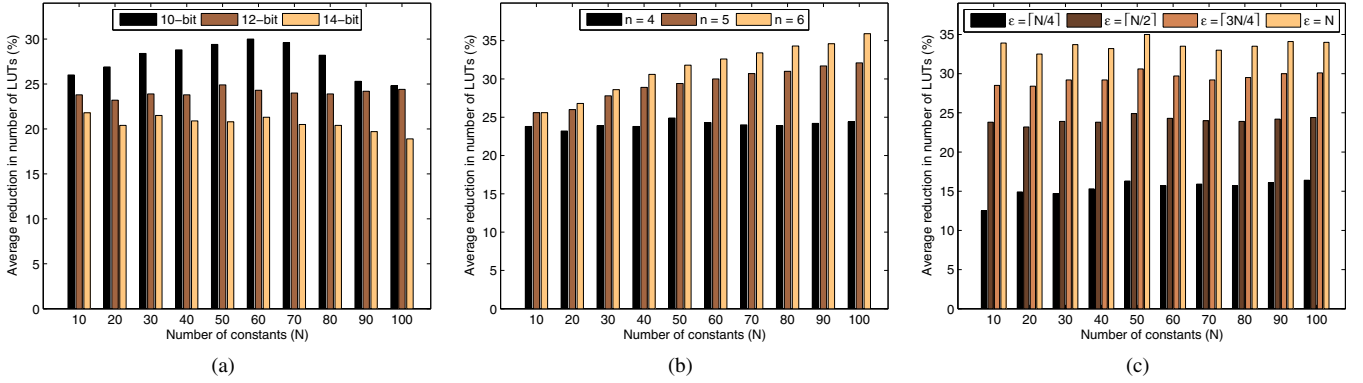
Fig. 4. Average reduction in number of LUTs: (a) with different sizes of constants; (b) with different sizes of LUTs; (c) with different values of ε.

and odd version of $r_j^{t_i}$. If $|r_j^{t_i}|$ is 0 or a power of 2, the same variables and 2-input AND gate are also generated, but the second input of this AND gate is set to 1, indicating that $Er_j^{t_i}@t_i$ and $Sr_j^{t_i}@t_i$ are equal to each other.

For our example, the network generated for the realization of constant multiplications is given in Fig. 3, but only showing two LUTs for each constant in P for the sake of clarity.

In the third part, the 0-1 ILP problem is generated. The cost function is a linear function of optimization variables Ol_m with weight values set to 1. To obtain the constraints of the 0-1 ILP problem: i) the conjunctive normal form (CNF) of each gate in the network is found and each CNF clause is converted to a linear inequality [12]; ii) the *error* constraint is generated as $\sum_{i=0}^{|T|-1} \sum_{j=0}^{|R^{t_i}|-1} |t_i - r_j^{t_i}| \cdot Er_j^{t_i}@t_i \le \varepsilon$; iii) for each target constant t_i, a *selection* constraint is generated as $\sum_{j=0}^{|R^{t_i}|-1} Sr_j^{t_i}@t_i = 1$, indicating that only one element must be selected from R^{t_i}.

In the forth part, the 0-1 ILP problem is solved using a generic 0-1 ILP solver. The constants to be multiplied by the input variable are determined as the ones whose variable $Sr_j^{t_i}@t_i$ is set to 1 in the solution of the 0-1 ILP solver.

For our example, 19 distinct 4-bit LUTs are needed for the generation of partial products of original constants 11 and 25. THETIS finds a solution with $12 = 3 \ll 2$ and $24 = 3 \ll 3$, which requires a total of 6 distinct 4-bit LUTs, satisfying the error constraint and achieving a 68% reduction in number of LUTs.

IV. EXPERIMENTAL RESULTS

This section gives the results of THETIS on MCM and filter instances. THETIS was written in MATLAB, run on a PC with Intel Core i5-2410M at 2.3GHz, and equipped with the 0-1 ILP solver SCIP2.0 [13]. The results of filter designs implemented based on the solutions of THETIS on FPGAs are also given. FIR filters were described in VHDL and synthesized using the Xilinx ISE Design Suite 13.1 on the Virtex 6 xc6vlx75T-2ff484 target device. In the synthesis script, relaxed timing constraints were used in order to provide more freedom to the tool to optimize area. The functionality of filters was verified on 10,000 randomly generated input signals in simulation, from which the switching activity information used by the tool to compute the power dissipation was also obtained.

To explore the impact of bitwidth of constants and values of n and ε on the results of THETIS, as the first experiment set, we

TABLE I

SPECIFICATIONS OF FIR FILTERS.

