Coefficient Optimizations for High Performance Parallel FIR Filters in FPGAs

Vagner S. Rosa, Sergio Bampi
Inst. Informatics - Univ. Fed. Rio Grande do Sul
Porto Alegre, RS – Brazil
{vsrosa,bampi}@inf.ufrgs.br

Eduardo Costa
Universidade Católica de Pelotas
Pelotas, RS – Brazil
ecosta@ucpel.tche.br

ABSTRACT

This paper presents a new method to optimize hardwired parallel finite impulse response (FIR) filters and its corresponding synthesis results showing hardware and power minimization. The optimization techniques are applied to a FPGAs synthesis process, for multiplier-less fixed coefficients filters. The proposed method combines three approaches: scaled coefficient generation, reduction of the coefficients to N-Power-of-Two (NPT) terms, where the maximum number of non-zero in each coefficient is taken as a constraint, and finally a transfer function directed selection. We present pre and post synthesis results using Quartus II FPGA tool, showing that NPT is a good approach to optimize FIR filters.

1. INTRODUCTION

Finite Impulse Response (FIR) filters are of great importance in the digital signal processing (DSP) world. Their linear phase and feed forward implementation characteristics make them very useful for building high performance filters.

There are two main aspects to be considered when designing a hardwired parallel filter, namely the number of bits required for the signal and the required transfer function of the filter. The former one determines the word length of the entire datapath. The later one is determined by two parameters, namely the number of taps, and the bit-width for each coefficient. The most expensive blocks in terms of area, delay, and power in a FIR filter are the multipliers needed to implement it. In this paper we address optimizations in the filter by a combination of three known approaches. First, we scale the coefficients for further synthesis, and second, convert to power-of-two terms, later selecting the best coefficient set taking into account the transfer function characteristics of the filter. This approach explores the reduction in complexity of the multiplier block by reducing each coefficient to a limited amount of power-of-two terms (nonzero bits). A scale factor to be multiplied by all coefficients prior to its conversion to fixed point is found by exhaustive search, in order to improve the transfer function generated by the N-power-of-two (NPT) coefficients. Second, using the transposed form of the filter and considering all the multipliers implemented as adder trees (hence, multiplier-less), we generate a dedicated adder tree for each multiplier in the multiplier block (multiplier-less implementation).

The goal of this work is to verify the performance of the NPT algorithm when targeting a FPGA implementation in reducing the overall filter complexity and check its applicability to ASIC synthesis as well.

We present a brief review of FIR filter design in section 2 and related work in section 3. In section 4 we present our algorithm, and in section 5 its implementation. Section 6 shows synthesis results and section 7 summarizes the conclusions and presents our proposals for future work.

2. FIR FILTER DESIGN

A FIR filter can be mathematically expressed by the equation (1) [8]:

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

where X represents the input signal, H the filter coefficients, Y the output signal, $Y[n]$ is the current output sample, and N is the number of coefficients (or taps) of the filter. This is a convolution operation of the filter coefficients along the signal. 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. In the sequential implementation a set of multiply-and-accumulate (MAC) operations is performed for each sample of the input data signal, multiplying the N delayed-input samples by coefficients and summing up the results together to generate the output signal. Parallel implementations may follow two architectures. The first one consists of unrolling of MAC loop where we have several delayed versions of the input signal entering in a fully parallel multiplier block, followed by a summation block. The other one consists of a multiplier block, which takes the same input signal and delivers each output to an input of a delayed summation block. The former (Fig. 1a) is the direct form parallel FIR filter and the latter (Fig. 1b) corresponds to its transposed form.

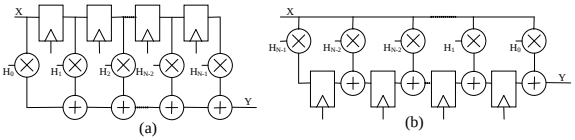


Figure 1. Parallel FIR filters in (a) direct form or (b) transposed direct form.

Both the direct and transposed architectures of the FIR filter have the same complexity [6]. For multiplier block optimization algorithms, the transposed form is preferred [1].

