ECHO: A Novel Method for the Multiplierless Design of Constant Array Vector Multiplication

Levent Aksoy
INESC-ID
Lisbon, Portugal
Email: levent@algos.inesc-id.pt

Paulo Flores
INESC-ID/Tecnico ULisboa
Lisbon, Portugal
Email: pff@inesc-id.pt

José Monteiro
INESC-ID/Tecnico ULisboa
Lisbon, Portugal
Email: jcm@inesc-id.pt

***Abstract*—The constant array vector multiplication (CAVM) operation realizes the multiplication of a constant array by a vector of variables and occurs in the design of direct form finite impulse response (FIR) and infinite impulse response (IIR) filters. In this paper, for the first time, we directly target the optimization of the multiplierless design of a CAVM operation and introduce a novel algorithm, called ECHO, that can find the fewest number of adders and subtractors required for its implementation. We also describe some hardware optimization techniques that can reduce the gate-level area and delay of the CAVM design. It is shown that the solutions of ECHO include significantly less number of operations and yield less area in FIR filter designs than those of previously proposed state-of-art algorithms.**

I. Introduction

In digital signal processing (DSP) systems, such as fast Fourier transforms (FFTs), FIR filters, and discrete cosine transforms (DCTs), the multiplication of constant(s) by variable(s) is a ubiquitous operation and has a significant impact on the design complexity and performance. Based on its occurrences, it can be categorized in four classes [1]:

1) *Single constant multiplication (SCM)* realizes the multiplication of a single constant c by a single variable x, *i.e.*, $y = cx$. It is used in FFTs and fast DCTs [2].
2) *Multiple constant multiplication (MCM)* implements the multiplication of a set of n constants C by a single variable x, *i.e.*, $y_i = c_i x$, with $1 \leq i \leq n$. It is found in transposed form FIR filters (Fig. 1a) [3].
3) *Constant array vector multiplication (CAVM)* realizes the multiplication of a $1 \times n$ constant array C by an $n \times 1$ variable vector X, *i.e.*, $y = \sum_j c_j x_j$, with $1 \leq j \leq n$. It occurs in IIR and direct form FIR filters (Fig. 1b) [4].
4) *Constant matrix vector multiplication (CMVM)* implements the multiplication of an $m \times n$ constant matrix C by an $n \times 1$ variable vector X, *i.e.*, $y_i = \sum_j c_{ij} x_j$, with $1 \leq i \leq m$ and $1 \leq j \leq n$. It appears in error correcting codes, DCTs, and integer cosine transforms (ICTs) [5].

Since the realization of a multiplier in hardware is expensive in terms of area and the constants are determined beforehand in these DSP systems, these constant multiplications are generally realized using shifts, adders, and subtractors under the shift-adds architecture. Note that shifts by a constant value can be realized using only wires having no hardware cost. Hence, the fundamental optimization problem is defined as finding the minimum number of adders and subtractors required to realize the constant multiplications. This is an NP-complete problem even in the case of SCM [6].

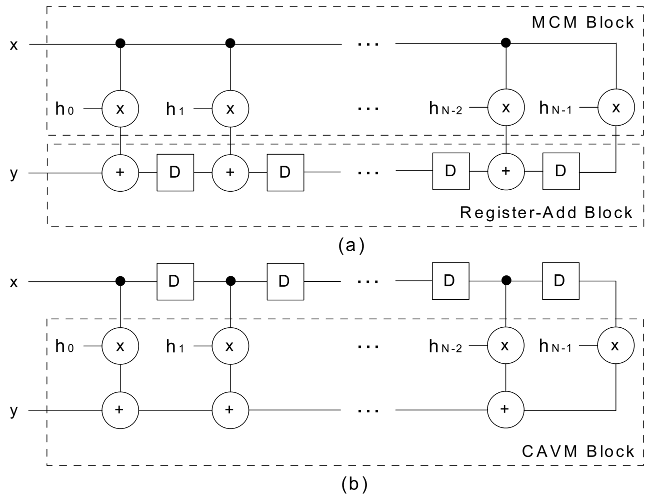

Fig. 1. Design of an N-tap FIR filter: (a) transposed form; (b) direct form.

