Design Optimization Techniques Evaluation for High Performance Parallel FIR Filters in FPGA

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

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

Sergio Bampi
Inst. Informatics - Univ. Fed. Rio Grande do Sul
Porto Alegre, RS – Brazil
bampi@inf.ufrgs.br

ABSTRACT

This paper presents synthesis results on a method to minimize the amount of hardware needed to implement a parallel digital finite impulse response (FIR) filters for hardwired (fixed coefficients) implementation targeted for high performance. The proposed method employ a combination of two approaches: first, the 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, followed by Common Subexpression Elimination (CSE) among multipliers. We present results for a range of different filter specifications, using Quartus II FPGA synthesis tool. We concluded that for FPGA applications, the NPT is a good approach, while CSE is worthless.

1. INTRODUCTION

Finite Impulse Response (FIR) filters are of great importance in the digital signal processing (DSP) world. Their characteristics of linear phase and feed forward implementation make it 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 number of bits in each coefficient. The most expensive block in terms of area, delay, and power in a FIR filter is the multipliers needed to implement it. In this paper we are addressing optimizations in the filter by a combination of two known approaches. First, we make use of power-of-two terms with scaling, and select 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). Also, it is found by exhaustive search a scale factor to be multiplied by all coefficients prior to its conversion to fixed point 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 as adder trees (constant coefficient multiplierless implementation), we make common subexpression elimination (CSE) by pairwise matching, generating a minimized adder tree of the whole multiplier block.

The goal of this work is verify the performance of these two algorithms when targeting a FPGA implementation in reducing the overall filter complexity. This is an extension of the work presented in [11], where only coefficient optimizations were addressed.

We present a brief review of FIR filter design in section 2 and related work on power-of-two coefficients and common subexpression elimination in section 3. In section 4 we present the employed algorithm, and in section 5 its implementation. Section 6 shows the results obtained and section 7 summarizes the conclusions and presents our proposals for future work.

2. PARALLEL FIR FILTER OVERVIEW

A FIR filter can be mathematically expressed by the equation (1) [10]:

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

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. In parallel implementations, we can have two main 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 and the last (Fig. 1b) is the transposed form of the FIR.

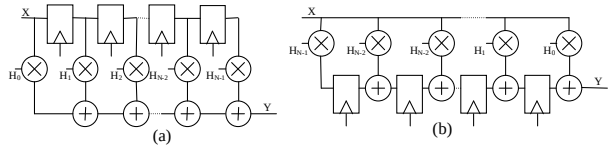


Figure 1. Parallel FIR filters in (a) direct form or (b) transposed direct form.

Both the direct form and transposed architectures of the FIR filter have the same complexity [10], but for some multiplier block optimization algorithms, the transposed form is preferred [1,2,3].

3. RELATED WORK

Several techniques for optimizing the multiplier block of parallel FIR filters were proposed in the literature. All of them consider the use the fixed-point representation and most [1-3] consider the transposed form implementation, because it is easier to obtain common sub expressions to be shared along two or more multipliers in this form. Many consider the use of some kind of signed digit (SD) representation, mainly the canonical signed digit (CSD) representation [2,3], which results in fewer non-zero digits in each coefficient, usually resulting in a smaller multiplier block. Previous research has been shown reductions of more than 50% [3] in the number of adders by using these techniques. The great advantage of these techniques is that the optimized filter has the same behavior of the original non-optimized one (i.e. same impulse response or transfer function). Other optimization techniques consist of the modification of the coefficients in order to generate sets of coefficients, which have a lower implementation, cost. Scaling and coefficient perturbations are examples of those techniques. Another approach consists of representing each coefficient as a sum of power-of-two terms and limiting the number of power-of-two terms in each coefficient [4,5,7,8]. That means the reduction of the number of bits in '1' state in each coefficient, reducing the number of adders needed to implement the multiplier for that coefficient. The best case is when we have just one power-of-two term in each coefficient, eliminating additions in the multiplier block at all, requiring operand shifting only (we are considering a hardwired implementation, where the shifting operation have no cost). 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 is not the same as obtained with the original fixed-point representation. In [4] an extensive review of the power-of-two technique is presented. In this work we use both these approaches separately and combined, showing the gain obtained against the unoptimized version, comparing the adder savings with the FPGA synthesized 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 followed by common subexpression elimination (CSE) is described below.