In this work the baseline "non-optimized" implementation for a multiplier-less filter (multipliers are converted to adder trees, using adders only to the nonzero partial products, i.e. nonzero bits) refers to the minimum coefficient width and the number of taps needed to implement it, using the symmetry optimization, i.e. the same multiplier is used for the n-*th* and (N-n)-*th* taps

3. RELATED WORK

There are two main approaches to optimize parallel hardwired constant coefficients FIR filters: coefficient optimizations and multiplier block optimizations. The coefficient optimization approach consists of representing each coefficient as sums of power-of-two terms and limiting the number of power-of-two terms in each coefficient [2,3,5,6]. That means a reduction of the number of bits in '1' state in each coefficient, reducing the number of adders needed to implement the resulting tree. The optimum case is to retain just one power-of-two term in each coefficient, totally eliminating additions in the multiplier block, requiring operand-shifts only. In hardwired implementations the shift operation has zero hardware cost and minimum wiring costs. We name this NPT (N-Power-of-Two), where N is the number of power-of-two terms. This approach has the advantage of preserving the full dynamic range of the coefficients and limiting the number of adders necessary to make the multiplication operation (leading to low power and high speed). The disadvantage of this approach is that the transfer function of the filter does not preserve the one with original fixed-point representation. In [2] an extensive review of the power-of-two technique is presented.

In this work we use the NPT thecnique and show the gain obtained against the non-optimized version. For each architecture a VHDL description is generated. We compare the adder savings with the FPGA synthesized area – in logic elements (LE), delay and power. The key point is to verify if the savings found in the effort to reduce the number of adders is mapped to a reduction in the FPGA parameters.

4. ALGORITHM DESCRIPTION

The algorithm to select the best NPT coefficient set, based on scaling of the coefficients before the conversion to fixed point format is described below.

The algorithm for the NPT phase is Algorithm 1. It takes the floating-point coefficients calculated previously from the filter specification and generates several NPT coefficients sets in the following way. First, the floating-point coefficient set is multiplied by a scale factor, then converted to fixed point and finally to NPT. In order to convert a fixed-point representation to NPT, the N most significant nonzero bits are maintained and the remaining are zeroed. The transfer function for each NPT set generated is also calculated. This process is repeated for each discrete scale factor specified. The next step is the selection phase where we select the "optimum" coefficient set based on the characteristics of the transfer function of each set. We use the in-band ripple as a design constraint, thus excluding the transfer functions for which the entire pass band response is not within the maximum specified ripple. Finally the coefficient set whose minimum attenuation in the stop band is the maximum among all the resulting transfer functions is elected as optimum, according to these criteria. The algorithm 1 shows the NPT coefficient selection process.

Algorithm 1: NPT coefficient selection by transfer function analysis

Step 1: Obtain the floating-point coefficients, scale factors range and step, transfer function allowable ripple and the maximum number of PT terms.

Step 2: For each element in scale factor vector: generate a new set of coefficients by multiplying each coefficient in floating point by the current scale factor; make the coefficients positive and save the signal in of each coefficient in a set of signals for later optimization; get the fixed point representation of this set of coefficients; convert the fixed point coefficients to NPT; obtain a transfer function of the filter with these NPT coefficients. Add the set of coefficients and transfer function to a set of filters.

Step 3: From the set of filters, eliminate those that do not respect the in-band ripple constraint.

Step 4: From the results of Step 4, find out the coefficient set that generates a filter with the highest minimum attenuation in the stop band (from all transfer functions considered, as in Fig. 3-a) and select this set as the solution of the NPT phase.

5. METHODOLOGY

The algorithm for NPT coefficient selection by transfer function analysis (Algorithm 1 described in the Section 3)

was implemented in Matlab, aided by DSP and visualization toolboxes. A set of filter parameters configure the input file to the algorithm, resulting in the search space and the selected solution graphically. The selection of filter coefficients proceeds efficiently, and the scale step factor is kept constant, at 1/100 increments. No heuristics has to guide this scaling, since it runs fast for up to hundreds of scaling factors in filters with hundreds of taps.

Since the filter coefficients can be positive or negative, and we deal only with positive numbers, our method saves signals to be treated separately. This was helpful for the task of optimizing the multiplier block (small negative numbers have a large amount of nonzero digits). Figure 2 shows the architecture developed for this implementation, where the signals S1..N of the coefficients C1..N were saved to be control signals in the final summation block (the signal actually selects between add or subtract functions). This is a transposed direct form FIR filter. A multiplier at the output Y is needed (for 1/Scaling_factor adjust) to keep the unity gain in the pass band, since our method scaled all the filter coefficients to determine a better representation in the NPT phase. This multiplier can be eliminated if the gain of the filter is not critical.