Over the years many efficient algorithms [1]–[8] have been introduced, including the exact methods of [2], [8] which can guarantee a solution with minimum number of operations in SCM and MCM instances of real-world DSP systems. However, the techniques designed for the CMVM operation have been based on heuristics, meaning that they can neither ensure the minimum solution nor provide any information on how far away their solutions are from the minimum [1]. This is because the complexity of the optimization problem increases as the number of variables and outputs increases. Also, there has been no algorithm designed especially for the CAVM operation for which a CMVM method is used in practice.

In this paper, an efficient method for the multiplierless realization of the CAVM operation, called ECHO, is introduced. It consists of two parts. First, given the $1 \times n$ constant array C, the multiplierless realization of the n constants of C is found using an MCM algorithm. Second, a set of transformations is iteratively applied to the constant multiplications in the CAVM operation, considering the realization of each constant in MCM. Since the MCM solution has a direct impact on the number of operations and adder-steps, *i.e.*, the maximum number of operations in series, of the CAVM design, in its first part, we preferred to use the exact algorithm of [8], which can find the fewest number of operations, and the approximate algorithm of [8], which is modified to handle a delay constraint. In its second part, we also consider some optimization techniques to further reduce the area and delay of the CAVM design. We show its effectiveness on randomly generated instances, comparing with efficient CMVM algorithms and on FIR filters, comparing with direct form FIR filters designed using efficient CMVM methods and with transposed form FIR filters designed using prominent MCM methods.

II. MCM Algorithms

Since the common input variable x is multiplied by multiple constants in MCM, the implementation of constant multiplications is in fact equal to the implementation of constants. For example, $3x$ realized as $3x = x \ll 1 + x$ can be rewritten as $3 = 1 \ll 1 + 1$ by removing the variable x from both sides. This terminology is used interchangeably through this paper.

Since the sign of a constant multiplication can be adjusted where it is required and shifts by a constant value have no hardware cost, given the set of constants C, the MCM methods find a solution on unrepeated positive and odd versions of the constants of C. The basic operation in the shift-adds design of MCM, called *A-operation* in [6], realizes an adder or a subtractor with an arbitrary number of shifts and is given as:

$$\begin{aligned} w = A(u,v) &= |2^{l_1}u + (-1)^s 2^{l_2} v| 2^{-r} \\ &= |(u \ll l_1) + (-1)^s (v \ll l_2)| \gg r \end{aligned} \quad (1)$$

where $l_1, l_2 \geq 0$ are integers denoting left shifts of the operands u and v, respectively, $r \geq 0$ is an integer indicating a right shift, and $s \in \{0,1\}$ is the sign which determines if an addition or a subtraction operation is to be performed.

Although the MCM algorithms are equipped with different search techniques, their common purpose is to maximize the sharing of partial products. They can be grouped in two classes as *common subexpression elimination* (CSE) and *graph-based* (GB) algorithms [1]. The CSE methods [4], [5], [7] first define the constants under a number representation such as binary or canonical signed digit (CSD)[1] [4]. Then, they consider possible subexpressions, which can be extracted from the nonzero digits in representations of constants, and choose the "best" subexpression, generally the most common, to be shared among the constant multiplications. The GB methods [3], [6], [8] are not restricted to any particular number representation and find the intermediate partial products which enable to realize the constant multiplications with minimum number of operations. They obtain better results than the CSE methods, since they consider a large number of alternative realizations of a constant [8]. As a simple example, consider $C = \{51, 77\}$. Their realizations obtained by the exact CSE algorithm [7] when constants are defined under CSD and by the exact GB algorithm [8] are respectively given in Figs. 2a-b.

The delay of a multiplierless MCM design is generally considered as the number of adder-steps. The minimum number of adder-steps of a single constant c is computed as $\lceil log_2 S(c) \rceil$, where $S(c)$ denotes the number of nonzero digits in the CSD representation of c [3]. Thus, for a set of n constants C, the minimum number of adder-steps, MAS_{MCM}, is determined as $\max_i \{\lceil log_2 S(c_i) \rceil\}$ with $1 \leq i \leq n$ [3]. Given the constant set C and a delay constraint dc with $dc \geq \mathrm{MAS}_{MCM}$, the optimization problem is to find the

[1]An integer is written in CSD representation using k digits as $\sum_{i=0}^{k-1} b_i 2^i$, where $b_i \in \{-1,0,1\}$. The nonzero digits are not adjacent and a constant is represented with minimum number of nonzero digits under CSD.