Filter	filter length	normalized passband	normalized stopband	passband ripple	stopband ripple
1	47	0.05	0.12	0.060	0.060
2	73	0.10	0.16	0.030	0.030
3	105	0.20	0.24	0.010	0.010

used 10-, 12-, and 14-bit randomly generated MCM instances, where the number of constants (N) ranges from 10 to 100 in steps of 10. For each group, there were 30 MCM instances. Fig. 4 shows average reduction in number of LUTs computed between the results of THETIS and those found for the original constants when a signed input is considered. Fig. 4a presents the results on different sizes of constants when ε is $\lceil N/2 \rceil$ and n is 4. Fig. 4b gives the results on different sizes of LUTs using 12-bit constants when ε is $\lceil N/2 \rceil$. Fig. 4c shows results on different values of ε using 12-bit constants when n is 4.

Observe from Fig. 4a that as the size of constants increases, the reduction in number of LUTs decreases due to the fact that the number of distinct positive and odd constants, whose multiplications are considered, increases in this case, decreasing the sharing of LUTs. The maximum (minimum) value of the average reduction is 30 (18.9)% on 10 (14)-bit constants when N is 60 (100). Observe from Fig. 4b that as the size of LUTs increases, the reduction in number of LUTs increases, which is simply because the size of partial products of constant multiplications, and consequently, the number of LUTs to be considered, increases in this case, increasing the sharing of LUTs. The maximum (minimum) value of the average reduction is 35.9 (23.2)% when n is 6 (4) and N is 100 (60). Observe from Fig. 4c that as the value of ε increases, the reduction in number of LUTs increases, since more alternative constants are considered in this case. The maximum (minimum) value of the average reduction is 35 (12.5)% when ε is N ($\lceil N/4 \rceil$) and N is 50 (10).

To explore the impact of solutions of THETIS on designs synthesized on FPGA, as the second experiment, we used three symmetric FIR filters whose specifications are given in Table I. On these FIR filters, we applied THETIS to the FDO problem under a tolerable error when ε was 0.001, 0.002, and 0.005. To do so, given the filter specifications, first, a vector of floating-point coefficients $\mathbf{h}$, which respects the filter characteristics, was found using a linear programming tool.

TABLE II

Summary of results of thetis on FIR filter instances.

Filter	ε	Q	Original filter						Optimized filter						CPU	Reduction (%)		
			luts	FFs	LUTs	slices	delay	power	luts	FFs	LUTs	slices	delay	power	(s)	luts	LUTs	slices
1	0.001	14	297	1143	1798	487	8.7	1414	231	1132	1579	427	8.6	1406	11.1	22.2	12.2	12.3
	0.002	13	248	1082	1552	444	8.4	1404	179	1069	1368	407	8.3	1403	5.8	27.8	11.9	8.3
	0.005	12	208	1045	1414	417	8.1	1400	127	1033	1222	342	8.1	1397	3.7	38.9	13.6	18.0
2	0.001	15	407	1842	2756	719	8.3	1424	301	1792	2512	667	8.4	1423	49.5	26.0	8.9	7.2
	0.002	14	356	1766	2550	685	8.4	1420	242	1720	2249	605	8.5	1416	27.0	32.0	11.8	11.7
	0.005	12	229	1610	2045	574	8.0	1401	146	1599	1856	520	8.2	1400	9.7	36.2	9.2	9.4
3	0.001	15	474	2668	4188	1084	8.9	1447	342	2637	3442	913	9.2	1440	61.4	27.8	17.8	15.8
	0.002	14	398	2555	3584	959	8.1	1418	265	2503	3067	845	8.7	1411	61.0	33.4	14.4	11.9
	0.005	13	305	2459	3177	858	8.3	1411	160	2391	2662	711	8.5	1406	23.9	47.5	16.2	17.1

Second, the minimum quantization (Q) value, that is used to convert the floating-point coefficients to integers was found taking into account the error constraint. Note that the use of a minimum Q value yields reduction on the size of filter coefficients and on the size of registers, adders, and multipliers in the filter design (Fig. 1). Thus, the vector of original integer coefficients were determined as $round(\mathbf{h} \cdot 2^Q)$ and in thetis, the alternative integer constants for each coefficient h_i ranges between $round((h_i - \varepsilon) \cdot 2^Q)$ and $round((h_i + \varepsilon) \cdot 2^Q)$.