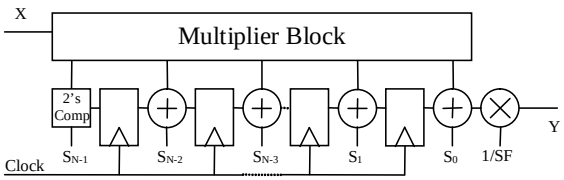


Figure 2. Architecture of the hardware description output.

The program was implemented in a way we can turn the NPT, or the CSE, or both phases off in the filter generation tool.

6. RESULTS

After implementing the algorithm described in the section 3, we analyzed the behavior of the NPT rounding technique for several filter specifications. Our experiments show that it is very hard to get more than 20dB per PT digit (a similar result was stated in [2] for CSD coefficients), and this is very dependent on the frequency response shape and on the number of taps. Fig. 3 shows graphical results for the Low Pass (LP) filter LP1 specified in Table 2.

Dotted line in Fig. 3(a) and 3(b) shows the transfer function for the FIR filter LP1 in fixed point. Solid lines in Fig. 3(a) show the transfer functions for the 2PT coefficient sets for all the 76 scale factors tested (0.5 to 2 in steps of 0.02). Fig. 3(b) compares the filter transfer function for the LP1 with fixed point coefficients and the optimized version with 2PT coefficient set selected by our method. The insert in Fig. 3(b) shows that the effect of the optimization is negligible in the in-band ripple.

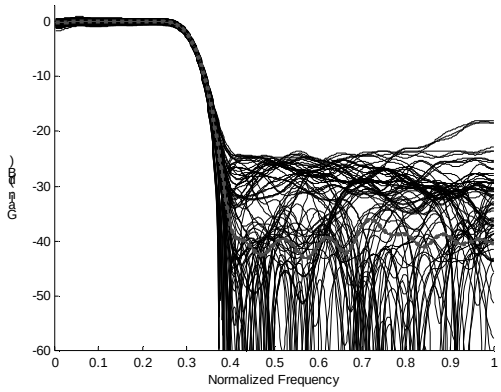
Table 1. Comparison of the coefficients before and after the NPT phase.

Table 1 shows a comparison between the original fixed-point coefficients and the reduced 2PT coefficients for LP1 FIR filter (using the same scale factor for both coefficient sets). As the coefficients are symmetrical, the table presents only the first $\lfloor N/2 \rfloor+1$ coefficients of the 49-tap filter.

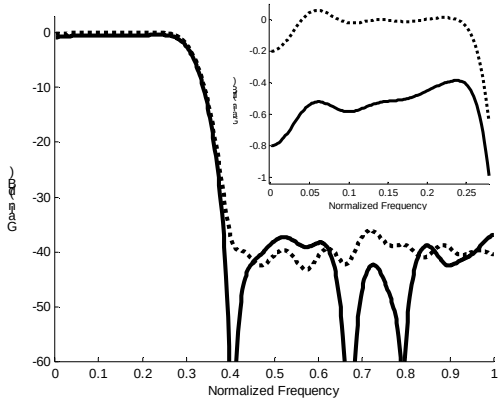
Tap	Fixed Point	Num Adders	Logic Depth	Optimized 2PT Coefficients	Num. Adders	Logic Depth
1	000000001	0	0	000000001	0	0
2	000000001	0	0	000000001	0	0
3	000000001	0	0	000000001	0	0
4	000000000	0	0	000000000	0	0
5	000000010	0	0	000000010	0	0
6	000000001	0	0	000000001	0	0
7	000000011	1	1	000000011	1	1
8	000000110	1	1	000000110	1	1
9	000000100	0	0	000000100	0	0
10	000000011	1	1	000000011	1	1
11	000001010	1	1	000001010	1	1
12	000000111	2	2	000001000	0	0
13	000000111	2	2	000001000	0	0
14	000010100	1	1	000010100	1	1
15	000010001	1	1	000010001	1	1
16	000001001	1	1	000001001	1	1
17	000100011	2	2	000100100	1	1
18	000100000	0	0	000100000	1	1
19	000001011	2	2	000001100	1	1
20	001000010	1	1	001000010	1	1
21	001001010	2	2	001001000	1	1
22	000001100	1	1	000001100	1	1
23	010101100	3	2	010100000	1	1
24	101000000	1	1	101000000	1	1
25	110011110	5	3	110000000	1	1
Total		27	3 (max)		16	1 (max)

