Optimization of Area in Digit-Serial Multiple Constant Multiplications at Gate-Level

Levent Aksoy and Cristiano Lazzari
INESC-ID
Lisboa, Portugal

Eduardo Costa
Universidade Católica de Pelotas
Pelotas, Brazil

Paulo Flores and José Monteiro
INESC-ID/IST TU Lisbon
Lisboa, Portugal

***Abstract*— The last two decades have seen many efficient algorithms and architectures for the design of low-complexity bit-parallel Multiple Constant Multiplications (MCM) operation, that dominates the complexity of Digital Signal Processing (DSP) systems. On the other hand, digit-serial architectures offer alternative low-complexity designs, since digit-serial operators occupy less area and are independent of the data wordlength. This paper introduces the problem of designing a digit-serial MCM operation with minimal area at gate-level and presents the exact formalization of the area optimization problem as a 0-1 Integer Linear Programming (ILP) problem. Experimental results show the efficiency of the proposed algorithm and digit-serial MCM designs in terms of area at gate-level.**

I. Introduction

Multiplication of a variable with a set of constants, known also as the MCM operation, is a central operation and performance bottleneck in many DSP applications such as, error correcting codes, linear DSP transforms, and Finite Impulse Response (FIR) filters. In hardware, the multiplication operation is considered to be expensive, as it occupies significant area. Hence, constant multiplications are generally realized using only addition, subtraction, and shift operations [1].

For the bit-parallel design of the MCM operation, the MCM problem is defined as finding the fewest number of addition and subtraction operations that realize the MCM, since shifts can be implemented using only wires in hardware. Many efficient algorithms [2]–[5] have been introduced for the MCM problem. In spite of various methods they use and different search space they explore, the main idea has always been the maximization of the sharing of common partial products among the constant multiplications. As an example, consider the constant multiplications $29x$ and $43x$. Observe from Figure 1(a)-(b) that the sharing of partial products $3x$ and $5x$ reduces the number of operations from 6 to 4.

On the other hand, in digit-serial arithmetic, the words are divided into digit sets of d bits that are processed one at a time [6]. The special cases of the digit-serial computation are called bit-parallel and bit-serial processing when the digit size d is equal to data wordlength and 1 respectively. The digit-serial computation plays an important role when the bit-serial implementations cannot meet delay requirements and the bit-parallel designs require excessive hardware. Thus, an optimal tradeoff between area and delay can be obtained by changing the digit size parameter (d).

This work was partially supported by the *Portuguese Foundation for Science and Technology* (FCT) research project PTDC/EIA-EIA/103532/2008.

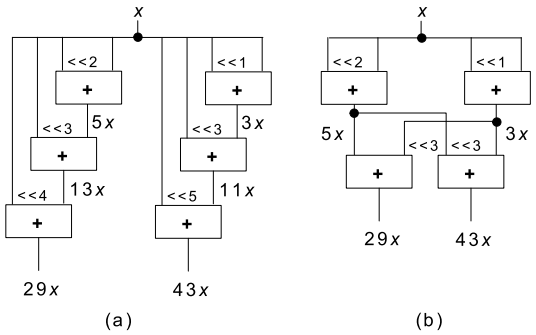

Fig. 1. Shift-adds implementations of $29x$ and $43x$ (a) without partial product sharing; (b) with partial product sharing.

The digit-serial MCM operation in shift-adds architecture consists of digit-serial addition and subtraction operations, and D flip-flops for the shift operations, as opposed to the bit-parallel MCM operation, where shifts are free in terms of hardware. The high-level algorithm of [7] aims to find a solution with the fewest number of additions, subtractions, and shift operations. To the best of our knowledge, there exists no algorithm that focuses on the minimization of area in digit-serial MCM operation at gate-level.

In this paper, we initially present the digit-serial realizations of addition, subtraction, and shift operations and determine their implementation costs in terms of gate-level metrics. Then, we introduce an exact algorithm that optimizes the area of the digit-serial MCM operation at gate-level. In the exact algorithm, all possible implementations of constant multiplications are found when constants are defined under a number representation and the optimization of area problem is formulated as a 0-1 ILP problem taking into account the hardware cost of each operation. Thus, the maximization of sharing of partial products and also the amount of shifts is realized based on a gate-level implementation. The experimental results indicate that the exact algorithm leads to low-complexity digit-serial MCM designs compared to those found by the exact algorithm of [3] designed for the MCM problem and those that are implemented using generic digit-serial multipliers [6].

