# Design of Low-Power Multiple Constant Multiplications Using Low-Complexity Minimum Depth Operations

Levent Aksoy
INESC-ID
Lisboa, Portugal
levent@algos.inesc-id.pt

Eduardo Costa
Univ. Católica de Pelotas
Pelotas, Brazil
ecosta@ucpel.tche.br

Paulo Flores, José Monteiro
INESC-ID/IST TU Lisbon
Lisboa, Portugal
{pff, jcm}@inesc-id.pt

## ABSTRACT

Existing optimization algorithms for the multiplierless realization of multiple constant multiplications (MCM) typically target the minimization of the number of addition and subtraction operations. Since power dissipation is directly related to the amount of hardware, some power reduction is indirectly achieved by these algorithms. However, in many cases, glitching plays an equally important role in defining the power consumption. This is specially true for arithmetic circuits, and in particular to MCM due to high logic depth and large number of re-convergent paths. This paper introduces exact algorithms that search the optimal area of an MCM design at gate-level where each constant multiplication is implemented in its minimum depth. Experimental results show that the proposed algorithms lead to MCM designs consuming significantly less power with respect to those obtained by the MCM algorithms.

## Categories and Subject Descriptors

B.2.0 [**Arithmetic and Logic Structures**]: General

## General Terms

Algorithms, Design

## Keywords

Multiple constant multiplications (MCM), 0-1 integer linear programming, high-level synthesis, low-power MCM design

## 1. INTRODUCTION

The Multiple Constant Multiplications (MCM) operation, that realizes the multiplication of a set of known constants by a variable, is a central operation and performance bottleneck in many Digital Signal Processing (DSP) applications such as digital Finite Impulse Response (FIR) filters, linear DSP transforms, and error correcting codes. Since the multiplication operation is expensive in terms of area, delay, and power dissipation in hardware and the constants to be multiplied with the same variable are known beforehand, the full-flexibility of a multiplier is not necessary in the design of the MCM operation. Hence, constant multiplications are generally realized using addition/subtraction and shift operations [12].


*GLSVLSI'11,* May 2–4, 2011, Lausanne, Switzerland.


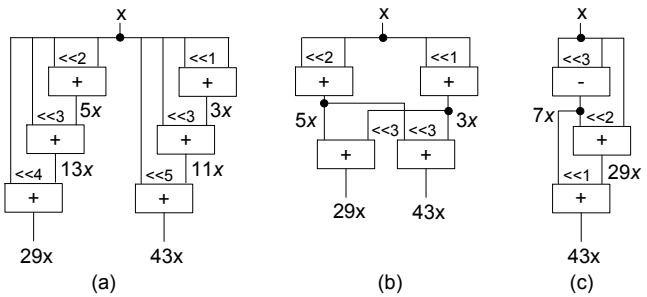


**Figure 1: Shift-adds implementations of $29x$ and $43x$: (a) without partial product sharing [9]; with partial product sharing: (b) the algorithm of [1]; (c) the algorithm of [2].**

For the implementation of constant multiplications using addition/subtraction and shift operations, a straightforward method, generally known as the digit-based recoding [9], initially defines the constants in multiplications in binary. Then, for each 1 in the binary representation of the constant, according to its bit position, it shifts the variable and adds up the shifted variables to obtain the result. As a simple example, consider the constant multiplications $29x$ and $43x$. Their decompositions in binary are listed as follows:

$$29x = (11101)_{bin}x = x \ll 4 + x \ll 3 + x \ll 2 + x$$
$$43x = (101011)_{bin}x = x \ll 5 + x \ll 3 + x \ll 1 + x$$

and require 6 addition operations using the digit-based recoding method [9], as presented in Figure 1(a).

However, the sharing of common partial products in the shift-adds architecture allows for great reductions in the number of operations and consequently, in area and power dissipation of the MCM design. Hence, the MCM problem is defined as finding the minimum number of addition/subtraction operations that implement the constant multiplications, since shifts can be realized using only wires in hardware without representing any area cost. Note that the MCM problem is an NP-complete problem [6].

The algorithms designed for the MCM problem can be categorized in two classes: Common Subexpression Elimination (CSE) algorithms [1, 10, 13] and graph-based (GB) methods [2, 16]. Although both CSE and GB algorithms aim to maximize the sharing of partial products, they differ in the search space that they explore. The CSE algorithms initially define the constants under a number representation. Then, all possible subexpressions are extracted from the representations of the constants and the "best" subexpression, generally, the most common, is chosen to be shared among the constant multiplications. The GB algorithms are not limited to any particular number representation and consider a larger number of alternative implementations of a constant multiplication, yielding better solutions than the CSE algorithms as shown in [2, 16].

Returning to our example in Figure 1, the exact CSE algorithm [1] gives a solution with 4 operations by finding the most common partial products $3x = (11)_{bin}x$ and $5x = (101)_{bin}x$ when constants are defined under binary, (Figure 1(b)). The exact GB algorithm [2] finds the minimum number of operations solution with 3 operations by sharing the common partial product $7x$ in both multiplications, (Figure 1(c)). Observe that the partial product $7x = (111)_{bin}x$ cannot be extracted from the binary representations of both multiplications $29x$ and $43x$ in the exact CSE algorithm [1].

In many DSP systems, performance is an important and crucial parameter. Hence, circuit area is generally expendable in order to achieve a given performance target. Although the delay parameter is dependent on several implementation issues, such as circuit technology, placement, and routing, the delay of the MCM operation is generally considered as the number of adder-steps, which denotes the maximal number of adders/subtracters in series to produce any constant multiplication [11]. The algorithms that find the fewest number of operations in an MCM instance while taking into account the delay constraint were introduced in [1, 11, 15].