The algorithm will search a wide range of discrete scaling factors and store the resulting transfer function of each NPT coefficient set associated and later select the best coefficient set based on the characteristics of the transfer function. We have adopted a slightly different criteria from other published literature for selecting the best coefficient set [4,5,7,8]. We use the in-band ripple as a constraint, selecting only the transfer functions for which the entire pass band are within the specified ripple, and select the coefficient set in which the minimum attenuation in the stop band is the maximum among all the resulting transfer functions. The algorithm 1 shows the NPT coefficient selection process.

Algorithm 1: NPT coefficient selection by transfer function analysis

Step 1: Obtain FIR filter parameters: Taps; Bits; NPT elements; transfer function; pass and stop bands region; in-band ripple; Scale factors region and increment.

Step 2: Obtain the floating-point coefficients for the specified transfer function.

Step 3: For each element in scale factor vector, generate a new set of coefficients by multiplying each coefficient in floating point the current scale factor; make the coefficients positive and save the signal in of each coefficient in a set of signs for later use; 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 4: From the set of filters, eliminate those that do not respect the in-band ripple constraint.

Step 5: From the results of Step 4, find out the coefficient set that generates a filter with the highest minimum attenuation in the stop band and select this set as the solution of the NPT phase.

Step 6: Make the common subexpression elimination of the solution of the NPT phase.

This algorithm identifies the scaling factor that generates the best NPT coefficients. The Step 6 will get the coefficient set selected in Step 5 and make the common sub expression elimination task. The output is a graph of the multiplier block that can easily be used to generate a hardware description of the filter. The algorithm 2 presents the process of eliminating common subexpressions.

Algorithm 2: Common Sub Expression Elimination

Step 1: Create a matrix C_{NxW} filled with the coefficients, where W is the Width of the coefficient and N is the number of coefficients; create a matrix F_{2xW} with all values equal -1.

Step 2: Create a set of triples $X(a,b,c)$, referencing the two columns of the matrix C and the number of bits in state '1' in both bit positions of these two columns.

Step 3: Sort X by descending order of the element c

Step 4: Get the first element of the set X. If $c<1$ in this element, go to step 6

Step 5: Create a new row in matrix C with the bits that are common in both columns of the element selected in Step 4; create a new row in matrix F, making the two values referencing the number of the columns that formed this one; go to step 2

Step 6: Sweep the matrix F backwards from the last index down to the index W, generating a tree of adders. Bits in '1' in the matrix C are used to mark interconnections from the multiplier block to the summation block.

This algorithm is a variation of the algorithm proposed in [1] for binary coefficients in the sense it treats only positive numbers, leaving the signs to control the final summation block in the architecture.

5. METHODOLOGY

The algorithm for NPT coefficient selection by transfer function analysis (Algorithm 1 described in the Section 3) was implemented in Matlab and its DSP and visualization toolboxes. It takes a set of parameters from a configuration file, process the algorithm, show the search space and the selected solution graphically, and write a file with the selected filter coefficients.

Since the filter coefficients can be positive or negative, and we deal only with positive numbers, our method saves signs to be treated separately. This was helpful for the task of optimizing the multiplier block. Figure 2 shows the architecture developed for this implementation, where the signs S1..N of the coefficients C1..N were saved to be control signs in the final summation block (the signal actually selects between add or subtract functions). The algorithm for common subexpression elimination (Algorithm 2 described in the Section 3) was implemented in C.

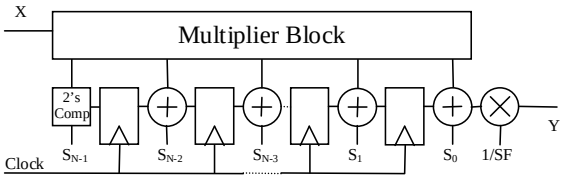


Figure 2. Architecture of the hardware description output.

This is a transposed direct form FIR filter where the multiplier block receives the input signal X, and delivers N (taps) multiplied outputs. A multiplier just before the output Y is needed if we need to maintain the unity gain in the pass band, since our method scaled all the coefficients to find a better representation in the NPT phase. This multiplier can be eliminated if the gain of the filter is not critical.

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 showed that it is very hard to get more than 20dB per PT digit (a similar result was stated in [5] 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.

