VHDL Generation of Optimized FIR Filters

Fábio Fabian Daitx
Informatics Institute, UFRGS, Porto Alegre – RS, Brazil
fabiodaitx@gmail.com

Vagner S. Rosa
Informatics Institute UFRGS, Porto Alegre, RS, Brazil
vsrosa@inf.ufrgs.br

Eduardo Costa
UCPel, Pelotas – RS, Brazil
ecosta@ucpel.tche.br

Paulo Flores
INESC-ID/IST, TULisbon, Portugal
pff@inesc-id.pt

Sérgio Bampi
Informatics Institute, UFRGS, Porto Alegre, Brazil
bampi@inf.ufrgs.br

Abstract – **This work proposes an VHDL generation software for optimized FIR filters. In this paper a near optimum algorithm for constant coefficient FIR filters was used. This algorithm uses general coefficient representation for the optimal sharing of partial products in Multiple Constants Multiplications (MCM). The developed tool was compared to Matlab FDA toolbox. Synthesis results show that our tool is able to produce significantly better hardware than FDA toolbox, doubling the speed and reducing the silicon area by 75%. The software produces a generic VHDL output, synthesizable to ASIC or FPGA.**

I. INTRODUCTION

Finite impulse response (FIR) digital filters are widely used in digital signal processing by virtue of stability and easy implementation. The main drawback of FIR filters is the amount of computation needed to process a signal through a FIR filter. The most expensive operators in terms of area, delay and power in a FIR filter structure are the multipliers. However, some applications like front-ends for software defined radio require high speed parallel filters with constant coefficients. These applications can benefit from substantial hardware reduction if a dedicated multiplier block is employed. This multiplier block can be significantly reduced if implemented as shift-adders [2] and common sub-expressions are shared among them. Since shifts are free in terms of hardware, the design problem can be defined as the minimization of the number of addition/subtraction operations to implement the coefficient multiplications. This is usually called the Multiple Constants Multiplication (MCM) problem.

In fact, the problem of designing FIR filters has received significant attention during the last decade and many algorithms have been proposed to minimize the number of addition/subtraction operations in the multiplier block of the filter [1,3,4,5,6,7,12,13], but none of them present synthesis results. In our work, we present a fully functional software to generate optimized FIR filters based on the algorithm developed by [1]. This algorithm was chosen because to the best of our knowledge it is one of the best heuristic algorithms for the MCM problem applied to FIR filters. Moreover, we had access to the source code of the original implementation that facilitates its adoption in the proposed software tool. However, other algorithms, such as BHM, RAG-n [14], Hcub [15] or ASSUME [16], could be implemented and easily integrated in the tool. The algorithm goal is the maximization of partial terms sharing in MCM operations under general number representation, in which coefficients are represented using decimal integers.

The near optimum algorithm for constant coefficient FIR filters developed on [1], accounts only the number of adders needed to implement the multiplier block. In this work, we propose a synthesis software tool, based on this algorithm, to generate an optimized VHDL hardware description of FIR filters. The designed software supports coefficient and data bit-width configuration at design time. The generated VHDL code is generic and can be synthesized using any common synthesis tool.

The proposed tool was validated by synthesizing a set of FIR filters benchmarks and comparing its implementation results with the implementations generated by Matlab Filter Design and Analysis (FDA) toolbox. This tool was chosen as reference for comparison due to its versatile set of optimization resources, and the ability to generate VHDL for either FPGAs or ASICs, like the proposed tool. As will be shown later, our tool produces significantly better VHDL descriptions for the FIR filters. Moreover, the fact of using the algorithm of [1] enables significant gains in terms of number of adders/subtracters (in the multiplier block) and operational frequency.

The rest of this paper is organized as follows. In Section 2 we give an overview of relevant work on FIR filter design. Section 3 briefly explains FIR filter design. In Section 4, we describe the algorithm used to optimize the FIR filters. In Section 5 it is presented the developed tool for the generation of optimized VHDL code. Experimental results comparing the filters implementations generated by our tool and Matlab FDA toolbox are presented in Section 6. Finally, in Section 7 we conclude this paper, discussing the main contributions and future work.