As can be observed from Figure 1, a constant multiplication can be implemented with a number of different addition/subtraction operations each having a different implementation cost in hardware. The gate-level implementation cost of all possible addition and subtraction operations in an MCM operation and an algorithm that reduces the hardware complexity were presented in [3]. In this model, an addition/subtraction operation is assumed to be implemented under the ripple carry adder architecture and its hardware cost is determined in terms of full adders, half adders, and additional logic gates.

With the increasing popularity of portable electronic devices that include many DSP systems, power consumption has become a matter of concern. Switching activity and power dissipation estimation models for the MCM operation have been introduced in [7, 8]. As shown in the Glitch Path Score (GPS) power dissipation estimation model [8], power dissipation in MCM circuits is highly related with the depth of each operation and its area at gate-level. Hence, in this paper, we introduce exact algorithms that yield MCM designs, where each constant multiplication is realized at its minimum depth using optimal area at gate-level. Experimental results indicate that the proposed algorithms yield low-power MCM designs when compared to those obtained by the algorithms designed for the MCM problem and for the MCM problem under a delay constraint.

The rest of the paper is organized as follows. Section 2 presents the background concepts. The exact algorithms are introduced in Section 3. Section 4 presents the experimental results and finally, conclusions are given in Section 5.

# 2. BACKGROUND

In this section, we give the basic concepts related with the proposed exact algorithms and introduce the problem definitions.

## 2.1 Number Representation

The *binary* representation decomposes a number in a set of additions of powers of two. The representation of numbers using a signed digit system makes the use of positive and negative digits, $\{\bar{1}, 0, 1\}$, where $\bar{1}$ stands for $-1$. The *Canonical Signed Digit* (CSD) representation [4] is a signed digit system that has a unique representation for each number and verifies the following properties: i) two non-zero digits are not adjacent; ii) the number of non-zero digits is minimal. Any $n$ digit number in CSD has at most $\lceil (n+1)/2 \rceil$ non-zero digits, and on average, the number of non-zero digits is reduced by 33% when compared to binary. The *Minimal Signed Digit* (MSD) representation [13] is obtained by dropping the first property of the CSD representation. Thus, a constant can have several representations under MSD, including its CSD representation, but all with a minimum number of non-zero digits.

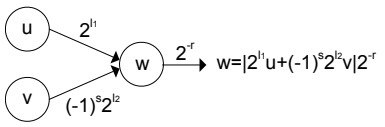


**Figure 2: The graph representation of an A-operation.**

## 2.2 Multiplierless Constant Multiplications

In multiplierless constant multiplications, the fundamental operation, called *A-operation* in [16], is an operation with two integer inputs and one integer output that performs a single addition or a subtraction, and an arbitrary number of shifts. It is defined as:

$$w = A(u,v) = |2^{l_1}u + (-1)^s 2^{l_2}v|2^{-r} \quad (1)$$

where $s \in \{0,1\}$ is the sign, which determines if an addition or a subtraction operation is to be performed, $l_1, l_2 \geq 0$ are integers denoting left shifts of the operands, and $r \geq 0$ is an integer indicating a right shift of the result. An *A-operation* can be represented in a graph where the vertices are labeled with the constants and the edges are labeled with the sign and shifts as illustrated in Figure 2.

In the MCM problem, the complexity of an adder and a subtracter in hardware is assumed to be equal. It is also assumed that the sign of the constant can be adjusted at some part of the design and the shifting operation has no cost in hardware. Thus, only positive and odd constants are considered in the MCM problem. Observe from Eqn. (1) that in the implementation of an odd constant considering any two odd constants at the inputs, one of the left shifts, $l_1$ or $l_2$, is zero and $r$ is zero, or both $l_1$ and $l_2$ are zero and $r$ is greater than zero. Also, it is necessary to constrain the left shifts, otherwise there exist infinite ways of implementing a constant. In [2], the number of shifts is allowed to be at most $mb+1$, where $mb$ is the maximum bit-width of the constants to be implemented. Thus, the MCM problem [16] can be defined as follows:

DEFINITION 1. THE MCM PROBLEM. *Given the target set composed of positive and odd unrepeated target constants to be implemented, $T = \{t_1, \ldots, t_n\} \subset \mathbb{N}$, find the smallest ready set, $R = \{r_0, r_1, \ldots, r_m\}$, with $T \subset R$, such that $r_0 = 1$ and for all $r_k$ with $1 \leq k \leq m$, there exist $r_i, r_j$ with $0 \leq i, j < k$ and an A-operation $r_k = A(r_i, r_j)$.*

The minimum adder-step realization of a target constant $t_i$ includes $\lceil log_2 S(t_i) \rceil$ operations in series, where $S(t_i)$ denotes the number of non-zero digits of $t_i$ in its CSD representation. Hence, for a target set, $T = \{t_1, \ldots, t_n\}$, the minimum adder-step of the MCM operation [11] is computed as:

$$min\_delay_{MCM} = \max_{t_i} \{\lceil log_2 S(t_i) \rceil\} \quad (2)$$

Thus, the optimization of the number of operations problem under a delay constraint can be defined as follows:

DEFINITION 2. THE MCM PROBLEM UNDER A DELAY CONSTRAINT. *Given the target set $T = \{t_1, \ldots, t_n\} \subset \mathbb{N}$ and the delay constraint $dc$ with $dc \geq min\_delay_{MCM}$, find the smallest ready set $R = \{r_0, r_1, \ldots, r_m\}$ such that under the same conditions on the ready set given in Definition 1, the set of A-operations yields an MCM design without exceeding $dc$.*