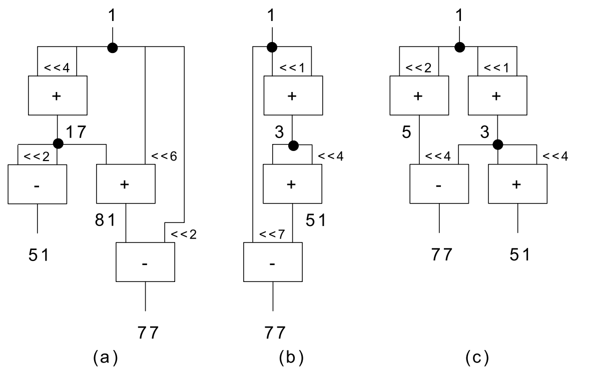

Fig. 2. Shift-adds design of 51 and 77: (a) exact CSE method [7]; (b) exact GB method [8]; (c) approximate GB algorithm [8] under a delay constraint.

fewest number of *A-operations*, realizing the constants of C without violating dc [3]. Returning to our example, the solution of the approximate GB algorithm [8] modified to handle the delay constraint is given in Fig. 2c, where dc was set to MAS_{MCM} which is 2. When compared to the minimum solution in terms of the number of operations (Fig. 2b), it includes one more operation, but one less adder-step.

III. ECHO: The CAVM Algorithm

This section introduces ECHO and describes its two main parts through a simple example with $C = [51, 77, 154]$.

In its first part, given the constant array C, we find the multiplierless realization of the constants of C using an MCM algorithm. As will be shown, the MCM solution has a direct impact on the CAVM design in terms of the number of operations and adder-steps. Hence, in ECHO, the exact GB method [8] and the modified version of the approximate GB method [8], which can handle the delay constraint, are used, since they obtain better solutions in terms of the number of operations than other prominent MCM algorithms. These GB methods were also needed to be modified to generate a solution including *A-operations* with right shifts r equal to 0 (Eq. 1), otherwise a constant multiplication in the CAVM operation may not be synthesized. Such operations rarely occur in GB algorithms, and depending on constants, the MCM design requires zero or a few extra operations under this constraint. For our example, the MCM solutions for the unrepeated odd constants of C, *i.e.*, $\{51, 77\}$, are given in Figs. 2b-c.

Then, the adders and subtractors in the MCM solution are stored in a set called O and the level of each constant p generated by an operation, $level(p)$, is found starting from the primary input 1 with $level(1) = 0$. The maximum value of the levels of constants, mlc, is also computed. For our example, considering the MCM solution of Fig. 2b, $level(3)$, $level(51)$, and $level(77)$ are 1, 2, and 3, respectively and mlc is 3.

Its second part starts by multiplying the $1 \times n$ constant array C by the $n \times 1$ variable vector X and obtaining y in the form of $y = d_1 x_1 \ll ls_1 + d_2 x_2 \ll ls_2 + \ldots + d_n x_n \ll ls_n$, where $c_i = d_i \ll ls_i$ with $1 \leq i \leq n$. For our example, y is written as $51x_1 \ll 0 + 77x_2 \ll 0 + 77x_3 \ll 1$.

Then, we apply the CONV2CAVM procedure in Fig. 3, where T will include linear subexpressions to be realized in the CAVM operation and $step$, set to mlc in the beginning, will be used in its iterative loop. First, we find the common constants

CONV2CAVM($O, level, mlc, y$)

```
1:  $T = [\,]$, $step = mlc$
2:  $y$ = factor($y$), $[T, y]$ = eliminate($T$, $y$)
3:  **repeat**
4:      **for** $k = 1$ to #variables of $y$ **do**
5:          **if** $step = level(d_k)$ **then**
6:              $y$ = substitute($y, O, d_k$)
7:      $y$ = expand($y$), $y$ = factor($y$), $[T, y]$ = eliminate($T$, $y$)
8:      $step = step - 1$
9:  **until** $step = 0$
10: $T = T \bigcup y$
11: **return** $T$
```

Fig. 3. The CONV2CAVM procedure in the second part of ECHO.