II. Digit-Serial Operations

The fundamental digit-serial operations have been introduced in [8]. Figure 2 illustrates the digit-serial addition, subtraction, and left shift operations when d is 3. Notice from Figure 2(a) that in a digit-serial addition operation, the number of required full adders (FAs) is equal to d and the number of necessary D flip-flops is 1 in general. Also, the

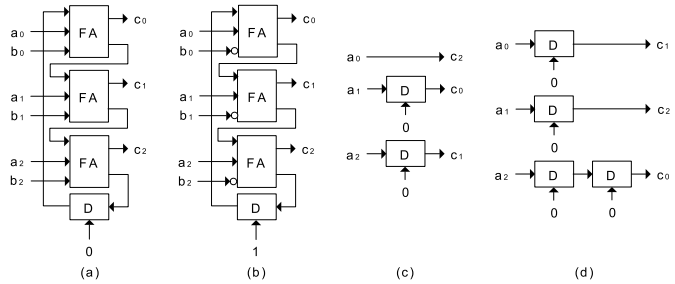

Fig. 2. The digit-serial operations when d is equal to 3: (a) addition operation; (b) subtraction operation; (c) left shift by 2 times; (d) left shift by 4 times.

subtraction operation (Figure 2(b)) is implemented using 2's complement, requiring the initialization of the D flip-flop with 1 and additional d inverter gates with respect to the digit-serial addition operation. In a left shift operation (Figure 2(c)-(d)), the number of required D flip-flops is equal to the amount of shift and is not related with d. Thus, the problem of optimization of area in digit-serial MCM operation can be defined as: given the target constants to be implemented and the digit size d, find a set of operations that implements the constant multiplications yielding minimum area at gate-level.

III. The Exact Algorithm

To design a digit-serial MCM operation with minimal area at gate-level, we consider the implementation cost of digit-serial addition, subtraction, and also the shift operations under the given digit size d. The exact algorithm consists of four main steps. Firstly, all possible implementations of constants are extracted from the non-zero digits of the constants defined under a number representation namely, binary, Canonical Signed Digit (CSD), or Minimal Signed Digit (MSD). Then, the implementations of constants are represented in a network. Thirdly, the area optimization problem is formalized as a 0-1 ILP problem with a cost function to be minimized and a set of constraints to be satisfied. Finally, a set of operations that yields the minimum area solution is obtained using a generic 0-1 ILP solver. These four steps are described in detail next.

A. Finding the Partial Terms

In the preprocessing phase, the constants to be multiplied by a variable are converted to positive and then, made odd by successive divisions by 2. The resulting constants are stored without repetition in the target set T. Thus, T includes the minimum number of necessary constants to be implemented. The part of the algorithm, where the implementations of the target constants and partial terms are found, is as follows:

1) Take an element from T, t_i, find its representation(s) under the given number representation, and store it(them) in a set called S. Form an empty set, O_i associated with t_i, that will include the inputs and the amount of left shifts at the inputs of all addition/subtraction operations which generate t_i.
2) For each representation of t_i in the set S,
   a) Compute all non-symmetric partial term pairs that cover the representation of t_i.
   b) Make each partial term positive and odd, and determine its amount of left shift.

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

Fig. 3. Possible implementations of 25 under MSD representation.

   c) Add each partial term pair and their amount of left shifts to the set O_i.
   d) Add each partial term to T, if it does not represent the input that the constants are multiplied with, *i.e.*, denoted by 1, and is not in T.
3) Repeat Step 1 until all elements of T are considered.

Observe that the target set T, that only includes the target constants to be implemented in the beginning of the iterative loop, is augmented with the partial terms that are required for the implementation of target constants. We note that all possible implementations of an element in the target set, t_i, are found by decomposing the non-zero digits in the representation of t_i into two partial terms. As an example, consider 25 as a target constant defined under MSD, that has two representations, 011001 and $10\bar{1}001$, where $\bar{1}$ stands for -1. All possible implementations of 25 are given in Figure 3.