Note that the area of an *A-operation* at gate-level, that is directly proportional to the number of logic gates, has a significant effect on the power dissipation as a larger number of logic gates produce more transitions. Also, the depth of an *A-operation* in the MCM design affects the power dissipation as the transitions generated at the

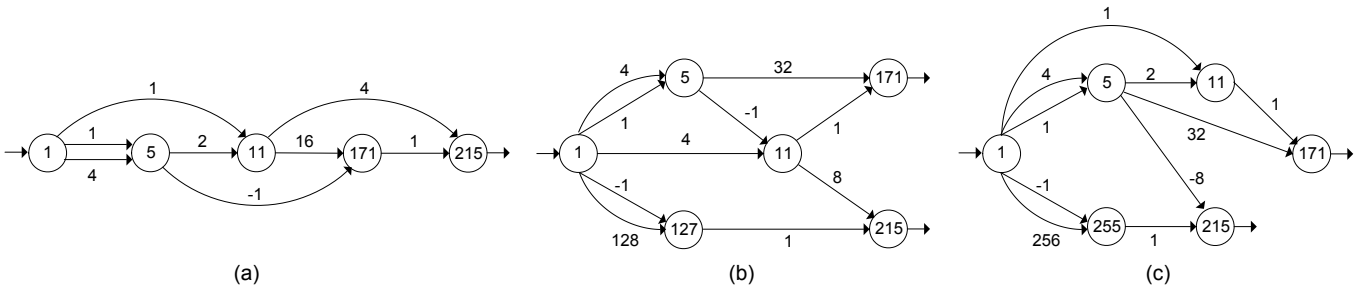


**Figure 3: Implementations of the target set $\{5,11,171,215\}$: (a) with the minimum number of operations; (b) with the minimum number of adder-steps; (c) with the minimum depth constraint for each operation.**

output of an operation produce more transitions on the next level operations and more glitching is generated and propagated along the re-convergent paths. Hence, a low-power MCM operation can be designed when each constant is realized in its minimum depth using optimal area at gate-level. Thus, the optimization of area problem under the minimum depth constraint can be defined as:

DEFINITION 3. THE OPTIMIZATION OF AREA PROBLEM UNDER THE MINIMUM DEPTH CONSTRAINT. *Given the target set $T = \{t_1,\ldots,t_n\} \subset \mathbb{N}$, find the ready set $R = \{r_0, r_1, \ldots, r_m\}$ such that under the same conditions on the ready set given in Definition 1, the set of A-operations yields an MCM design where each constant is implemented under its minimum depth using optimal area at gate-level.*

An an example on shift-adds realization of constant multiplications with the minimum adder-step constraint and the minimum depth constraint for each operation, consider the target constants in $T = \{5,11,171,215\}$ The minimum adder-steps of 5, 11, 171, and 215 are 1, 2, 3, and 2 respectively. Thus, the minimum adder-step realization of the MCM operation includes 3 adder-steps as computed by Eqn. (2). The exact GB algorithm of [2] designed for the MCM problem finds a solution with 4 operations and 4 adder-steps without requiring any intermediate constant, as illustrated in Figure 3(a). The GB algorithm of [15], called Hcub-DC, that is designed for the MCM problem under a delay constraint gives a solution with 5 operations under the minimum delay constraint, *i.e.*, 3, as shown in Figure 3(b). The exact GB algorithm proposed in this paper finds a solution including 5 operations, where each constant is implemented at its minimum depth, as presented in Figure 3(c).

## 2.3 0-1 Integer Linear Programming

The *0-1 Integer Linear Programming* (ILP) problem is the minimization or the maximization of a linear cost function subject to a set of linear constraints and is generally defined as follows[1]:

$$\text{Minimize} \quad \mathbf{c}^T \cdot \mathbf{x} \tag{3}$$

$$\text{Subject to} \quad \mathbf{A} \cdot \mathbf{x} \geq \mathbf{b}, \quad \mathbf{x} \in \{0,1\}^n \tag{4}$$

In (3), $c_j$ in $\mathbf{c}$ is an integer value associated with each of $n$ variables $x_j$, $1 \leq j \leq n$, in the cost function, and in (4), $\mathbf{A} \cdot \mathbf{x} \geq \mathbf{b}$ denotes the set of $m$ linear constraints, where $\mathbf{b} \in \mathbb{Z}^n$ and $\mathbf{A} \in \mathbb{Z}^m \times \mathbb{Z}^n$.

# 3. THE EXACT ALGORITHMS

The exact CSE and GB algorithms designed for the optimization of area problem under the minimum depth constraint consist of four main parts:

1. Generation of all possible implementations of the constants.
2. Construction of the Boolean network that represents the implementations of the constants.
3. Formalization of the problem as a 0-1 ILP problem.
4. Obtaining the minimum solution using a 0-1 ILP solver.

In next sections, these parts are described in detail.

## 3.1 Generation of Constant Implementations

In the preprocessing phase of the exact algorithms, the target constants to be implemented are converted to positive and then, made odd by successive divisions by 2. The resulting constants are stored in a set called target set, $T$, without repetition. Thus, the target set includes the minimum number of necessary target constants to be implemented. The implementations of the target and intermediate constants are found as follows:

1. Take an element from $T$, $t_i$, and determine its minimum depth value, $d_{t_i}$, as $\lceil log_2 S(t_i) \rceil$. Form an empty set, $O_i$, associated with $t_i$ that will include the gate-level implementation cost of all *A-operations* that generate $t_i$ and their inputs as a pair.
2. For each *A-operation* that computes $t_i$ where the minimum depth values of the inputs, $d_u$ and $d_v$, are smaller than $d_{t_i}$;
   (a) Determine the implementation cost of the *A-operation*.
   (b) Add the inputs of the *A-operation* to $O_i$ as a pair, *i.e.*, $u$ and $v$, with its implementation cost.
   (c) Add $u$ and $v$ to $T$, if they do not represent the input that is the constants are multiplied with, *i.e.*, '1', and are not in $T$.
3. Repeat Step 1 until all the elements of $T$ are considered.

