Design of Low-Complexity and High-Speed Digital Finite Impulse Response Filters

Diego Jaccottet and Eduardo Costa
Universidade Catolica de Pelotas
Pelotas-RS, Brazil

Levent Aksoy
INESC-ID
Lisbon, Portugal

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

***Abstract*—In this paper, we introduce a design methodology to implement low-complexity and high-speed digital Finite Impulse Response (FIR) filters. Since FIR filters suffer from a large number of constant multiplications, in the proposed method the constant multiplications are replaced by addition/subtraction and shift operations. Also, based on the design objective, *i.e.*, low-complexity or high-speed, the addition/subtraction operations are implemented using Ripple Carry Adder (RCA) or Carry-Save Adder (CSA) architectures respectively. Furthermore, high-level algorithms designed for the optimization of the number of RCA and CSA blocks are used to reduce the complexity of the FIR filter. Thus, a Computer-Aided Design (CAD) tool that synthesizes low-complexity and high-speed FIR filters in a shift-adds architecture is developed. It is observed from the experimental results on FIR filter instances that the developed CAD tool can find better FIR filter designs in terms of area and delay than those obtained using efficient general multipliers.**

I. Introduction

Finite Impulse Response (FIR) filters are widely used in Digital Signal Processing (DSP) applications due to their stability and linear-phase property. The multiplier block of the FIR filter, where the multiplications of filter coefficients by the filter input are realized as illustrated in Figure 1, has significant impact on the complexity and performance of the design, since a large number of constant multiplications are required. This operation is generally known as the Multiple Constant Multiplications (MCM) and is also a central operation and performance bottleneck in many DSP systems, such as Infinite Impulse Response (IIR) filters, Fast Fourier Transforms (FFT), and Discrete Cosine Transforms (DCT).

Although area, delay, and power efficient multiplier architectures, such as Wallace [1], modified Booth [2], and radix-2^m binary array [3] multipliers, have been proposed, the full-flexibility of a multiplier is not necessary for the constant multiplications, since filter coefficients are determined beforehand by the DSP algorithms. Hence, the multiplication of filter coefficients with the filter input is generally implemented under a shift-adds architecture [4], where each constant multiplication is realized using addition/subtraction and shifting operations.

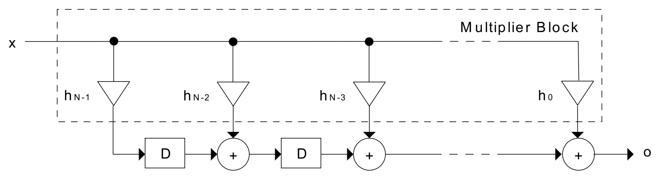

Fig. 1. Transposed form of a digital FIR filter implementation.

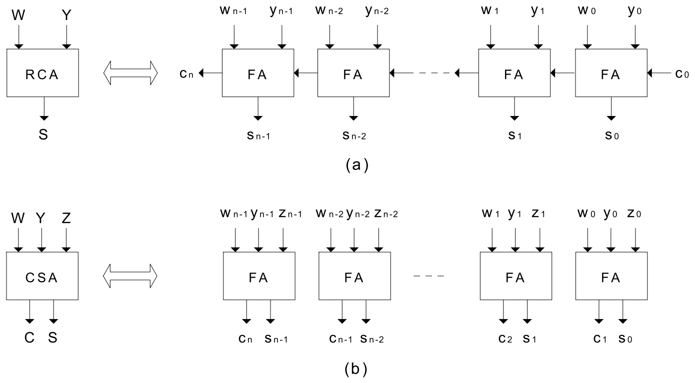
W Y
RCA
S
w_{n-1} y_{n-1} w_{n-2} y_{n-2} w_1 y_1 w_0 y_0
c_n FA FA FA FA c_0
s_{n-1} s_{n-2} s_1 s_0
(a)
W Y Z
CSA
C S
w_{n-1} y_{n-1} z_{n-1} w_{n-2} y_{n-2} z_{n-2} w_1 y_1 z_1 w_0 y_0 z_0
FA FA FA FA
c_n s_{n-1} c_{n-1} s_{n-2} c_2 s_1 c_1 s_0
(b)

Fig. 2. Addition architectures: (a) Ripple carry adder; (b) Carry-save adder.