II. RELATED WORK

A large amount of work has addressed the use of efficient implementations of multiplier-less MCMs. The techniques include the use of different number representation schemes, the use of different architectures and implementation styles and the coefficient optimization techniques, e.g., [1-8]. Most of these work use heuristic algorithms to minimize the total number of adders/subtracters in the multiplier blocks of the filters.

Authorized licensed use limited to: POLO BIBLIOTECARIO DI INGEGNERIA. Downloaded on January 10,2022 at 12:14:31 UTC from IEEE Xplore. Restrictions apply.

An exact algorithm that finds the best representation, but only for a single coefficient is presented in [7]. A more efficient exact algorithm that maximizes the sharing of partial terms in MCM operations, based on solving a 0-1 Integer Linear Programming (ILP) over a Boolean network, is proposed in [5].

Although the mentioned heuristic and the exact algorithms perform well on computing solutions for sharing of partial terms in MCM approach, none of them proposes a fully functional tool to generate and implement the optimized FIR filter architectures. A parallel FIR filters generation tool is proposed in [8]. This generation tool employs Canonical Signed Digit (CSD) representation, because this representation results in fewer non-zero digits in each coefficient. Recently, Park and Kang [6] proposed the usage of a more efficient representation for the coefficients: Minimal Signed Digit (MSD). Under the MSD representation, a given numerical value can have multiple representations. In our work, we have used the algorithm proposed by [1], which uses a general number representation for the coefficients. In this representation, common signed decimal integer values are used for the coefficients and their redundancy in terms of the corresponding internal binary number representation is exploited by selecting the instance that minimizes the total hardware.

In commercial tools from FPGA design, such as Altera and Xilinx [9, 10], core generators for optimized FIR filters can be obtained. However, these tools use specific hardware resources embedded in some FPGAs, therefore limiting its usage to specific FPGA technologies. Our proposed software uses only generic VHDL constructors such as adders, subtractors and registers. All optimizations are done by software using efficient state of the art algorithms in order to produce a synthesizable VHDL description that implements an efficient architecture.

A toolbox for Matlab was also analyzed. This tool can produce HDL descriptons with great versatility in the design phase. However, our results show that this tool is not able to produce efficient HDL descriptions for optimized FIR architectures as proposed in our work. Section 6 shows comparative results between VHDL descriptions obtained by Matlab and our proposed software.

III. FIR FILTER DESIGN

In digital circuits, a FIR (Finite Impulse Response) filter can be viewed as a functional block, as shown in Figure 1, that implements the equation (1) [11]:

$$Y[n] = \sum_{i=0}^{N-1} H[i]X[n-i] \ , \qquad (1)$$

where N is the number of coefficients (or taps) of the filter, X is the input signal, Y the output signal, $Y[n]$ the current output sample and H represents the filter coefficients.

The coefficients of the FIR filter are obtained using the Discrete Fourier Transform (DFT) of the required frequency transfer function with some known windowing method.

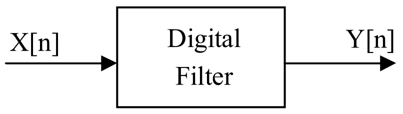

Figure 1: Digital filter functional block.

In this work we will use a fully-parallel implementation of FIR filters. In particular, we will explore the transposed form architecture, as shown in Figure 2. This architecture involves multiplying all the coefficients by the same input data. Because registers are now between the adders, the resulting circuit is faster, since critical path is smaller. Moreover, due to register localization most of the glitching activity is filtered, resulting in a significant power reduction. Thus, for some multiplier block optimization algorithms, the transposed form is the preferred choice [11,12,13].