Observe that the target set, that only includes the target constants to be implemented in the beginning of the iterative loop, is augmented with the intermediate constants that are required for the implementation of target constants, *i.e.*, the inputs of each *A-operation* that generates a target constant.

The only difference between the exact CSE and GB algorithms is how the minimum depth implementations of a constant are found. In our realization of the exact CSE and GB algorithms, all possible minimum depth implementations of a constant are found beforehand and stored in a look-up table. Hence, in Step 2 of the algorithm, each *A-operation* that generates a constant is taken from these look-up tables. The implementation cost of an *A-operation*, Step 2(a) of the algorithm, is computed as described in [3].

Before the construction of the look-up tables, we determine all the constants at minimum depth 1, 2, and 3. Note that any positive and odd constant in between $[3, 2^{15}-1]$ has the minimum depth value 1, 2, or 3. In the construction of the look-up table for the exact GB algorithm, while finding a minimum depth implementation for the constant $c$, *i.e.*, an *A-operation*, $c$ is assigned to the output and the constants from depths less than that minimum of $c$ are assigned to the inputs of the *A-operation*. The possible input

[1] The maximization objective can be easily converted to a minimization objective by negating the cost function. Less-than-or-equal and equality constraints are accommodated by the equivalences, $\mathbf{A} \cdot \mathbf{x} \leq \mathbf{b} \Leftrightarrow -\mathbf{A} \cdot \mathbf{x} \geq -\mathbf{b}$ and $\mathbf{A} \cdot \mathbf{x} = \mathbf{b} \Leftrightarrow (\mathbf{A} \cdot \mathbf{x} \geq \mathbf{b}) \wedge (\mathbf{A} \cdot \mathbf{x} \leq \mathbf{b})$, respectively.

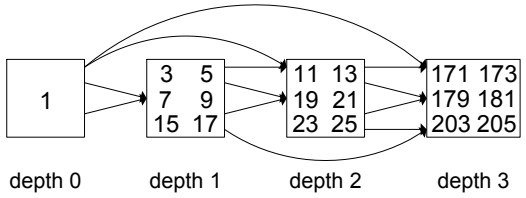


**Figure 4: Possible minimum depth realizations of constants.**

assignments in the *A-operation* are illustrated in Figure 4, where an arrow pair intersecting at one point represents an *A-operation* and each box includes some constants at a depth value. For an example, a depth 2 constant can be implemented using an operation where both inputs are depth 1 constants or using an operation where one of its input is a depth 1 and the other is a depth 0 constant. Then, we search for the values on the left and right shifts and sign value in the *A-operation* that computes $c$. Note that shifts are restricted with the bit-width of the constant plus 1 and only one of the shifts is set to a value greater than 0 and the others are set to 0, since only positive and odd constants are considered. As an example, 3 implementations out of 17 of a depth 2 constant 25 can be given as, $25 = 3 \ll 3+1$, $25 = 5 \ll 2+5$, and $25 = 7 \ll 2-3$.

In the construction of the look-up table for the exact CSE algorithm, we represent $c$ under a particular number representation, *i.e.*, CSD or MSD. Then, we decompose the non-zero digits in its representation(s) in two parts and find all possible implementations considering the minimum depth of each input of the operation, as illustrated in Figure 4. All implementations of the depth 2 constant 25 obtained from its MSD representations, 11001 and $10\bar{1}001$, are given in Figure 5. Observe that there are 5 possible different implementations of 25 (the last implementation on each representation is the same hence, one of them can be eliminated).

$$25 = \begin{cases} 011001 = \begin{cases} 010000 + 001001 = 1 \ll 4+9 \\ 001000 + 010001 = 1 \ll 3+17 \\ 000001 + 011000 = 1+3 \ll 3 \end{cases} \\ 10\bar{1}001 = \begin{cases} 100000 + 00\bar{1}001 = 1 \ll 5-7 \\ 00\bar{1}000 + 100001 = -1 \ll 3+33 \\ 000001 + 10\bar{1}000 = 1+3 \ll 3 \end{cases} \end{cases}$$

**Figure 5: Possible implementations of 25 under MSD.**

In Table 1, we present the number of constants, *noc*, and the total number of operations that generate all the constants, *tno*, for depth values 1, 2, and 3 when constants are in between 8 and 13 bits, *bw*. Each entry in columns *tno* of Table 1, $a/b$, stands for the value considered in the exact CSE algorithm under MSD and in the exact GB algorithm, respectively. For example, for positive and odd constants in between 3 and 255, *i.e.*, 8-bit constants, there are 13, 106, and 8 depth 1, 2, and 3 constants respectively and for the eight depth 3 constants, there are 432 and 2060 possible implementations to be considered in the exact CSE algorithm under MSD and the exact GB algorithm respectively. Observe that the significant difference in the number of possible operations considered in the exact CSE and GB algorithms shows itself for the depth

**Table 1: Number of constants and operations for depths 1, 2, and 3.**

| bw | Depth 1 | | Depth 2 | | Depth 3 | |
|---|---|---|---|---|---|---|
| | noc | tno | noc | tno | noc | tno |
| 8 | 13 | 14 / 14 | 106 | 514 / 947 | 8 | 432 / 2,060 |
| 9 | 15 | 16 / 16 | 192 | 880 / 1,503 | 48 | 1,904 / 20,612 |
| 10 | 17 | 18 / 18 | 318 | 1,390 / 2,237 | 176 | 7,304 / 111,074 |
| 11 | 19 | 20 / 20 | 492 | 2,068 / 3,173 | 512 | 21,936 / 418,092 |
| 12 | 21 | 22 / 22 | 722 | 2,938 / 4,335 | 1,304 | 62,424 / 1,239,820 |
| 13 | 23 | 24 / 24 | 1,016 | 4,024 / 5,747 | 3,056 | 163,376 / 3,122,306 |