Furthermore, the implementation of constant multiplications in a shift-adds architecture enables high-level algorithms [5]–[10] to obtain significant reductions in area and power dissipation of the MCM design by sharing the common partial products among the constant multiplications. In these algorithms, the optimization problem is defined as finding the minimum number of addition/subtraction operations that implement the constant multiplications, since shifts can be implemented using only wires in hardware without representing any area cost. This optimization problem is known as the MCM problem and has been proven to be an NP-complete problem in [11].

In high-level algorithms designed for the MCM problem, an adder/subtracter is assumed to be a two-input operation that is generally implemented using a Ripple Carry Adder (RCA) block yielding small area but great latency in the design of the MCM operation. On the other hand, in high-speed DSP applications, Carry-Save Adder (CSA) blocks are preferred to RCA blocks in spite of the increase in area. The CSA block has three inputs and two outputs, Sum (S) and Carry (C), that together form the result. An n-bit CSA block includes n Full Adders (FAs). Since there is no need to propagate the carry as required in an RCA block, Figure 2(a), the latency of addition is equal to the gate delay of a full adder, independently of the data wordlength, Figure 2(b). Note that to obtain the final sum, a Vector Merging Adder (VMA) that can be implemented using a faster adder such as Kogge-Stone adder [12] is required.

The digital FIR filter proposed for high-speed DSP applications in [13] is illustrated in Figure 3. In this figure, each addition represents a CSA block and the filter output is obtained using only one VMA. Also, each constant multiplication is

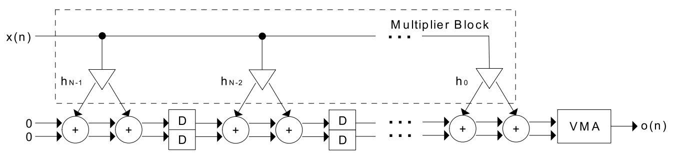

Fig. 3. Transposed form of a digital high-speed FIR filter implementation.

implemented in the shift-adds architecture using CSA blocks. The high-level algorithms [14]–[16], that aim to maximize the sharing of partial products generated by the CSA blocks, are used to find a solution with the fewest number of CSA blocks.

In this paper, we introduce a Computer-Aided Design (CAD) tool that is capable of designing low-complexity and high-speed digital FIR filters. The CAD tool initially obtains the fewest number of addition/subtraction operations that are required to implement the MCM operation using a high-level algorithm based on the design objective, *i.e.*, low-complexity or high-speed. For the low-complexity FIR filter design, a high-level algorithm designed for the optimization of the number of operations is used to find the fewest number of RCA blocks solution in the multiplier block of the FIR filter. For the high-speed FIR filter design, the complexity of the multiplier block is optimized by finding the fewest number of CSA blocks solution with a high-level algorithm. Then, the CAD tool converts the multiplier block optimized under the RCA or CSA architectures into a synthesizable description format and describes the FIR filter at gate-level using this optimized multiplier block by including additional circuits. Finally, it uses the Sequential Interactive System (SIS) tool [17] as a synthesizer to map the gate-level description of the FIR filter in a sequential circuit using a gate library. In this paper, we present the CAD tool developed for the design of FIR filters under RCA and CSA architectures and compare its results with those of the FIR filters whose multiplier blocks are designed using an efficient radix-8 array multiplier [3] and a parallel multiplier generated by the Mullet tool [18]. It is observed that the design of an FIR filter in a shift-adds architecture using RCA blocks yields the best area results. On the other hand, the implementation of an FIR filter in a shift-adds architecture using CSA blocks leads to high-speed FIR filter designs with respect to those whose multiplier blocks are realized using efficient multipliers and FIR filters designed using RCA blocks. It is also observed that the use of high-level algorithms that optimize the number of RCA or CSA blocks in MCM reduces the complexity of the FIR filter.

The rest of the paper is organized as follows. Section II describes the high-level algorithms designed for the optimization of the number of RCA and CSA blocks in the multiplier block of the digital FIR filter. The CAD tool is introduced in Section III and experimental results are given in Section IV. Finally, the paper is concluded with Section V.

II. The High-level Algorithms

This section presents an overview on the algorithms proposed for the problems of optimization of the number of RCA and CSA blocks in the multiplier block of a digital FIR filter given in Figure 1 and 3 respectively.