The results presented in Table 1 show the capability of the NPT phase in reducing the number of nonzero digits in the coefficients, the number of adders needed for each coefficient and the logic depth of the adder tree. As stated in section 4, all coefficients are set positive, as the sign of

the coefficients are treated separately in the final summation block (not considered here). Note that the NPT technique not only reduces the total number of adders but also the logic depth in terms of number of adders needed to implement the multipliers, which produces a delay reduction.



(a)



(b)

Figure 3. (a) Fixed point (dotted) x 2PT for each scale factor tested (solid), and (b) fixed point (dotted) x 2PT selected (solid). Ripple in pass band (detail).

Table 2 presents the specifications for a few low pass (LP) and high pass (HP) filters used to test our methodology. The parameters have been selected to cover the 1PT to 4PT-reduced coefficients and 10- to 16-bits fixed point. Table 3 summarizes the results for the filter specifications presented in Table 2, showing the number of adders for each filter with and without the application of the optimization algorithm.

Table 4 shows that significant reduction in the number of adders is achieved by applying NPT optimization separately. The great advantage of using the NPT algorithm as described is that we greatly simplify the complexity of the multipliers by controllably modifying the coefficients with small and acceptable changes in the filter transfer function. Also the logic depth is guaranteed to be low in the NPT optimization phase. Limiting the number of power-of-two (PT) terms in each coefficient also reduces the summation tree needed to implement each multiplier [2] (as shown in Table 1 for LP1 filter), reducing the delay. Additionally, a lower logic depth can lead to a reduction in the glitching activity, hence reducing the power further. The NPT approach improve results, reducing the number of adders needed in all cases, as we expected.

A benchmark was made to verify the performance of the method for FPGA synthesis. The parameters for the filter and for the tool used to synthesize the various VHDL descriptions of the filter to a target FPGA are presented in Table 3. The optimization steps in the synthesis tool were left to its default values.

We present three performance results obtained after FPGA synthesis, namely the number of logic elements (LE), the worst-case delay, and the overall power consumption estimated with random input samples. We can observe in Table 5 a significant reduction in the number of adders and the power consumption when we make the NPT coefficient reduction. The delay and power estimates shown in Table 5 corroborate that the NPT is very effective in reducing both, while the previous effort to optimize the architectural level specification of the adder tree by the non-optimized version is less effective (consumes more power) in the FPGA implementation

Table 2. Filters used to test the proposed methodology.

Parameter	LP1	LP2	LP3	HP1	HP2
# of Taps	49	31	71	31	51
Scale Range (increment)	0.5-2 (0.02)	0.5-2 (0.02)	0.5-2 (0.02)	0.5-2 (0.02)	0.5-2 (0.02)
Bits Fixed Point	10 (sign+9)	16 (sign+15)	16 (sign+15)	12 (sign+11)	16 (sign+15)
NPT digits	2	4	3	1	4
Pass Band (normalized)	0-0.3	0-0.3	0-0.05	0.6-1	0.7-1
Max. Pass Band Ripple	0.1	0.01	0.1	0.1	0.01
Stop Band (normalized)	0.35-1	0.35-1	0.07-1	0-0.4	0-0.6
Stop Gain (dB)	-40	-60	-60	-20	-60
Window Type	Hamming	Hamming	Blackman	Hamming	Hamming

Table 3. Parameters Set for the synthesis results.

Filter Form	Transposed Direct Form
Input	16 bits samples
Output	32 bits samples
Coefficients	16 bits
FPGA	EP20K200E
Synthesis tool	Quartus II 2.2
Power estimation	500 random input vectors
Sample-rate frequency	10MHz

Table 4. Optimization results for the filters in Table 2.