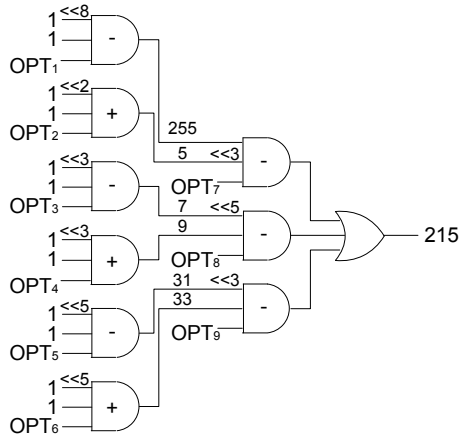


**Figure 6: Network generated for the constant 215.**

3 constants, while these values for the depth 2 constants are close to each other and they are the same for the depth 1 constants.

## 3.2 The Boolean Network

After all possible implementations of the target and intermediate constants are found, these implementations are represented in a Boolean network that includes only AND and OR gates. The properties of the Boolean network are as follows:

1. The primary input of the network is the input to be multiplied with the constants denoted by '1'.
2. An AND gate in the network represents an addition or a subtraction operation.
3. An OR gate in the network represents a target or an intermediate constant and combines all possible implementations of the constant.
4. The outputs of the network are the OR gate outputs associated with the target constants.

The part of the algorithm, where the network is constructed, is given as follows:

1. Take an element from $T$, $t_i$.
2. For each pair in $O_i$, generate a two-input AND gate. The inputs of the AND gate are the elements of the pair, *i.e.*, '1' or the outputs of the OR gates representing the target and intermediate constants in the network.
3. Generate an OR gate associated with $t_i$ where its inputs are the outputs of AND gates determined in Step 2.
4. If $t_i$ is a target constant, assign the output of the corresponding OR gate as the output of the network.
5. Repeat Step 1 until all elements in $T$ are considered.

After the network is constructed, we include the optimization variables into the network so that the cost function of the 0-1 ILP problem can be formed. To do this, we associate the optimization variables with the operations. Hence, for each AND gate in the network, we add a third input representing an optimization variable.

As an example, suppose the depth 2 constant 215 as a target constant, that has the same implementations under the exact GB algorithm and the exact CSE algorithm when constants are defined under CSD or MSD. These implementations are $215 = 255 - 5 \ll 3$, $215 = 7 \ll 5-9$, and $215 = 31 \ll 3-33$. Also, note that there is only one possible implementation for each intermediate constant, 5, 7, 9, 31, 33, and 255, that are required to implement 215. The Boolean network generated for the target constant 215 after the optimization variables are added, is illustrated in Figure 6. In this figure, the 1-input OR gates for the intermediate constants are omitted and the type of each operation is given inside of each AND gate.

**Table 3: FIR filter specifications, size of 0-1 ILP problems generated by the exact algorithms, and their runtime.**

| Fil. | Filter Specifications | | | | 0-1 ILP Problem Size and Run Time | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | Exact CSE - CSD | | | | Exact CSE - MSD | | | | Exact GB | | | |
| | pass | stop | tap | width | vars | cons | ovars | sol / CPU | vars | cons | ovars | sol / CPU | vars | cons | ovars | sol / CPU |
| 1 | 0.20 | 0.25 | 30 | 14 | 1583 | 3129 | 728 | 23748 / 0.43 | 4172 | 8631 | 1947 | 22704 / 4.60 | 39483 | 82147 | 19271 | 21636 / 1860.2 |
| 2 | 0.15 | 0.20 | 30 | 14 | 1312 | 2557 | 608 | 26674 / 0.52 | 2962 | 6057 | 1373 | 23938 / 5.42 | 29668 | 62479 | 14370 | 22994 / 1321.3 |
| 3 | 0.10 | 0.15 | 30 | 14 | 1315 | 2621 | 599 | 25848 / 0.32 | 3590 | 7488 | 1653 | 24990 / 2.33 | 28927 | 61651 | 13971 | 23732 / 1843.5 |
| 4 | 0.15 | 0.20 | 40 | 14 | 817 | 1498 | 394 | 24952 / 0.40 | 2004 | 3782 | 972 | 24094 / 1.69 | 30846 | 63446 | 15050 | 23142 / 1343.1 |
| 5 | 0.10 | 0.12 | 40 | 14 | 1277 | 2509 | 586 | 34444 / 1.40 | 3161 | 6536 | 1465 | 32636 / 7.24 | 36650 | 76887 | 17834 | 30266 / 255.6 |
| 6 | 0.10 | 0.15 | 60 | 14 | 1137 | 2090 | 548 | 39288 / 0.30 | 2678 | 5134 | 1289 | 36214 / 4.39 | 43856 | 88195 | 21658 | 34346 / 292.8 |
| 7 | 0.15 | 0.20 | 80 | 14 | 1470 | 2819 | 689 | 42902 / 0.43 | 4745 | 9645 | 2220 | 41034 / 3.99 | 35710 | 72813 | 17454 | 39804 / 206.0 |
| 8 | 0.15 | 0.25 | 80 | 14 | 1539 | 2956 | 724 | 48922 / 0.38 | 2762 | 5386 | 1314 | 47916 / 1.12 | 52528 | 106072 | 25855 | 45418 / 371.9 |