A. Optimization of the Number of RCA Blocks

For the implementation of constant multiplications using addition/subtraction and shift operations, a straightforward way, generally known as the digit-based recoding method [19], initially defines the constants in multiplications in binary representation. 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, suppose the constant multiplications of 29 and 43 by the variable x, *i.e.*, $29x$ and $43x$. The decompositions of constant multiplications are given as follows:

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

where 6 addition operations are required as illustrated in Figure 4(a). To further improve the solution, one can also define the constants under Canonical Signed Digit (CSD) representation, where each constant has a unique representation with the minimum number of non-zero digits [20].

However, the digit-based recoding techniques do not consider the sharing of partial products among the constant multiplications that may yield an MCM design with less number of operations. The algorithms that aim to maximize the partial product sharing in MCM can be categorized in two classes: Common Subexpression Elimination (CSE) algorithms [5]–[7] and graph-based (GB) methods [8]–[10]. The CSE algorithms, also referred to as the pattern search methods, initially define the constants under a number representation, namely binary, CSD, or Minimal Signed Digit (MSD), and then, recursively find the "best" subexpression, generally the most common, among the constant multiplications. The exact CSE algorithm [7], that formulates the MCM problem as a 0-1 Integer Linear Programming (ILP) problem, first finds all possible implementations of the constant multiplications when constants are defined under a particular number representation and then, obtains the minimum number of operations solution using a generic 0-1 ILP solver. Returning to our example, the exact CSE algorithm of [7] identifies the most common partial products, $3x = (11)_{bin}x$ and $5x = (101)_{bin}x$, in both multiplications when the constants are defined in binary, obtaining a solution with 4 operations as given in Figure 4(b).

On the other hand, the GB algorithms are not limited to any particular number representation and consider a large number of alternative implementations of a constant multiplication, yielding better solutions than the CSE algorithms as shown in [9], [10]. The prominent GB heuristics [8], [9] consist of optimal and heuristic parts. In the optimal part, the constant multiplications that can be realized using a single operation are synthesized. If there are still constant multiplications to be implemented, then these algorithms switch to their heuristic parts, where in each iteration, a constant multiplication is synthesized using a single operation including a partial product. The exact GB algorithms that search the minimum number of operations solution in breadth-first and depth-first manners were proposed in [10]. For our example, the exact GB algorithm [10] finds

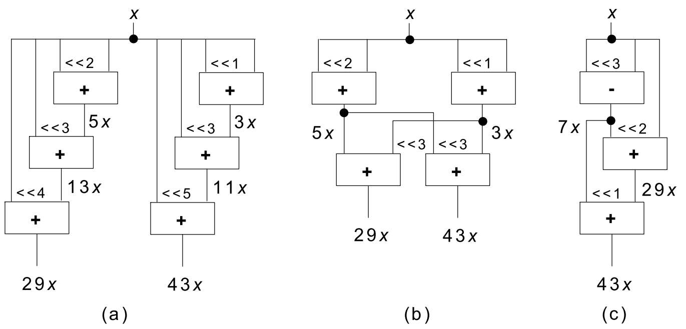

Fig. 4. Implementation of $29x$ and $43x$ using RCA blocks obtained with: (a) a digit-based recoding technique [19]; (b) the exact CSE algorithm [7]; (c) the exact GB algorithm [10].

the minimum number of operations solution with 3 operations as presented in Figure 4(c).

Note that optimizing the number of operations in MCM directly corresponds to finding a solution with the minimum number of RCA blocks, since each addition/subtraction operation is synthesized using a single RCA block.

B. Optimization of the Number of CSA Blocks

In many designs, particularly in DSP systems, performance is an important and a crucial parameter. Hence, circuit area is generally expandable in order to achieve a given performance target. In this work, the delay of the FIR filter design is reduced by using CSA blocks for the implementation of constant multiplications in the multiplier block and for the implementation of addition operations used in computation of the filter output as illustrated in Figure 3.

The direct way of implementing constant multiplications with CSA blocks is again to apply a digit-based recoding method [19]. Returning to our constant multiplications $29x$ and $43x$, the representation of constants in binary leads to a solution with 4 CSA blocks[1] as given in Figure 5(a). The prominent algorithms proposed for the problem of finding the fewest number of CSA blocks in MCM can be again categorized in two classes as CSE and GB algorithms. The CSE heuristic of [15] initially defines the constant multiplications in expressions and then, iteratively extracts all possible three-term divisors from the expressions, finds the best divisor, *i.e.*, the most common divisor, among these divisors, and redefines the expressions by replacing the best divisor in the expressions with two terms, *i.e.*, sum and carry outputs of a CSA block. The exact CSE algorithm of [16] formalizes the problem as a 0-1 ILP problem and finds the minimum number of CSA blocks solution when the possible CSA implementations of a constant multiplication are extracted from the number representation of the constant. For the implementation of constant multiplications $29x$ and $43x$, the exact CSE algorithm [16] finds a solution with 3 CSA blocks as illustrated in Figure 5(b) when constants are defined under binary representation.