in y if available and group them using the *factor* function and replace the factored subexpressions with new variables and remove them from y to T using the *eliminate* function. In our example, y is obtained as $51x_1 \ll 0 + 77(x_2 \ll 0 + x_3 \ll 1)$ after the *factor* function and as $51x_1 \ll 0 + 77a \ll 0$ after the *eliminate* function, where $a = x_2 \ll 0 + x_3 \ll 1$ which is stored in T. Then, in its iterative loop, all the constants of y having the $step$ value are replaced with their realizations in O using the *substitute* function. For our example, when $step$ is 3, 77 is replaced with its realization in Fig. 2b as $y = 51x_1 \ll 0 + (128 - 51)a \ll 0$. Then, y is expanded using the *expand* function, and the *factor* and *eliminate* functions are applied. For our example, after expansion, y is computed as $51x_1 \ll 0 + a \ll 7 - 51a \ll 0$, and after factorization and elimination at the end of the first iteration, it is found as $51b \ll 0 + a \ll 7$, where $b = x_1 \ll 0 - a \ll 0$. The iterative loop continues until $step$ is 0, considering all constants in the MCM solution. For our example, y is $3c \ll 0 + a \ll 7$ and $c + c \ll 1 + a \ll 7$ at the end of the second and third iteration, respectively, where $c = b + b \ll 4$. Then, the final y is added to T. Note that all the linear subexpressions of T require a total of $M + nzc - 1$ operations, where M is the number of operations found by the MCM algorithm and nzc is the number of nonzero constants of C. For our example, both M and nzc are 3 and a single operation is required for a, b, and c and 2 adders are needed for the final y, a total of 5 operations.

Finally, each linear subexpression in T is realized in order using 2-input adders/subtractors. For subexpressions including t terms with $t > 2$, since there are more than one possible realization, we consider the bitwidth or adder-step of each term to reduce the area or delay of the CAVM design, respectively. We iteratively find 2 terms with the minimum bitwidth (adder-step), realize them using an adder or a subtractor, and replace these two terms with the output of this operation. If there exist more than 2 terms with minimum bitwidth (adder-step), we favor the ones with the smallest adder-step (bitwidth). This iterative loop continues until the number of terms is 1. For our example, considering the bitwidths of terms, the final y with t is 3, is realized as $d = c + c \ll 1$ and $y = d + a \ll 7$.

Thus, two variations of ECHO were developed. ECHO-A uses the exact GB algorithm [8] and realizes the linear subexpressions in T considering the bitwidths of their terms. ECHO-D uses the approximate GB algorithm [8], which obtains a solution under the delay constraint equal to MAS_{MCM}, and realizes the linear subexpressions in T considering the adder-steps of their terms. For our example, the solutions of ECHO-A (as described in this section) and ECHO-D obtained respectively based on the MCM solutions in Figs. 2b-c are

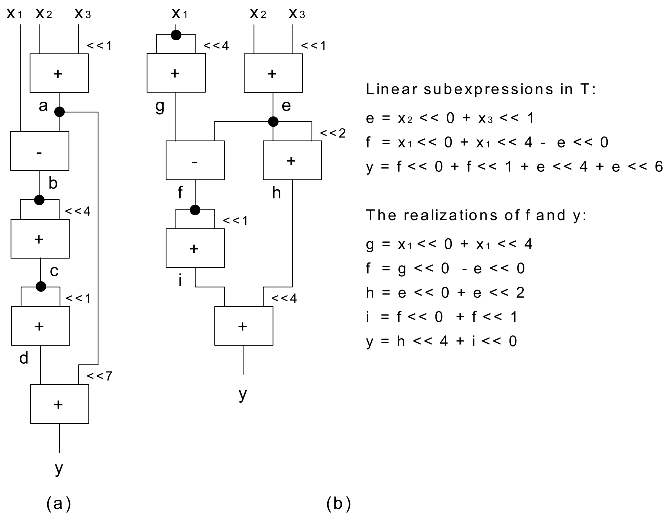

Fig. 4. Realizations of $51x_1 + 77x_2 + 154x_3$: (a) ECHO-A; (b) ECHO-D.

given in Figs. 4a-b. The linear subexpressions found by ECHO-D and their realizations are also given in Fig. 4b. Observe that ECHO-D finds a CAVM solution with one more operation than ECHO-A, but with one less adder-step, showing the impact of the MCM solution and the strategy used in the realization of subexpressions in T.