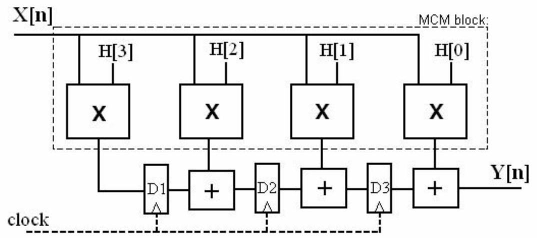

Figure 2: Transposed form of a 4 taps FIR filter implementation. The MCM block is shown inside the striped line rectangle.

Common optimization strategies explore the fact that the FIR filter multiplications can be implemented by sharing partial results/terms through the adders/subtracters circuit tree that implements the MCM block. The MCM block, shown in Figure 2, is the combinational hardware resulting from the MCM problem solution, with all the possible optimizations specified in an algorithm [1-6, 14-16] that implements all the filter multipliers simultaneously. We have used shift-add multiplier circuits in order to optimize the filter architectures. In this approach, when using some coefficient representation (binary, CDS or MSD), for each bit 1 over the first non-zero digit in the coefficient, an adder/subtractor must be used in order to provide the partial product terms. Therefore, when a higher the number of non-zero digits is used in the coefficients, a higher number of adders and subtractors operators are needed for their implementation.

IV. ALGORITHM DESCRIPTION

In this section, we briefly describe the software tool that we propose to automatically generate synthesizable and optimized VHDL descriptions for digital FIR filters.

A Algorithm for General Coefficient Representation

In the algorithm of [1], there are no considerations about number representations, because only the numerical values of the coefficients are taken into account. In this algorithm, *Cset* maintains numerical values of all coefficients not yet covered and *Patset* is the set with the partial terms found so far. Before being

Authorized licensed use limited to: POLO BIBLIOTECARIO DI INGEGNERIA. Downloaded on January 10,2022 at 12:14:31 UTC from IEEE Xplore. Restrictions apply.

inserted into *Cset*, all values are made odd by successive shifted rights and any duplicates that may appear in this process are eliminated. *Patset* is initialized with a single element, the value 1. Finally, the algorithm enters a loop where all shifted versions of elements in *Patset* are pairwised added and subtracted:

1. remove all coefficients in *Cset* that have the same value as any shifted value of an element in *Patset*.
2. remove all coefficients in *Cset* whose value can be obtained by adding or subtracting shifted versions of two elements in *Patset*. Insert the elements removed into *Patset*.
3. remove all coefficients in *Cset* whose value can be obtained by adding or subtracting shifted versions of three elements in *Patset*. Insert the elements removed into *Patset*. If no element was removed from *Cset* in the previous steps, go to Step 4. Otherwise, go to Step 1.
4. check which of the partial terms obtained by adding or subtracting shifted versions of two elements in *Patset* maximally matches a subset of bits of a CSD representation. Register the combination as a new partial term in *Patset* and insert into *Cset* a new element obtained by removing the new partial term from the selected decimal representation. Go to Step 1.
5. This loop is repeated until there are no more coefficients in *Cset*.

An important point in the algorithm shown above is the fact that any pairwise combination is valid. This aspect enables significant hardware reduction by this algorithm, when compared to the algorithm of [6], which does not consider pairwise combinations of shifted elements of *Patset* if non-zero bits have the same position.

V. IMPLEMENTATION

The software implementation was developed in C language and produces VHDL code for the optimized FIR filter from a coefficient specification file. The generation is done in two steps. The first step read the coefficients and apply the algorithm depicted in Section IV, producing an intermediate file containing the result from the MCM algorithm. This intermediate file is produced in a human-readable format (figure 4). Then the VHDL implementation of a transposed form FIR filter is generated using the intermediate file. The intermediate file allows other MCM algorithms to be implemented without the effort to produce the VHDL output. For a given set of coefficients the software generates transposed form FIR architecture with the dedicated MCM block, as shown in the 7 tap example filter in Figure 3. Figure 4 presents the intermediate file generated by the MCM algorithm. Figure 5 presents some samples of the VHDL generated. The coefficients used in Figures 3, 4, and 5 were (`2,-3,9,12,9,-3,2`).