An approximate algorithm that considers more possible implementations of the constant multiplications using the general

[1]Recall that a CSA block has three inputs and two outputs, S and C, and these outputs together indicate the result.

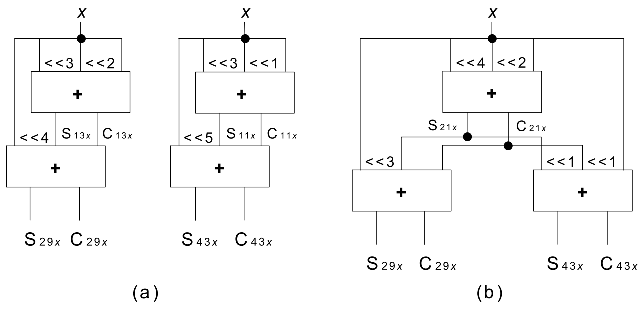

Fig. 5. Implementation of $29x$ and $43x$ using CSA blocks obtained with: (a) a digit-based recoding technique [19]; (b) the exact CSE algorithm [16].

number (GN) representation than the exact CSE algorithm of [16] was also proposed in [16]. Due to the larger search space, the approximate algorithm generally obtains better solutions than the exact CSE algorithm. The GB algorithm of [14] includes optimal and heuristic parts. In the optimal part, the constant multiplications that can be implemented using one CSA block are synthesized. If there exist unrealized constant multiplications, then in each iteration of the heuristic part, an unrealized constant multiplication is synthesized with two CSA blocks or with its minimum number of CSA block implementation obtained from [21] by including partial products.

III. The CAD Tool

In this section, we introduce the CAD tool designed for the implementation of low-complexity and high-speed digital FIR filters. The FIR filter design process includes three main steps: i) Obtaining the solutions of high-level algorithms designed for the optimization of the number of RCA or CSA blocks in the multiplier block of the FIR filter based on the design objective; ii) Defining the FIR filter in a synthesizable description format[2] using the optimized multiplier block obtained by the high-level algorithms; iii) Synthesizing the FIR filter at gate-level by mapping the synthesizable description in a sequential circuit.

In the first step of the design process, for the design of low-complexity and high-speed FIR filters, the multiplier block of the FIR filter is optimized in terms of the number of RCA and CSA blocks by a high-level algorithm described in Section II-A and II-B, respectively.

In the second and third steps of the design process, an interpreter is used to define the digital FIR filter in a synthesizable description format and synthesize the filter at gate-level. The interpreter takes the solution of a high-level algorithm (*i.e.*, a set of RCA or CSA blocks that generates the multiplications of filter coefficients with the filter input), the set of filter coefficients, and the bit-width of the filter input as inputs. Then, it generates the description of the digital FIR filter with the multiplier block optimized for the number of RCA or CSA blocks and with the registers and addition operations required to compute the filter output.

[2]Currently, the CAD tool only supports the Berkeley Logic Interchange Format (BLIF) and will be extended to describe the FIR filter in VHDL (VHSIC (Very High Speed Integrated Circuit) Hardware Description Language.

A. Design of the FIR Filter using RCAs

In this case, the multiplications of filter coefficients with the filter input are implemented with partial products that are generated using RCAs. Initially, each addition/subtraction operation in the solution of the high-level algorithm is defined as $out_i \gg r_i = in_{i1} \ll l_{i1} \pm in_{i2} \ll l_{i2}$, where r and l denote the amount of right and left shifts respectively. Each addition/subtraction operation implementing a constant multiplication is described in the RCA architecture as given in [22]. Then, each filter coefficient is defined as $h_j = \pm out_k \ll l_k$.

As can be also observed from Figure 1, for the first coefficient multiplication, we include a register block whose input is the first filter coefficient multiplication. Then, for each filter coefficient multiplication except the last one, we describe an adder block to sum the output of the previous register with the filter coefficient multiplication and a register block to store the current sum. For the last filter coefficient multiplication, we only need to include an adder block. Hence, the output of the last adder block determines the filter output.