IV. Experimental Results

As the first experiment set, we used 12-bit randomly generated $1 \times n$ constant arrays, where n is in between 10 and 80. For each n, there were 30 CAVM instances. The results of the proposed algorithms are presented in Fig. 5, which are compared to those of the state-of-art CMVM algorithms [1], HCMVM and HCMVM-DC, that respectively can find a solution without a delay constraint and with a delay constraint set to the minimum adder-steps of the CAVM operation [1]. All these algorithms were written in MATLAB and run on a PC with Intel Xeon at 2.4GHz and 10GB memory.

The proposed algorithms find significantly better solutions than the CMVM methods on CAVM instances in terms of the average number of operations, where the difference between the results of HCMVM (HCMVM-DC) and ECHO-A (ECHO-D) increases as n increases and reaches up to 24.47 (26.13). The CPU time required for the proposed techniques is related to the performance of the MCM algorithms and is less than those of the CMVM algorithms when $n \geq 30$. Although ECHO-D obtains solutions with less number of adder-steps than ECHO-A and close to those of HCMVM-DC, it cannot ensure the minimum number of adder-steps in CAVM as HCMVM-DC.

As the second experiment set, we used 3 low-pass FIR filters whose specifications are given in Table I. The results of algorithms on the multiplierless design of the MCM (CAVM) blocks of the transposed (direct) form FIR filters as shown in Fig. 1a (Fig. 1b) are presented in Table II. In this table, *MBO* and *AS* stand respectively for the number of operations and adder-steps in the multiplier blocks and *TO* is the total number of operations in the whole filter, considering the ones in the register-add block of the transposed form (Fig. 1a). These filters were described in VHDL and synthesized using the Synopsys Design Compiler with the UMCLogic 0.18-μm Generic II library, when the bitwidth of the filter input was

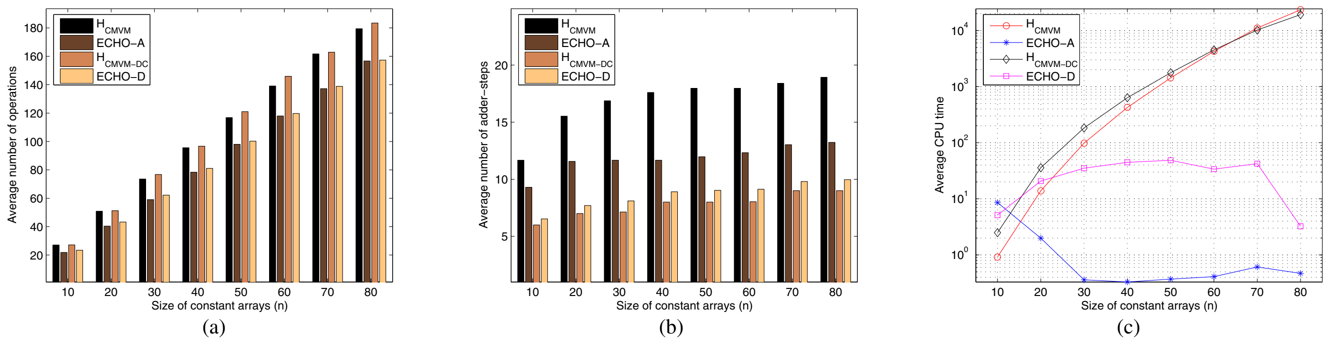

Fig. 5. Average results on randomly generated CAVM instances: (a) number of operations; (b) number of adder-steps; (c) CPU time in seconds in log scale.

TABLE I. SPECIFICATIONS OF FILTERS WITH SYMMETRIC COEFFICIENTS.

Filter	filter length	normalized passband	normalized stopband	quantization value
1	60	0.15	0.2	16
2	80	0.12	0.15	16
3	100	0.18	0.2	16

16. Their gate-level results are also given in Table II, where A, D, and P are respectively the total area in mm^2, the critical path delay in ns, and the dynamic power dissipation in mW obtained with the use of 10,000 input vectors in simulation.

The proposed algorithms obtain better solutions in terms of the number of operations for the CAVM blocks of the direct form FIR filters with respect to the CMVM algorithms. This advantage also yields significant area reduction in the filter designs, where the maximum gain is found as 12.1% on Filter 1 when the solutions of HCMVM-DC and ECHO-D are compared. The solutions of ECHO-D lead to filters with less delay than those of ECHO-A, where the maximum gain is computed as 22.9% on Filter 1, taking into account a slight increase in area. The filters designed based on the solutions of the proposed algorithms and those of [1] have similar power consumption.