Filter	**Mul.-less**		**NPT**	
	Add.	%	Add.	%
LP1	76	100	60	79
LP2	90	100	69	77
LP3	206	100	146	71
HP1	47	100	31	65
HP2	138	100	106	77
Mean		100		74

7. CONCLUSION

From the results obtained we conclude that using NPT lead to an improved, i.e. reduced, number of adders to an average of 74% (equivalent to a reduction of 26% - Table 4), compared to the non-optimized version of the multiplier-less dedicated adder tree, for which the coefficient are fixed-length and the symmetrical characteristics of FIR filters are also exploited.

The tool we developed generate the VHDL output for later targeting to an ASIC or FPGA. In this work we show FPGA synthesis results, leading to an average of 81% of the area required for synthesis of the non-optimized version. This means an average area reduction of 19%. We conclude that the NPT optimization lead to a reduction of area in FPGA synthesis similar to the reduction in the number of adders. The number of LE in the synthesized version includes the registers, which are the same in the optimized and in the non-optimized version, which leads to a percentual reduction of area smaller than the reduction obtained in the number of adders (19% in area in FPGA against 26% in the number of adders).

Another important result is the power required by the filter for its filtering work. Using NPT optimization we were able to achieve reductions to an average of 44% of power required in the non-optimized version using a random signal as input of the filter. This means a reduction of 66% in power consumption, which is a large power improvement.

The average delay in the FPGA synthesized optimized version was 73% of the non-optimized version (a reduction of 27%). This enables higher operating frequency without any architectural optimization other than the NPT coefficients.

Table 5. FIR filter synthesis results in FPGAs

Filtro	Optmization Method	VHDL		#LE		Delay		Power	
		Adders	%	#LE	%	ns	%	mW	%
LP1	Unopt.	76	100	3017	100	44	100	325	100
	NPT	60	79	2630	87	35	80	169	52
LP2	Unopt.	90	100	3428	100	66	100	1078	100
	NPT	69	77	2798	82	55	83	530	49
LP3	Unopt.	206	100	8062	100	86	100	2990	100
	NPT	146	71	5742	74	65	76	967	32
HP1	Unopt.	47	100	1976	100	43	100	258	100
	NPT	31	65	1573	80	21	49	96	37
HP2	Unopt.	138	100	5572	100	72	100	1463	100
	NPT	106	77	4649	83	56	78	765	52

We finally conclude that for FPGA synthesis, the NPT coefficient optimization lead to filters with smaller area, delay and power, which combined lead to a much better performance filter, with small and acceptable modifications in the transfer function in order to produce a coefficient set with limited non-zero bits per coefficient.

In our future works we will evaluate optimizations achievable using cannonical signed digit (CSD) representation in the coefficients and also architectural optimization techniques to reduce area, delay and power comsumption of the entire filter..

8. REFERENCES

[1] R. Pasko, P. Schaumont, V. Derudder, S. Vernalde, and D. Iuraekova, “A new algorithm for elimination of common subexpressions”, *IEEE Trans. Computer-Aided Design*, 18. (Jan 1999), 58-68.

[2] H. Samueli, “An improved search algorithm for the design of multiplier-less FIR filters with powers-of-two coefficients”, *IEE Trans. Circuits Syst.*, 36 (July 1989), 1044-1047.

[3] K-H Chen, T-D Chiueh, “Design and implementation of a reconfigurable FIR filter”, *Proc of 2003 Int. Symp. Circuits Systems, ISCAS ’03*, 3, (May 2003), 25-28.

[4] K. Hwang, “Computer arithmetic Principles, Architecture and Design”: *Wiley*, 1979.

[5] C. Lim, J. B. Evans, and B. Liu, “Decomposition of binary integers into signed power-of-two terms”, *IEEE Trans. Circuits Syst.*, 38, (June 1991) 667-672.

[6] J. Portela, E. Costa. J. Monteiro, “Optimal Combination of Number of Taps and Coefficient Bit-Width for Low Power FIR Filter Realization”, *IEEE European Conference on Circuit Theory and Design*. (Sep. 2003), 145-148.

[7] Q. Zhao, Y. A. Tadokoro, “Simple Design of FIR Filters with Powers-of-Two Coefficients”, *IEEE Transactions on Circuits and Systems*. 35, 5 (May, 1988).

[8] R. W. Hamming, “Digital Filters”, *Prentice Hall*, 3rd ed., (1989).

-