B. Design of the FIR Filter using CSAs

In this case, the multiplier block of the FIR filter is implemented with partial products that are generated using CSAs. Initially, the interpreter obtains the implementations of constant multiplications from the solution of the high-level algorithm. Then, it finds the inputs of the CSA blocks with the minus sign and generates the 2's complement of the partial product by inverting both the sum and carry parts of the product and adding them together with 1 using a single CSA. Then, the implementation of CSA blocks generated for the partial products and the filter coefficient multiplications in the multiplier block of the FIR filter is realized. If a CSA block that implements a constant multiplication has a minus signal in one of its inputs (both sum and carry always have the same signal, but the other input does not need to have the same signal), the associated negated input is used. Finally, the outputs of the multiplier block are defined from the filter coefficients.

After the multiplier block of the filter is described, the filter output is obtained as illustrated in Figure 3. In this case, two CSA blocks are used to add the filter coefficient multiplication with the previous sum and carry results and two registers are used to store the sum and carry parts of the current result. The filter output is obtained with the VMA block where the outputs of the final registers are added together using a Kogge-Stone adder [12]. The Kogge-Stone adder is used in order to speed-up the carry calculation, since this block represents a bottleneck in the performance of the filter. The Kogge-Stone adder is a parallel-prefix adder and this structure appears as a good alternative, because it introduces a prefix network with a minimal fan-out of an unit at each node. However, the reduction of logic depth is achieved at cost of more area.

C. Mapping the FIR Filter at Gate-Level

In the third step of the design process, after the FIR filter is defined in BLIF, the SIS tool is used to map the filter using a gate-library and to obtain the area and delay results of the filter at gate-level. In the technology mapping process, while the FIR filter using RCA blocks is synthesized under the minimum area design strategy, the FIR filter using CSA blocks is mapped under the minimum delay design strategy. Finally, the power dissipation in the FIR filter is determined using the Switch Level Simulator (SLS) tool [23].

TABLE I
FIR FILTER SPECIFICATIONS.

Filter	pass	stop	#tap	width
1	0.10	0.25	100	10
2	0.15	0.25	40	12
3	0.20	0.25	80	12
4	0.15	0.25	60	14
5	0.15	0.20	60	14
6	0.10	0.15	60	14

IV. EXPERIMENTAL RESULTS

In this section, we present the results of high-level algorithms designed for the optimization of the number of RCA and CSA blocks in the multiplier block of the FIR filter and give the gate-level area, delay, and power dissipation results of filters obtained by the CAD tool. Also, we introduce the gate-level results of FIR filters when their multiplier blocks are realized using radix-8 array [3] and Mullet [18] multipliers.

As an experiment set, we used FIR filter instances[3] where filter coefficients were computed with the *remez* algorithm in MATLAB. The specifications of filters are presented in Table I where *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.

The results of high-level algorithms are given in Table II, where *#RCA*, *#CSA*, and *step* denote the number of RCA blocks, the number of CSA blocks, and the maximum number of operations in series, generally known as the number of adder-steps, respectively. Also, *CPU* presents the CPU time required for the high-level algorithms to obtain their solutions on a PC with Intel Xeon at 2.33GHz and 4GB of memory. Note that the results of the exact CSE algorithms [7], [16] were obtained when the filter coefficients were defined under MSD representation. In the CSE heuristic [15] and the approximate algorithm [16], the filter coefficients were defined under CSD and general number (GN) representation respectively.

As can be observed from Table II, the exact GB algorithm [10] obtains significantly better solutions than the exact CSE algorithm [7] in terms of the number of RCA blocks. This is simply because the GB algorithm is not restricted to any particular number representation and considers a larger number of alternative implementations than the CSE algorithms, increasing the possible sharing of partial products. On the other hand, the approximate algorithm of [16] that considers the filter coefficients under general number representation finds similar or better solutions in terms of the number of CSA blocks, obtaining better solutions on overall FIR filter instances, than the heuristic [15] and exact [16] CSE algorithms.

[3]The FIR filter instances are available at http://algos.inesc-id.pt/multicon.

TABLE II
RESULTS OF HIGH-LEVEL ALGORITHMS DESIGNED FOR THE OPTIMIZATION OF THE NUMBER OF RCA AND CSA BLOCKS IN THE MULTIPLIER BLOCK OF THE FIR FILTER.