Observe from Figure 3 that the last implementations of 25 on both representations, *i.e.*, $1 + 3 \ll 3$, are identical thus, one of them can be eliminated. Also, the duplications of implementations such as, $1 \ll 4 + 9 = 9 + 1 \ll 4$, are not listed in Figure 3. After the partial terms required for the implementation of 25 under MSD, *i.e.*, 3, 7, 9, 17, and 33, are found, they are added to the target set without repetition and their implementations are determined in a similar way.

B. Construction of the Boolean Network

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

1) The primary input of the network is the input variable to be multiplied with the constants.
2) An AND gate in the network represents an addition/subtraction operation and has two inputs.
3) An OR gate in the network represents a target constant or a partial term and combines all its possible implementations.
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 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, denoting the input that the constants are multiplied with, or the outputs of OR gates representing target constants and partial terms in the network.

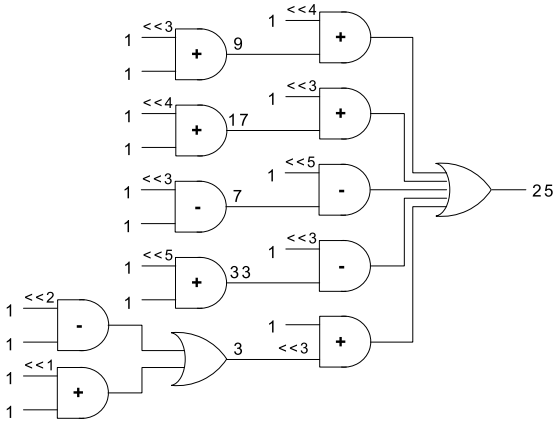

Fig. 4. The network constructed for the target constant 25 under MSD.

3) Generate an OR gate associated with t_i where its inputs are the outputs of the AND gates determined in Step 2.
4) If t_i is a target constant, assign the output of the corresponding OR gate as an output of the network.
5) Repeat Step 1 until all elements in T are considered.

The network generated for the target constant 25 defined under MSD is given in Figure 4, where 1-input OR gates for the partial terms 7, 9, 17, and 33 are omitted, and the type of each operation is shown inside of each AND gate. Observe that the network represents which operations are required to implement the constants.

C. Formalization of the 0-1 ILP Problem

We have to include optimization variables to the network in order to formalize the gate-level area optimization problem as a 0-1 ILP problem. The optimization variables are associated with the operations and the left shifts of the constants. For each AND gate that represents an operation in the network, we add an additional input standing for the optimization variable associated with the operation, *i.e.*, $opt_{a\pm b}$, where a and b denote the inputs of an operation. In the cost function to be minimized, the cost value of this type of optimization variable is determined as the implementation cost of the digit-serial addition/subtraction operation at gate-level as described in Section II. In order to maximize the sharing of left shifts, *i.e.*, the D flip-flops at gate-level, for each constant c in the network, we initially find the maximum amount of left shifts, mls_c, that a constant has. Then, for each constant c that has the maximum amount of left shift mls_c greater than zero, we introduce mls_c optimization variables representing left shifts of c from 1 to mls_c, *i.e.*, $opt_{c\ll 1}, opt_{c\ll 2}, \ldots, opt_{c\ll mls_c}$. Thus, for each AND gate in the network, we include ls additional inputs standing for the optimization variables associated with the left shifts of an input signal of the operation. In the cost function to be minimized, the cost value of each of this optimization variable is determined as the implementation cost of one D flip-flop, as described in Section II.

Some simplifications on this network can be also achieved. Note that the variable denoted by 1, representing the MCM input signal, can be eliminated from the inputs of the AND gates, because its logic value is always one (*i.e.*, it is always available). The network generated for the target constant 25 under MSD is depicted in Figure 5 after the optimization variables are added and simplifications are done.