## 3.3 The 0-1 ILP Formalization

The cost function of the 0-1 ILP problem is determined as the linear function of optimization variables representing operations, where the cost value of each optimization variable is the implementation cost of each operation determined as described in [3]. The constraints of the 0-1 ILP problem are obtained by finding the Conjunctive Normal Form (CNF) formulas of each gate in the network and expressing each clause in CNF formulas as a linear inequality as described in [5]. For example, a 3-input AND gate, $d = a \wedge b \wedge c$, is translated to CNF as $(a+\overline{d})(b+\overline{d})(c+\overline{d})(\overline{a}+\overline{b}+\overline{c}+d)$ and converted to linear constraints as $a-d \geq 0$, $b-d \geq 0$, $c-d \geq 0$, $-a-b-c+d \geq -2$. The outputs of the network, *i.e.*, the outputs of OR gates associated with the target constants, are also set to 1, since the implementation of target constants is aimed. Thus, the obtained model can serve as an input to a generic 0-1 ILP solver.

## 3.4 Finding the Minimum Solution

On the generated 0-1 ILP problem, a generic 0-1 ILP solver will search the minimum value of the cost function while satisfying the constraints that represent how target and intermediate constants are implemented. The solution of the 0-1 ILP solver, *i.e.*, the operations whose optimization variables are set to 1, will directly determine the set of operations that yields the minimum area solution.

## 3.5 Problem Complexity

As a worst case scenario, suppose that all the constants in depth 3 are to be implemented when *bw* is again in between 8 and 13 bits. In this case, all possible implementations of positive and odd constants in between $[3,\ 2^{bw}-1]$ should be considered in both algorithms. Table 2 presents the size of 0-1 ILP problems in terms of the number of variables (*vars*), constraints (*cons*), and optimization variables (*ovars*), for the exact CSE and GB algorithms in this scenario. Observe that although the current 0-1 ILP solvers can easily cope with all the 0-1 ILP problems generated by the exact CSE algorithm under MSD, they may find hard to handle those generated by the exact GB algorithm when *bw* is in between 11 and 13.

Note that the possible implementations of constants considered in the exact GB algorithm cover all possible implementations considered in the exact CSE algorithm under MSD. The same observation is also true for the exact CSE algorithm under MSD and CSD, since the MSD representations of a constant include its CSD representation. Although a CSE algorithm cannot guarantee the global minimum solution due to its limited search space, it can be applied on MCM instances that the exact GB algorithm cannot handle.

**Table 2: Worst-case complexity on the 0-1 ILP problem size.**

| bw | Exact CSE - MSD | | | Exact GB | | |
|---|---|---|---|---|---|---|
| | vars | cons | ovars | vars | cons | ovars |
| 8 | 2,047 | 4,495 | 960 | 6,169 | 13,172 | 3,021 |
| 9 | 5,855 | 12,351 | 2,800 | 44,517 | 90,298 | 22,131 |
| 10 | 17,935 | 36,767 | 8,712 | 227,169 | 456,082 | 113,329 |
| 11 | 49,071 | 99,207 | 24,024 | 843,593 | 1,689,356 | 421,285 |
| 12 | 132,815 | 266,543 | 65,384 | 2,490,401 | 4,983,112 | 1,244,177 |
| 13 | 338,943 | 677,839 | 167,424 | 6,260,249 | 12,522,174 | 3,128,077 |

# 4. EXPERIMENTAL RESULTS

This section presents high-level and gate-level results of the exact algorithms introduced in this paper and compares them with those of the algorithms designed for the MCM problem and the MCM problem under a delay constraint. In the design of an MCM operation at gate-level, first, we obtain a set of addition/subtraction operations that implements the constant multiplications using a high-level algorithm. Then, we describe these operations in VHDL under the ripple carry architecture as shown in [3] and synthesize the MCM circuit using the Cadence Encounter® RTL Compiler with the UMC Logic $0.18\mu m$ Generic II library under the minimum area design strategy during the technology mapping.

As an experiment set, we used FIR filter instances given in Table 3, where filter coefficients were computed with the *remez* algorithm in MATLAB. In this table, *pass* and *stop* are normalized frequencies that define the passband and stopband respectively, *tap* is the number of filter coefficients, and *width* is the bit-width of the coefficients. This table also presents the size of 0-1 ILP problems generated by the exact CSE and GB algorithms. Note that *sol* and *CPU* stand respectively for the minimum area value obtained by the 0-1 ILP solver [14] and its runtime in seconds on a PC with Intel Xeon at 2.33GHz and 4GB memory under Linux.

As can be observed from Table 3, since the exact GB algorithm is not restricted to any particular number representation, it generates a 0-1 ILP problem larger than that of the exact CSE algorithms, where the 0-1 ILP solver [14] requires more CPU time to find the minimum solution. The exact CSE algorithms obtain a solution using a little computational effort, but their solutions, especially those obtained under CSD, are far away from the global minima.

The high-level results of algorithms on FIR filters are presented in Table 4. In the algorithms designed for the MCM problem under a delay constraint, the delay constraint was taken as the minimum number of adder-steps of the MCM operation computed by Eqn. (2). In Table 4, *Op* and *GPS* denote respectively the number of operations and the power dissipation estimation value [8] of the MCM operation. In computation of the GPS value, the bit-width of the filter input was 16, as in the design of MCM operations at gate-level. Also, *As*/*Aas* indicates the maximum number of adder-steps and the average number of adder-steps over all operations in the MCM operation respectively.