Table II presents the results of filter designs when the bitwidth of the signed filter input was 12. In this table, *luts* denotes the number of distinct 6-input LUTs obtained based on the original and optimized coefficients in the MCM block of the filter. Also, *FFs*, *LUTs*, and *slices* stand for the number of flip-flops, LUTs, and slices used in the design of FIR filters on the target FPGA device, respectively and *delay* and *power* are the delay in the critical path in *ns* and total power dissipation in *mW*, respectively. Finally, *CPU* is the CPU time of thetis in seconds required to find the vector of optimized coefficients.

Observe from Table II that the solutions of thetis need significantly fewer LUTs than original filter coefficients for the realization of partial products in the MCM block. Hence, the FIR filters designed based on the solutions of thetis require fewer LUTs and slices than those designed using original filter coefficients. As can be seen from the frequency responses generated for Filter 3 when ε is 0.005 in Fig. 5 and Table II, a reduction of 17.1% in number of slices in the filter design can be achieved under slight tolerable errors. Also, the filters designed using optimized coefficients require fewer flip-flops than those designed based on original coefficients, since the sizes of coefficients can be reduced by thetis. However, the original and optimized filter designs have similar delay and power dissipation values. Observe that thetis can find a minimum solution in a reasonable time. Note that similar results given in Table II were obtained when 4- and 5-bit LUTs were considered and Virtex 4 and 5 FPGA devices were used.

V. Conclusions

This paper addressed the problem of approximating the MCM operation realized using LBMs on FPGA and introduced an exact algorithm thetis which can find a minimum number of distinct LUTs required to generate the partial products of constant multiplications. Experimental results showed that thetis can find significant reductions in number of LUTs in the MCM operation. Furthermore, the application of thetis
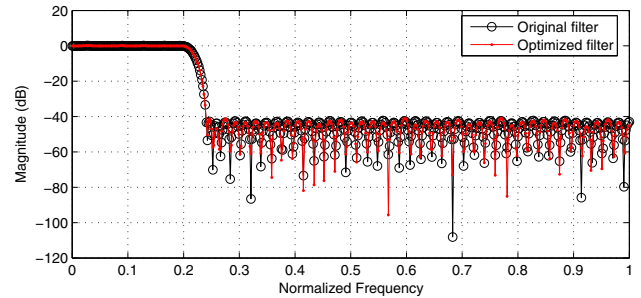

Fig. 5. Zero-phase frequency responses of Filter 3 when ε is 0.005.

to the FDO problem under a tolerable error indicated that its solutions lead to filter designs requiring fewer LUTs and slices on FPGA than original filter designs.

VI. Acknowledgment

References

[1] H. Nguyen and A. Chatterjee, "Number-Splitting with Shift-and-Add Decomposition for Power and Hardware Optimization in Linear DSP Synthesis," *IEEE TVLSI*, vol. 8, no. 4, pp. 419–424, 2000.

[2] K. Chapman, "Fast Integer Multipliers Fit in FPGAs," in *EDN*, 1994, pp. 79–80.

[3] M. Wirthlin, "Constant Coefficient Multiplication Using Look-Up Tables," *Journal of VLSI Signal Processing*, vol. 36, no. 1, pp. 7–15, 2004.

[4] Altera Corporation, Implementing Multipliers in FPGA Devices, Application Note 304, 2004.

[5] Xilinx, Inc., Constant Coefficient Multipliers for the XC4000E, Application Note 054, 1996.

[6] M. Faust and C.-H. Chang, "Bit-Parallel Multiple Constant Multiplication Using Look-Up Tables on FPGA," in *ISCAS*, 2011, pp. 657–660.

[7] R. Venkatesan, A. Agarwal, K. Roy, and A. Raghunathan, "MACACO: Modeling and Analysis of Circuits for Approximate Computing," in *ICCAD*, 2011, pp. 667–673.

[8] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy, "Low-Power Digital Signal Processing Using Approximate Adders," *IEEE TCAD*, vol. 32, no. 1, pp. 124–137, 2013.

[9] L. Aksoy, E. Costa, P. Flores, and J. Monteiro, "Multiple Tunable Constant Multiplications: Algorithms and Applications," in *ICCAD*, 2012, pp. 473–479.

[10] J. Petrlik and L. Sekanina, "Multiobjective Evolution of Approximate Multiple Constant Multipliers," in *DDECS*, 2013, pp. 116–119.

[11] M. Gately, M. Yeary, and C. Tang, "Reduced-Hardware Digital Filter Design via Joint Quantization and Multiple Constant Multiplication Optimization," in *ICASSP*, 2011, pp. 4368–4371.

[12] P. Barth, "A Davis-Putnam Based Enumeration Algorithm for Linear Pseudo-Boolean Optimization," MPII, Tech. Rep., 1995.

[13] Solving Constraint Integer Programs website, http://scip.zib.de/.