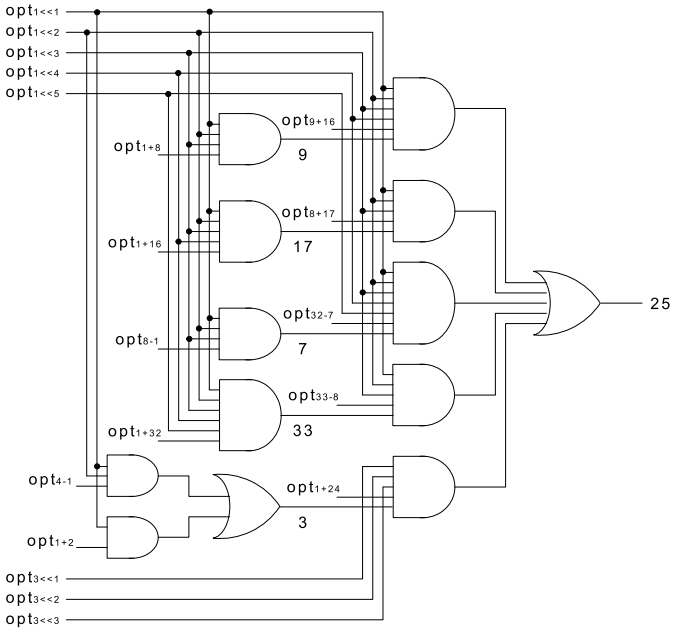

Fig. 5. The Boolean network constructed for the target constant 25 under MSD after optimization variables are added.

The cost function of the 0-1 ILP problem is constructed as the linear function of optimization variables, where the cost value of each optimization variable is determined as described above. 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 of the CNF formulas as a linear inequality, as described in [9]. Also, the outputs of the network are set to 1, since the implementation of target constants is required.

D. Finding the Minimum Area Solution

A generic 0-1 ILP solver will search the minimum value of the cost function on the generated 0-1 ILP problem by satisfying the constraints that represent how target constants and partial terms are implemented. Note that the set of operations that yields the minimum area solution is obtained directly from the solution of the 0-1 ILP solver, *i.e.*, the operations whose optimization variables are set to 1.

IV. Experimental Results

In the design of a digit-serial MCM operation at gate-level, we initially find a set of addition/subtraction operations that realizes the constant multiplications using a high-level algorithm. Given digit size d, then, each operation is translated into a digit-serial addition/subtraction operation considering the left shift of an input in VHDL with the necessary control and storage logic. Finally, the digit-serial MCM operation is designed using the Cadence Encounter® RTL Compiler with the UMCLogic 0.18μm Generic II library.

As a working example, we used the FIR filter instance denoted as Filter 9 in [3]. Table I presents the high-level results

TABLE I

HIGH-LEVEL RESULTS OF DIGIT-SERIAL MCM OPERATIONS.

Digit size	1				2				4				8			
Algorithm	adder	shift	HCAS	HCS	adder	shift	HCAS	HCS	adder	shift	HCAS	HCS	adder	shift	HCAS	HCS
Proposed Exact	57	32	8232	1664	54	42	12780	2444	50	65	21008	3380	49	75	38596	3900
[3]	49	94	7126	4888	49	94	11704	4888	49	94	20860	4888	49	94	39172	4888

TABLE II

GATE-LEVEL RESULTS OF DIGIT-SERIAL MCM DESIGNS.

Arch. Type	Digit size	1			2			4			8			16		
	Des. str.	area	delay	power	area	delay	power	area	delay	power	area	delay	power	area	delay	power
Shift-adds	MA	34158	7332	7.1	36291	7067	9.3	39987	7071	10.1	47238	6385	14.3	49566	8404	21.6
	MAuMCF	38722	1400	60.3	39887	1601	57.2	43574	1840	57.1	49855	2481	55.5	53300	4400	44.6
Generic Imp. [6]	MA	37878	5605	8.3	48375	7336	10.1	51154	7322	12.0	65480	7332	16.9	59434	5470	20.0
	MAuMCF	43503	1250	64.0	58430	1350	73.4	60777	1450	71.6	75360	1850	68.9	61606	3960	40.7

of the proposed exact algorithm when d is 1, 2, 4, and 8, where *adder* and *shift* denote the number of operations and shifts, respectively. Also, *HCAS* and *HCS* represent the hardware cost of the digit-serial addition/subtraction operations and shifts respectively. The implementation cost (in μm^2) of an FA, a D flip-flop, and an inverter was taken as 90, 52, and 6 respectively from the library. This table also gives the high-level results on digit-serial implementations of the MCM operations optimized by the algorithm of [3] designed for the optimization of the number of addition/subtraction operations in the bit-parallel MCM design. In this experiment, the bit-width of the filter input was taken as 16 and constants were defined under MSD in both algorithms.