Observe from Table 4 that the GB algorithms generally obtain better solutions in terms of the number of operations than those obtained by the exact CSE algorithms. Also, the solutions of the exact CSE and GB algorithms designed for the optimization of area problem under the minimum depth constraint generally include more number of operations, because the possible implementations of constants are restricted due to the minimum depth constraint and the optimization of area rather than the optimization of the number of operations is targeted in these algorithms. However, on overall instances, they find an MCM operation with the smallest *Aas* and *GPS* value with respect to the algorithms designed for the MCM problem and the MCM problem under a delay constraint.

**Table 4: Summary of high-level results of algorithms on FIR filter instances.**

| Fil. | Optimization of #Operations | | | | | | Opt. of #Operations under a Delay Constraint | | | | | | Opt. of Area under the Minimum Depth Constraint | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Exact CSE - MSD [1] | | | Exact GB [2] | | | Exact CSE - MSD [1] | | | Hcub-DC [15] | | | Exact CSE - MSD | | | Exact GB | | |
| | Op | As/Aas | GPS | Op | As/Aas | GPS | Op | As/Aas | GPS | Op | As/Aas | GPS | Op | As/Aas | GPS | Op | As/Aas | GPS |
| 1 | 22 | 4 / 2.59 | 1574 | 17 | 8 / 4.76 | 2807 | 22 | 3 / 2.27 | 1370 | 22 | 3 / 2.32 | 1698 | 22 | 3 / 2.27 | 1326 | 21 | 3 / 2.29 | 1358 |
| 2 | 22 | 4 / 2.27 | 1374 | 17 | 11 / 7.00 | 4191 | 22 | 3 / 2.09 | 1292 | 23 | 3 / 2.48 | 1745 | 23 | 3 / 2.04 | 1280 | 21 | 3 / 2.05 | 1303 |
| 3 | 23 | 4 / 2.30 | 1596 | 18 | 7 / 4.50 | 2505 | 23 | 3 / 2.09 | 1351 | 23 | 3 / 2.39 | 1780 | 23 | 3 / 2.09 | 1296 | 22 | 3 / 2.09 | 1408 |
| 4 | 23 | 4 / 2.17 | 1349 | 19 | 4 / 2.63 | 1556 | 23 | 3 / 2.09 | 1268 | 20 | 3 / 2.35 | 1433 | 24 | 3 / 1.96 | 1173 | 23 | 3 / 2.00 | 1232 |
| 5 | 29 | 3 / 2.28 | 1862 | 23 | 7 / 4.09 | 3420 | 29 | 3 / 2.24 | 1840 | 27 | 3 / 2.52 | 2195 | 31 | 3 / 2.10 | 1708 | 27 | 3 / 2.15 | 1739 |
| 6 | 33 | 4 / 2.30 | 2040 | 28 | 6 / 3.57 | 2944 | 33 | 3 / 2.24 | 1992 | 31 | 3 / 2.58 | 2438 | 35 | 3 / 2.14 | 1892 | 33 | 3 / 2.18 | 1945 |
| 7 | 40 | 4 / 2.33 | 2426 | 35 | 4 / 2.54 | 2501 | 40 | 3 / 2.15 | 2243 | 37 | 3 / 2.38 | 2452 | 42 | 3 / 2.02 | 2052 | 41 | 3 / 2.02 | 2085 |
| 8 | 47 | 3 / 2.21 | 2793 | 39 | 6 / 3.77 | 4614 | 47 | 3 / 2.17 | 2771 | 44 | 3 / 2.61 | 3499 | 48 | 3 / 2.10 | 2611 | 44 | 3 / 2.14 | 2562 |
| Avg. | 29.9 | 3.8 / 2.3 | 1876 | 24.5 | 6.6 / 4.1 | 3067 | 29.9 | 3 / 2.2 | 1765 | 28.4 | 3 / 2.5 | 2155 | 31 | 3 / 2.1 | 1667 | 29 | 3 / 2.1 | 1704 |

**Table 5: Summary of gate-level results of algorithms on FIR filter instances.**

| Fil. | Optimization of #Operations | | | | | | Opt. of #Operations under a Delay Constraint | | | | | | Opt. of Area under the Minimum Depth Constraint | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Exact CSE - MSD [1] | | | Exact GB [2] | | | Exact CSE - MSD [1] | | | Hcub-DC [15] | | | Exact CSE - MSD | | | Exact GB | | |
| | area | delay | power | area | delay | power | area | delay | power | area | delay | power | area | delay | power | area | delay | power |
| 1 | 12.2 | 6.0 | 1629 | 10.9 | 8.4 | 1548 | 11.5 | 6.7 | 1533 | 13.1 | 6.3 | 1783 | 11.2 | 6.7 | 1357 | 10.8 | 6.8 | 1496 |
| 2 | 11.8 | 6.4 | 1805 | 11.9 | 9.1 | 2072 | 11.6 | 6.4 | 1781 | 13.5 | 7.2 | 1921 | 11.6 | 6.4 | 1574 | 11.2 | 6.3 | 1724 |
| 3 | 12.2 | 6.0 | 1689 | 9.9 | 7.7 | 1534 | 11.3 | 6.0 | 1542 | 13.0 | 6.5 | 2014 | 10.8 | 6.2 | 1442 | 10.3 | 6.1 | 1411 |
| 4 | 11.5 | 5.5 | 1261 | 10.4 | 6.6 | 1360 | 11.1 | 6.0 | 1284 | 10.8 | 6.2 | 1310 | 10.7 | 5.9 | 1217 | 10.6 | 5.5 | 1183 |
| 5 | 16.9 | 6.7 | 2883 | 15.5 | 8.1 | 3133 | 16.8 | 7.0 | 2673 | 16.4 | 7.6 | 2862 | 16.3 | 6.8 | 2446 | 15.5 | 6.8 | 2599 |
| 6 | 19.1 | 6.9 | 3720 | 17.2 | 9.0 | 4427 | 19.0 | 6.8 | 3466 | 19.8 | 7.2 | 4292 | 18.4 | 6.7 | 3076 | 17.8 | 7.4 | 2728 |
| 7 | 21.6 | 6.8 | 3010 | 20.1 | 6.7 | 2957 | 21.0 | 6.6 | 2734 | 20.4 | 6.5 | 2891 | 20.3 | 6.3 | 2721 | 19.9 | 6.5 | 2588 |
| 8 | 25.6 | 6.9 | 6692 | 26.1 | 9.1 | 7704 | 25.8 | 7.2 | 6647 | 26.0 | 7.8 | 6823 | 24.7 | 7.1 | 6106 | 23.7 | 7.1 | 6168 |
| Avg. | 16.4 | 6.4 | 2836.1 | 15.3 | 8.1 | 3091.9 | 16.0 | 6.6 | 2707.5 | 16.6 | 6.9 | 2987.0 | 15.5 | 6.5 | 2492.4 | 15.0 | 6.6 | 2487.1 |