Objective	Optimization of #RCA blocks						Optimization of #CSA blocks								
Algorithm	Exact CSE [7]			Exact GB [10]			CSE Heuristic [15]			Exact CSE [16]			Approximate GN [16]		
Filter	#RCA	step	CPU	#RCA	step	CPU	#CSA	step	CPU	#CSA	step	CPU	#CSA	step	CPU
1	18	3	0.01	17	3	0.09	15	3	0.04	14	4	0.01	14	3	0.12
2	16	3	0.01	15	4	0.28	16	3	0.02	16	4	0.03	15	4	1.53
3	29	4	0.01	28	3	0.14	30	3	0.05	27	4	0.08	25	4	0.17
4	22	3	0.01	20	4	0.20	25	3	0.04	21	5	0.19	21	5	2.23
5	34	3	0.01	29	4	0.30	36	3	0.06	34	4	0.22	34	3	9.58
6	33	4	0.03	28	6	0.54	36	3	0.07	32	4	0.28	32	4	4.19
Total	152	20	0.08	137	24	1.55	158	18	0.28	144	25	0.81	141	23	17.82

TABLE III
GATE-LEVEL DESIGN RESULTS OF FIR FILTERS OBTAINED WITH THE SOLUTIONS OF ALGORITHMS DESIGNED FOR THE OPTIMIZATION OF THE NUMBER OF RCA AND CSA BLOCKS GIVEN IN TABLE II.

Objective	Optimization of #RCA blocks						Optimization of #CSA blocks								
Algorithm	Exact CSE [7]			Exact GB [10]			CSE Heuristic [15]			Exact CSE [16]			Approximate GN [16]		
Filter	area	delay	power	area	delay	power	area	delay	power	area	delay	power	area	delay	power
1	134.1	208.4	808.3	131.8	205.1	772.2	274.7	35.8	873.0	273.4	37.2	870.6	274.9	35.9	872.0
2	75.1	106.4	521.8	74.7	105.7	814.0	161.6	38.6	564.0	160.3	38.4	546.2	161.4	36.7	596.6
3	157.2	229.1	1187.5	159.3	231.2	1188.3	367.5	40.0	1301.2	366.5	40.5	1307.7	364.8	40.2	1366.4
4	108.2	154.1	865.4	104.6	147.0	1057.2	237.1	40.0	883.7	234.5	43.4	919.6	234.8	44.4	941.4
5	132.3	170.3	1172.8	128.2	170.3	1318.9	298.2	38.1	1139.9	298.2	38.9	1171.1	298.5	39.2	1154.4
6	132.5	170.0	1335.6	130.8	171.3	2197.7	304.0	40.3	1266.3	302.7	40.9	1286.5	301.4	43.6	1315.2
Total	739.4	1038.3	5891.4	729.4	1030.6	7348.3	1643.1	232.8	6028.1	1635.6	239.3	6101.7	1635.8	240.0	6246.0

TABLE IV
GATE-LEVEL DESIGN RESULTS OF FIR FILTERS USING GENERAL MULTIPLIERS.

Multiplier	Radix-8 [3]						Mullet [18]						#mul
Strategy	Minimum Area			Minimum Delay			Minimum Area			Minimum Delay			
Filter	area	delay	power	area	delay	power	area	delay	power	area	delay	power	
1	193.8	210.3	1325.3	264.9	64.8	1192.0	187.9	210.3	1119.0	254.0	54.8	1049.1	17
2	127.3	108.2	1129.0	170.8	63.3	1006.0	121.9	108.2	949.5	161.1	57.0	893.6	14
3	260.0	229.1	2139.0	350.0	62.3	1880.2	249.4	229.1	1833.9	330.3	56.3	1573.3	28
4	186.7	154.0	1778.7	249.1	65.7	1591.4	175.1	154.0	1430.8	230.0	56.5	1238.3	20
5	247.4	172.5	2432.3	325.2	66.7	1094.7	230.3	172.5	1988.4	297.8	57.2	1671.9	29
6	252.9	169.6	2683.3	332.5	67.0	2452.3	233.5	169.6	2181.1	301.2	57.3	1868.0	28
Total	1268.1	1043.7	11487.6	1692.5	389.8	9216.6	1198.1	1043.7	9502.7	1574.4	339.1	8294.2	136

Also, observe from Table II that the solutions of the high-level algorithms are obtained using a little computational effort.