Observe also that the number of total operations in the direct form obtained by the proposed algorithms is the same or very close to that of the transposed form, since the only difference between the MCM algorithms is that, in ECHO-A and ECHO-D, they are modified not to generate an operation with a right shift greater than 0. Moreover the direct form filters occupy less area due to the size of registers and consume less power, but may have a larger delay than the transposed form.

V. CONCLUSIONS

This paper presented a novel method for the multiplierless design of the CAVM operation which is generally realized using a CMVM algorithm. It uses the solutions of MCM algorithms on the constants of the CAVM operation, where the sharing of partial products is maximized. It is also equipped with area and delay reduction techniques. The experiments showed its effectiveness on the solution quality and runtime with respect to the prominent CMVM algorithms.

VI. ACKNOWLEDGMENT

The authors thank to Prof. Ya Jun Yu for providing the main idea behind the proposed algorithms. This work was supported by national funds through FCT, Fundação para a Ciência e a Tecnologia, under the project EXPL/EEI-ELC/1016/2012 *QCell - Configurable Logic Block Cell for Quaternary FPGAs* and the project PEst-OE/EEI/LA0021/2013.

TABLE II. SUMMARY OF RESULTS OF ALGORITHMS ON FIR FILTERS.

Fil.	Form	Algo.	High-level			Gate-level		
			MBO	AS	TO	A	D	P
1	Tran.	[8]*	31	11	90	100.1	12.4	10.3
		[8]**	36	3	95	102.5	9.1	11.0
	Dir.	[1]+	109	20	109	71.7	11.5	2.4
		[1]++	109	7	109	74.3	10.1	2.5
		ECHO-A	91	20	91	63.2	13.1	2.5
		ECHO-D	95	9	95	65.3	10.1	2.4
2	Tran.	[8]*	41	14	120	135.5	13.3	13.9
		[8]**	49	3	128	139.5	9.4	14.7
	Dir.	[1]+	145	22	145	97.7	10.6	3.3
		[1]++	150	8	150	100.1	10.6	3.3
		ECHO-A	121	16	121	85.9	12.7	3.4
		ECHO-D	128	10	128	89.4	10.5	3.3
3	Tran.	[8]*	48	7	147	167.3	13.0	18.2
		[8]**	56	3	155	171.9	9.4	18.9
	Dir.	[1]+	173	20	173	116.6	10.5	4.1
		[1]++	177	8	177	118.8	10.8	4.0
		ECHO-A	147	18	147	103.9	11.9	4.1
		ECHO-D	155	10	155	108.5	10.7	4.4

* The exact GB algorithm [8] + HCMVM ++ HCMVM-DC
** The approximate GB algorithm [8] with a dc set to MAS_{MCM}

REFERENCES

[1] L. Aksoy, E. Costa, P. Flores, and J. Monteiro, "Multiplierless Design of Linear DSP Transforms," in *VLSI-SoC: Advanced Research for Systems on Chip*. Springer, 2012, ch. 5, pp. 73–93.

[2] J. Thong and N. Nicolici, "A Novel Optimal Single Constant Multiplication Algorithm," in *Proceedings of Design Automation Conference (DAC)*, 2010, pp. 613–616.

[3] H.-J. Kang and I.-C. Park, "FIR Filter Synthesis Algorithms for Minimizing the Delay and the Number of Adders," *IEEE Transactions on Circuits and Systems II*, vol. 48, no. 8, pp. 770–777, 2001.

[4] R. Hartley, "Subexpression Sharing in Filters Using Canonic Signed Digit Multipliers," *IEEE Transactions on Circuits and Systems II*, vol. 43, no. 10, pp. 677–688, 1996.

[5] N. Boullis and A. Tisserand, "Some Optimizations of Hardware Multiplication by Constant Matrices," *IEEE Transactions on Computers*, vol. 54, no. 10, pp. 1271–1282, 2005.

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

[7] L. Aksoy, E. Costa, P. Flores, and J. Monteiro, "Exact and Approximate Algorithms for the Optimization of Area and Delay in Multiple Constant Multiplications," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 6, pp. 1013–1026, 2008.

[8] L. Aksoy, E. Gunes, and P. Flores, "Search Algorithms for the Multiple Constant Multiplications Problem: Exact and Approximate," *Elsevier Journal on Microprocessors and Microsystems: Embedded Hardware Design*, vol. 34, no. 5, pp. 151–162, 2010.