The gate-level results of MCM designs that are obtained using the solutions of high-level algorithms given in Table 4 are presented in Table 5. In this table, *area* ($mm^2$), *delay* ($ns$), and *power* ($\mu W$) denote the area, delay, and power dissipation results of MCM designs at gate-level respectively. Power dissipation values were obtained using simulation results under 10,000 random input vectors.

Although the exact GB algorithm [2] finds an MCM design including the minimum number of operations as presented in Table 4, its solution does not always yield an MCM design with the optimal area as can be observed on Filters 1, 2, 7, and 8 in Table 5 when its results are compared with the exact GB algorithm proposed in this paper. Also, although the GB Hcub-DC algorithm can find an MCM with the minimum delay constraint as shown in Table 4, its solutions does not always lead an MCM design with the minimum delay as can be observed on Filters 2-5 and 8 in Table 5 when its results are compared with the exact GB algorithm introduced in this paper. On average, it obtains better solutions on area and delay than these algorithms. The same observation can be also made on the exact CSE algorithms. This is simply because the exact CSE and GB algorithms proposed in this paper take into account the implementation of an addition/subtraction operation at gate-level while synthesizing each constant at its minimum depth. Furthermore, this approach enables the proposed exact algorithms to find MCM designs that consume less power with respect to the MCM designs obtained by the high-level algorithms designed for the MCM problem and the MCM problem under a delay constraint.

# 5. CONCLUSIONS

This paper addressed the problem of optimization of gate-level area in the MCM operation where each constant is implemented at its minimum depth and introduced the 0-1 ILP formalization of this problem. It was shown that the proposed exact algorithms achieve significant improvements not only on power dissipation but also, on area and delay of the MCM design, when compared to the solutions obtained by the prominent algorithms designed for the MCM problem and the MCM problem under a delay constraint.

# 6. ACKNOWLEDGMENT

This work was supported by the *Portuguese Foundation for Science and Technology* (FCT) research project *Multicon - Architectural Optimization of DSP Systems with Multiple Constants Multiplications* and by FCT through the PIDDAC Program funds.

# 7. REFERENCES

[1] 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 TCAD*, 27(6):1013-1026, 2008.

[2] L. Aksoy, E. Gunes, and P. Flores. Search Algorithms for the Multiple Constant Multiplications Problem: Exact and Approximate. *Elsevier Journal on Microprocessors and Microsystems*, 34:151-162, 2010.

[3] L. Aksoy, E. Costa, P. Flores, and J. Monteiro. Optimization of Area and Delay at Gate-Level in Multiple Constant Multiplications. In *Euromicro Conference on Digital System Design*, pages 3-10, 2010.

[4] A. Avizienis. Signed-digit Number Representation for Fast Parallel Arithmetic. *IRE Transactions on Electronic Computers*, EC-10:389-400, 1961.

[5] P. Barth. A Davis-Putnam Based Enumeration Algorithm for Linear Pseudo-Boolean Optimization. *Technical Report Max-Planck-Institut Fur Informatik*, 1995.

[6] P. Cappello and K. Steiglitz. Some Complexity Issues in Digital Signal Processing. *IEEE Tran. on Acoustics, Speech, and Signal Processing*, 32(5):1037-1041, 1984.

[7] J. Chen, C.-H. Chang, and H. Qian. New Power Index Model for Switching Power Analysis from Adder Graph of FIR Filter. In *ISCAS*, pages 2197-2200, 2009.

[8] S. Demirsoy, A. Dempster, and I. Kale. Power Analysis of Multiplier Blocks. In *ISCAS*, pages 297-300, 2002.

[9] M. Ercegovac and T. Lang. Digital Arithmetic. *Morgan Kaufmann*, 2003.

[10] R. Hartley. Subexpression Sharing in Filters using Canonic Signed Digit Multipliers. *IEEE TCAS II*, 43(10):677-688, 1996.

[11] H-J. Kang and I-C. Park. FIR Filter Synthesis Algorithms for Minimizing the Delay and the Number of Adders. *IEEE TCAS II*, 48(8):770-777, 2001.

[12] H. Nguyen and A. Chatterjee. Number-Splitting with Shift-and-Add Decomposition for Power and Hardware Optimization in Linear DSP Synthesis. *IEEE Tran. on VLSI*, 8(4):419-424, 2000.

[13] I-C. Park and H-J. Kang. Digital Filter Synthesis Based on Minimal Signed Digit Representation. In *DAC*, pages 468-473, 2001.

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

[15] Spiral webpage. http://spiral.ece.cmu.edu/mcm/gen.html.

[16] Y. Voronenko and M. Püschel. Multiplierless Multiple Constant Multiplication. *ACM Tran. on Algorithms*, 3(2), 2007.