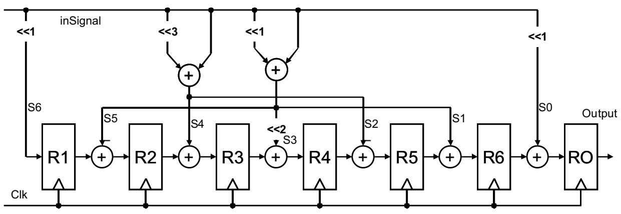

Figure 3: Generated FIR block diagram.

```
# Coeficientes de entrada do filtro FIR
filter:
2
-3
9
12
9
-3
2

# Coeficientes sintetizados do filtro FIR
implement:
3 = 1 << 1 + 1 << 0
9 = 1 << 3 + 1 << 0
```

Figure 5: Intermediate file.

```
...
ENTITY FIR_filter IS PORT (
clk           : IN  STD_LOGIC;
inSignal      : IN  STD_LOGIC_VECTOR(15 DOWNTO 0);
Output : OUT STD_LOGIC_VECTOR(32 DOWNTO 0) );
END FIR_filter;
...
s0<=("000000000000000"&inSignal&"0");
s1<=("000000000000000"&inSignal&"0")+inSignal;
s2<=("0000000000000"&inSignal&"000")+inSignal;
s3<=s5(29 DOWNTO 0)&"00";
s4<=s2;
s5<=s1;
s6<=s0;
Output<=reg_out;
...
Process(...)
  reg1 <= "0" & s6 ;
  reg2 <= reg1 - s5 ;
  reg3 <= reg2 + s4 ;
  reg4 <= reg3 + s3 ;
  reg5 <= reg4 + s2 ;
  reg6 <= reg5 - s1 ;
  reg_out <= reg6 + s0 ;
...
```

Figure 5: Sample of the VHDL code generated by the proposed software.

In the example of figure 3 and 4, only two adders were used in the multiplier block. The shift by a constant operation is a area-free operation, as only the wires need to be properly connected to perform the operation. The VHDL generation software accepts parameters to configure the input and output bitwidth and also the presence of RESET and clock Enable inputs.

VI. EXPERIMENTAL RESULTS

The developed software was applied to several FIR filters and compared to Matlab's Filter Design & Analysis (FDA). The FDA toolbox includes a feature to generate optimized VHDL code from the generated coefficients and was used to compare to the developed software. Other tools from Xilinx [10] and Altera [9] were also investigated. These tools produce much more optimized FIR filters implementations but targeting their vendor specific devices only, which results on higher performance FIR filters implementations. However, their solutions cannot be compared with the results obtained by our tool since the code generated is vendor/device

Authorized licensed use limited to: POLO BIBLIOTECARIO DI INGEGNERIA. Downloaded on January 10,2022 at 12:14:31 UTC from IEEE Xplore. Restrictions apply.

specific and not generic VHDL as ours. Therefore, no comparison with those tools is done on this paper.

The specification of the filters used in this work is summarized in the leftmost column of Table 1. In this table *Filter* is just an index for each example*, Design Method* corresponds to the mathematical techniques used to generate the coefficients, *Type* is the characteristic of the filters, *Fs*, *Fpass* and *Fstop* are the sampling frequency, passband frequency and stopband frequency (in Hz), respectively and, finally, *Order* is the number of taps minus one. The right side of Table 1 presents the filters parameters, where *input* is the size of the input signal and *Width* is the coefficients word size, both in terms of number of bits, and *#coef* indicates the total number of different coefficients. The generated codes were synthesized using the Xilinx ISE 9.1.01i tool for the Virtex 2 Pro FPGA device. All the optimizations in the proposed software were done in the multiplier block with the algorithm described in the Section IV.