Table III presents the gate-level area, delay, and power dissipation results of the FIR filters whose multiplier blocks are optimized for the number of RCA and CSA blocks by the algorithms given in Table II. Note that in the SIS tool, each filter designed using RCA and CSA blocks is synthesized under the minimum area and the minimum delay design strategy respectively. In this table, *area (mm^2)*, *delay (ns)*, and *power (mW)* denote the area, delay, and power dissipation of the filter at gate-level respectively. The *sis.genlib* gate library of the SIS tool was used during the technology mapping and the gate-level results were obtained when the bit-width of the filter input was 16. Also, the power dissipation results were determined when 10,000 random input vectors were applied.

As can be observed from Tables II and III, the reduction of the number of RCA blocks in the multiplier block of the FIR filter results in the reduction of area of the FIR filter at gate-level. The same situation generally occurs when the number of CSA blocks is reduced in the multiplier block of the FIR filter by the high-level algorithms, as can be clearly observed when the results of CSE heuristic [15] are compared with the results of the exact CSE and approximate algorithms of [16] on overall instances. On the other hand, the design of FIR filters using RCA blocks yields low-complexity implementations compared to the FIR filter designs using CSA blocks. In this case, the area improvement on overall instances between the results of the exact GB algorithm [10] and the CSE heuristic [15] is 55.6%. However, the design of FIR filters using CSA blocks leads to high-speed implementations with respect to FIR filter designs using RCA blocks. In this case, the delay improvement on overall instances between the CSE heuristic [15] and the exact CSE algorithm [7] is 77.6%. Although one can easily tune on the traditional tradeoff between area and delay of the design during the technology mapping, we observed that the delay results of FIR filter designs using CSA blocks were always better than those of designs using RCA blocks mapped under the minimum delay design strategy and the area results of FIR filter designs using RCA blocks were always better than those of designs using CSA blocks mapped under the minimum area design strategy.

Table IV presents the gate-level area, delay, and power dissipation results of the FIR filters whose multiplier blocks are realized using radix-8 array [3] and Mullet [18] multipliers (the FIR filters were also designed using the modified Booth multipliers [2], but they lead to the worst results in terms of area and delay with respect to these multipliers). In this case, the FIR filters are designed as illustrated in Figure 1, where the multiplier block is implemented using general multipliers. We note that as done in the design of FIR filters using RCA and

CSA blocks, in the design of the multiplier block of the FIR filter using general multipliers, we initially convert the filter coefficients to positive and odd numbers and then, implement the filter coefficient multiplications without a repetition. Thus, the number of required multipliers in the multiplier block of the FIR filter is equal to the number of unrepeated positive and odd filter coefficients denoted by *#mul* in the last column of Table IV. In order to find out the pareto-optimal points of the design and compare the FIR filter designs with those given in Table III, we present the gate-level design results of FIR filters when they are mapped under both minimum area and delay design strategies. Note that the redundancy in the general multipliers implementing the multiplications of constant filter coefficients with the filter input is removed during the technology mapping. The experimental settings used in the design of these FIR filters are the same as those used in the design of the FIR filters given in Table III.

As can be easily observed from Tables III and IV, under the minimum area design strategy, the design of FIR filters using RCA blocks yields FIR filter designs with significantly better area and also delay results than those obtained using all kind of multipliers given in Table IV on overall instances. On the other hand, under the minimum delay design strategy, the design of FIR filters using CSA blocks yields better delay designs with respect to FIR filter designs obtained using each type of multiplier given in Table IV. Also, this design methodology yields better area designs than FIR filters whose multiplier block is designed using radix-8 array multipliers on overall instances. Furthermore, observe from Tables III and IV that the use of shift-adds architecture in the design of FIR filters and the use of high-level algorithms for the optimization of the number of RCA and CSA blocks lead to low-power digital FIR filter designs.

V. Conclusions

In this paper, we introduced a CAD tool that can synthesize low-complexity and high-speed digital FIR filters based on a design objective. The most crucial property of the tool is that in the implementation of the multiplier block of the filter, it replaces each multiplication by a constant with addition/subtraction and shifting operations. While the low-complexity design of the filter is realized using RCA blocks, the high-speed design of the filter is achieved using CSA blocks. Furthermore, to reduce the complexity of the design, the developed CAD tool incorporates high-level algorithms that find a solution with the minimum number of RCA or CSA blocks required to generate the filter coefficient multiplications. It is observed that while the use of RCA blocks leads to a low-complexity filter design, the use of CSA blocks yields a high-speed filter. Also, the use of high-level algorithms in the design of the multiplier block has significant impact on the complexity of the filter. It is shown that the proposed design methodology yields low-complexity and high-speed FIR filters with respect to those whose multiplier blocks are designed using general multipliers.