Observe from Table I that since the algorithm of [3] focuses on the optimization of the number of operations, its solution may yield a digit-serial MCM design that includes a large number of left shifts which are realized using D flip-flops in digit-serial designs. On the other hand, the proposed algorithm finds a solution with the minimum area at gate-level by maximizing the sharing of partial products and shifts, taking into account the cost of digit-serial operations at gate-level.

Table II presents the gate-level results of the digit-serial MCM designs, where *area*, *delay*, and *power* denote respectively the area in μm^2, the delay of the longest path in *ps* (the minimum clock period), and the power dissipation in *mW*. The gate-level results obtained by the proposed algorithm are given in the *Shift-adds* row when d is in between 1 and 8. In this row, for the bit-parallel implementation, *i.e.*, when d is equal to 16, the solution of [3] was synthesized at gate-level. The results given in the *Generic Imp.* row, where d is in between 1 and 8, were obtained when each constant multiplication of the MCM operation is defined using a digit-serial multiplier [6]. The results in this row when d is 16 were found when the constant multiplications are described using parallel constant multipliers in VHDL. During the technology mapping, the MCM operations were designed under two design strategies, *i.e.*, the minimum area (*MA*) and the minimum area under the maximum clock frequency (*MAuMCF*) constraint. In the former, there was no constraint on the clock frequency and in the latter, we iteratively found the maximum clock frequency that can be applied to the digit-serial MCM design.

Observe from the *Shift-adds* row of Table II that as the digit-size increases, the area of the MCM design and the minimum value of the clock period increases. Note that the number of clock cycles to obtain the actual constant multiplications is 32, 16, 8, and 4 when d is 1, 2, 4, and 8 respectively. In bit-parallel implementation, only one clock cycle is required. Also, observe from Table II that the design of digit-serial MCM operations in shift-adds architecture, where the addition, subtraction and also shift operations are shared, yields low-complexity and low-power digit-serial MCM designs when compared to those that are implemented using generic digit-serial multipliers [6]. Note that under the generic implementation of the bit-parallel MCM operation, the synthesis tool has more possibilities to optimize area with respect to the generic implementations of digit-serial MCM designs.

V. CONCLUSIONS

In this paper, we introduced a 0-1 ILP formalization for designing a digit-serial MCM operation with minimal area at gate-level by considering the implementation costs of digit-serial addition, subtraction, and shift operations. Although the search space of the exact algorithm is restricted by the number representation, the given 0-1 ILP formalization can be applied in algorithms that are not limited to any particular number representation. The experimental results indicate that the complexity of the MCM design can be further reduced using digit-serial architectures and the area and delay tradeoff can be explored by changing the digit-size.

REFERENCES

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

[2] R. Hartley, "Subexpression Sharing in Filters using Canonic Signed Digit Multipliers," *IEEE TCAS II*, vol. 43, no. 10, pp. 677–688, 1996.

[3] 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*, vol. 27, no. 6, pp. 1013–1026, 2008.

[4] A. Dempster and M. Macleod, "Use of Minimum-Adder Multiplier Blocks in FIR Digital Filters," *IEEE TCAS II*, vol. 42, no. 9, pp. 569–577, 1995.

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

[6] R. Hartley and K. Parhi, *Digit-Serial Computation*. Kluwer Academic Publishers, 1995.

[7] K. Johansson, O. Gustafsson, and L. Wanhammar, "Multiple Constant Multiplication for Digit-Serial Implementation of Low Power FIR Filters," *WSEAS Transactions on Circuits and Systems*, vol. 5, no. 7, pp. 1001–1008, 2006.

[8] R. Hartley and P. Corbett, "Digit-Serial Processing Techniques," *IEEE TCAS*, vol. 37, no. 6, pp. 707–719, 1990.

[9] P. Barth, "A Davis-Putnam Based Enumeration Algorithm for Linear Pseudo-Boolean Optimization," Max-Planck-Institut Fur Informatik, Tech. Rep., 1995.