A Results from the Proposed software

After designing the filters and generating the coefficients, the target VHDL code was synthesized using the Matlab FDA toolbox feature (version 7.0.1.24704(R14) – service pack 1). To achieve a fair comparison the Matlab generation tool was set to use the following parameters: *Factored-CSD* multipliers, *Optimize for HDL* option and the *Tree* FIR adder Style. These parameters enable the Matlab FDA toolbox to optimize the generated VHDL description by taking into account the use of adders and shifts to make the multiplications. This aspect enables code optimization, increase of circuit performance and the use of adder trees in the filter architecture in a similar way as the proposed software. Table 2 shows FIR filter results obtained from Matlab FDA toolbox and our proposed software after the synthesis of the circuits with Xilinx ISE.

As can be observed in Table 2, the architectures generated by the proposed software have a lower number of adders/subtracters for all the filters. This is due to the optimized architecture generated by our software that removes redundant adders and subtracters more efficiently than the Matlab FDA toolbox. It can be also observed in Table 2 that the filters generated by the proposed software present higher performance, when compared to the filters generated by Matlab FDA toolbox. In fact, we have observed that the filter architectures generated by our software present a short critical path (smaller adders/subtracters logic depth) leading to higher frequency. Moreover, the VHDL code generated by FDA toolbox does not use good sub-expression sharing algorithms for coefficient optimizations. We have also observed that the VHDL code generated using Matlab FDA tool is much longer and complicated to understand than the proposed software.

As we can observe in Table 2, filter 7 generated by our proposed software, presents the best operation frequency among the filters. We have observed that this occurs because the coefficient values for this filter are easier to be built, since they present a lower number of non-zero digits. On the other hand, the opposite is observed for the filter 2, which present the lower operation frequency value.

The last two columns of Table 2 show the significant improvements, in percentage, presented by our software when compared to the results presented by Matlab tool. As can be observed in Table 2, significant gains in area (number of adders/subtracters) and frequency operation are reported.

VII. CONCLUSION

We have implemented a software tool that generates optimized VHDL descriptions for FIR filters. The developed software uses the algorithm of [1], which uses general coefficient representation in the optimization of the MCM problem.

We presented results for several benchmark FIR filters. For these filters, our software is able to produce significantly better descriptions than commercial Matlab FDA tool. Moreover, the final synthesized filter circuits, targeting an FPGA have significant improvements. An important reduction in the number of operators (with directly impact on circuit area) was achieved while, simultaneous, the maximum frequency of operation was increased. We got expressive results by reducing the number of adders/subtracters up to 76% and increasing the operation frequency of some filters over 100%. As future developments of this work, we plan to integrate in our software a graphical interface to integrate the complete synthesis flow, from the filter specification to a synthesizable VHDL description.

Acknowledgements
This work was supported (in part) by the National Council of Scientific and Technologic Development (CNPq) trough the Federal University of Rio Grande do Sul (UFRGS) at Porto Alegre, Brazil.

REFERENCES

[1] E. Costa, P. Flores, J. Monteiro. “Exploiting General Coefficient Representation for the Optimal Sharing of Partial Products in MCMs”. In Symposium on Integrated Circuits and System Design, Ouro Preto (Minas Gerais), Brazil, 2006.

[2] M. Mehendale, S. D. Sherlekar and G. Venkatesh, “Syntesis of multiplier-less FIR filters with minimum number of additions”, in Proc. IEEE/ACM Int. Conf. Computer-Aided Design, pp. 668-671, 1995,

[3] M. Potkonjak, M. B. Srivastava, and A. Chandrakasan. “Multiple Constant Multiplications: Efficient and Versatile Framework and Algorithms for Exploring Common Subexpression Elimination”. In IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 15, no. 2, pp. 151-165, 1996.

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

Authorized licensed use limited to: POLO BIBLIOTECARIO DI INGEGNERIA. Downloaded on January 10,2022 at 12:14:31 UTC from IEEE Xplore. Restrictions apply.

[5] P. Flores, J. Monteiro, E. Costa. "An Exact Algorithm for the Maximal Sharing of Partial Terms in Multiple Constant Multiplications". In International Conference on Computing Aided Design, San Jose(California):USA, 2005.