VI. Acknowledgment

This work was partially supported by the *Portuguese Foundation for Science and Technology* (FCT) under the research project *Architectural Optimization of DSP Systems with Multiple Constants Multiplications* (Multicon) PTDC/EIA-EIA/103532/2008 and by FCT (INESC-ID multiannual funding) through the PIDDAC Program funds.

References

[1] C. Wallace, “A Suggestion for a Fast Multiplier,” *IEEE Transactions on Electronic Computers*, vol. 13, no. 1, pp. 14–17, 1964.
[2] W. Gallagher and E. Swartzlander, “High Radix Booth Multipliers using Reduced Area Adder Trees,” in *Proc. of Asilomar Conference on Signals, Systems and Computers*, 1994, pp. 545–549.
[3] E. Costa, S. Bampi, and J. Monteiro, “A New Architecture for Signed Radix 2m Pure Array Multipliers,” in *Proc. of IEEE International Conference on Computer Design*, 2002, pp. 112–117.
[4] 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*, vol. 8, no. 4, pp. 419–424, 2000.
[5] R. Hartley, “Subexpression Sharing in Filters using Canonic Signed Digit Multipliers,” *IEEE TCAS II*, vol. 43, no. 10, pp. 677–688, 1996.
[6] I.-C. Park and H.-J. Kang, “Digital Filter Synthesis Based on Minimal Signed Digit Representation,” in *Proc. of DAC*, 2001, pp. 468–473.
[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 TCAD*, vol. 27, no. 6, pp. 1013–1026, 2008.
[8] 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.
[9] Y. Voronenko and M. Püschel, “Multiplierless Multiple Constant Multiplication,” *ACM Transactions on Algorithms*, vol. 3, no. 2, 2007.
[10] L. Aksoy, E. Gunes, and P. Flores, “Search Algorithms for the Multiple Constant Multiplications Problem: Exact and Approximate,” *Elsevier Journal on Microprocessors and Microsystems*, vol. 34, pp. 151–162, 2010.
[11] P. Cappello and K. Steiglitz, “Some Complexity Issues in Digital Signal Processing,” *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 32, no. 5, pp. 1037–1041, 1984.
[12] P. M. Kogge and H. S. Stone, “A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations,” *IEEE Transactions on Computers*, no. 8, pp. 786–793, 1973.
[13] R. Hawley, B. Wong, T.-J. Lin, J. Laskowski, and H. Samueli, “Design Techniques for Silicon Compiler Implementations of High-Speed FIR Digital Filters,” *IEEE Journal of Solid-State Circuits*, vol. 31, no. 5, pp. 656–667, 1996.
[14] O. Gustafsson, A. Dempster, and L. Wanhammar, “Multiplier Blocks using Carry-Save Adders,” in *Proc. of ISCAS*, 2004, pp. 473–476.
[15] A. Hosangadi, F. Fallah, and R. Kastner, “Optimizing High Speed Arithmetic Circuits using Three-Term Extraction,” in *Proc. of DATE*, 2006, pp. 1294–1299.
[16] L. Aksoy and E. Gunes, “Area Optimization Algorithms in High-Speed Digital FIR Filter Synthesis,” in *Proc. of SBCCI*, 2008, pp. 64–69.
[17] Sequential interactive system webpage. [Online]. Available: http://embedded.eecs.berkeley.edu/pubs/downloads/sis/index.htm
[18] K. Tsoi and P. Leong, “Mullet - A Parallel Multiplier Generator,” in *Proc. of FPL*, 2005, pp. 691–694.
[19] M. Ercegovac and T. Lang, *Digital Arithmetic*. Morgan Kaufmann, 2003.
[20] A. Avizienis, “Signed-digit Number Representation for Fast Parallel Arithmetic,” *IRE Transactions on Electronic Computers*, vol. EC-10, pp. 389–400, 1961.
[21] O. Gustafsson, H. Ohlsson, and L. Wanhammar, “Minimum-Adder Integer Multipliers using Carry-Save Adders,” in *Proc. of ISCAS*, 2001, pp. 709–712.
[22] K. Johansson, O. Gustafsson, and L. Wanhammar, “Bit-Level Optimization of Shift-and-Add Based FIR Filters,” in *Proc.of ICECS*, 2007, pp. 713–716.
[23] Switch-level simulator webpage. [Online]. Available: http://www.space.tudelft.nl/