[6] I-C. Park, H-J. Kang. "Digital Filter Synthesis Based on Minimal Signed Digit Representation". In Design Automation Conference, pp. 468-473, 2001.

[7] A. Dempster; M. Macleod. "Using All Signed-Digit Representations to Design Single Integer Multipliers Using Subexpression Elimination". In International Symposium on Circuits and Systems, pp. 24-26, 2004.

[8] V. S. Rosa, E. Costa, S. Bampi. "A High Performance Parallel FIR Filters Generation Tool". In Iberchip, San Jose:Costa Rica, 2006.

[9] Altera Corporation, 101 Innovation Drive, San Jose, California 95134, USA. <http://www.altera.com>

[10] Xilinx, Inc. http://www.xilinx.com

[11] Hamming, R. W. "Digital Filters", Prentice Hall, 3rd ed., 1989.

[12] Potkonjak, M.; Srivastava, M. B.; Chandrakasan, A. "Efficient substitution of multiple Constant multiplication by shifts and addition using iterative pairwise matching". In Proc. 31st ACM/IEEE Design Antomation Conf., pp. 189-194, 1994.

[13] Pasko, R.; Schaumont, P.; Derudder, V.; Vernalde, S.; Iuraekova, D. "A new algorithm for elimination of common subexpressions", IEEE Trans. Computer-Aided Design, vol. 18. pp. 58-68, Jan 1999.

[14] . Dempster. M. Macleod. "Use of minimum-Adder Multiplier Blocks in FIR Digital Filters", IEEE Trans. on Circuits and Systems II, vol. 42, no. 9, pp.567-577, 1995

[15] Y. Voronenko, M. Puschel, "Multiplierless Multiple Constant Multiplication", ACM Trans. On Algorithms, vol. 3, no. 2, 2007.

[16] L. Aksoy, E . Costa, P. Flores, J. Monteiro, "Exact and Approximante Algorithm for the Optimization of Area and Delay in Multiple Constant Multiplications", IEEE Tans. Computer-Aided Design, vol. 27, no. 6, 2008.

Table 1: Benchmark FIR filter specifications and parameters.

Filter	Filter Specification						Filter Parameters		
	Design Method	Type	Fs	Fpass	Fstop	Order	input	width	#coef
1	Equiripple	Lowpass	48000	9600	12000	50	16	16	26
2	Equiripple	Lowpass	48000	7200	9600	50	16	20	26
3	Equiripple	Lowpass	48000	4800	6400	76	16	14	39
4	Window (Kaiser)	Lowpass	48000	4800	6400	151	16	16	73
5	Equiripple	Highpass	48000	12000	9600	56	16	18	29
6	Equiripple	Highpass	44100	12000	9600	50	20	18	26
7	Least-squares	Highpass	44100	6615	4410	50	16	12	22
8	Window (Hamming)	Highpass	48000	N/A	10800 (cutoff)	79	16	13	34

Table 2: Experimental Results for FIR Filters.

Filter	Matlab		Proposed Tool		Proposed Tool Improvements	
	# Add./Sub.	Max. Freq. (MHz)	# Add./Sub.	Max. Freq. (MHz)	# Adders/Subtracters	Maximum Frequency
1	251	37.663	77	59.149	-69,32%	+57,05%
2	392	30.687	92	43.195	-76,53%	+40,76%
3	350	34.409	114	74.343	-67,43%	+116,06%
4	668	32.372	212	60.932	-68,26%	+88,22%
5	317	36.613	92	47.621	-70,98%	+30,07%
6	349	30.267	87	44.710	-75,07%	+47,72%
7	138	43.091	60	90.610	-56,52%	+110,28%
8	237	33.622	96	89.617	-59,49%	+166,54%

Authorized licensed use limited to: POLO BIBLIOTECARIO DI INGEGNERIA. Downloaded on January 10,2022 at 12:14:31 UTC from IEEE Xplore. Restrictions apply.