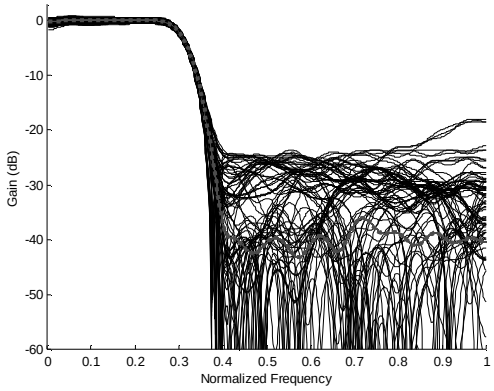
Table 1 shows a comparison between the original fixed-point coefficients and the reduced 2PT coefficients for LP1 FIR filter (using the same scale factor for both coefficient sets). As the coefficients are symmetrical, the table presents only the first $\lfloor N/2 \rfloor+1$ coefficients of the 49-tap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

The results presented in Table 1 show the capability of the NPT phase in reducing the multiplier block in terms of number of adders with small changes in the transfer function, with the methodology adopted. As stated in section 4, the sign of the coefficients are treated separately in the final summation block (not considered here), so all coefficients are positive. 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.

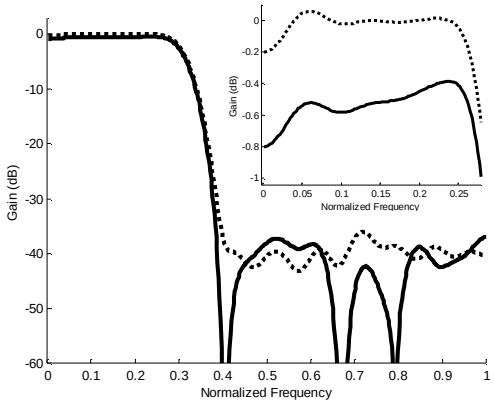
Table 1. Comparison of the coefficients before and after the NPT phase.

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 last phase of the algorithm is the common subexpression elimination. Table 2 presents the specifications some 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 4 summarizes the results for the filter specifications presented in Table 2, showing the number of adders for each case.



(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 3 shows that significant reduction in the number of adders is achieved by applying either CSE or NPT optimization separately. Our proposed methodology, combining them appropriately, improves the results even more. The great advantage of using the NPT phase 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 [6] (as shown in Table 1 for LP1 filter), reducing the delay. This delay can be exchanged for lower power consumption through the reduction of the supply voltage. Additionally, a lower logic depth can lead to a reduction in the glitching activity, hence reducing the power further.

The CSE phase improves the results by eliminating any redundancies (common subexpressions among multipliers) in the multiplier block, thus further reducing the number of adders. A grouped bar graph is showed in Figure 4.

Another benchmark was made to verify the performance of the method for FPGA synthesis. Table 3 summarizes the parameters for the filter and for the tool employed to synthesize the various VHDL descriptions of the filter to a target FPGA. All the optimization phases in the synthesis tool were left unchanged.

Table 2. Filters employed to evaluate the algorithms.

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 for the obtained 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 shows the number of adders employed to generate the multiplier block in some filters for all the combinations of the two employed approaches. We can notice in this table significant reductions in the number of adders needed to implement the multiplier block are achieved with each of the algorithms individually. When combined together, the results improve further, as we expected.

We present several design parameter results after FPGA synthesis, namely the number of logic elements (LE), the worst case delay, and the overall power consumption under random inputs. 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. In this table we can notice a few lines with the same value, indicating that the CSE phase does not lead to any improvement in the results (even generating worse results in some cases). The explanation of the seemingly poor performance of this optimization phase is that the tool we are using (Quartus II 2.2) finds out common subexpressions as redundant logic and eliminates it.

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

Filter	Mul.-less		CSE		NPT		NPT+CSE	
	Add.	%	Add.	%	Add.	%	Add.	%
LP1(fig.3)	27	100	16	59	11	41	9	33
LP2	59	100	31	53	38	64	25	42
LP3	135	100	68	50	75	56	47	35
HP1	16	100	14	88	0*	0	0*	0
HP2	87	100	50	57	55	63	37	43
Mean		100		61		44		31

*Only one PT term to compute; shift-only operation.

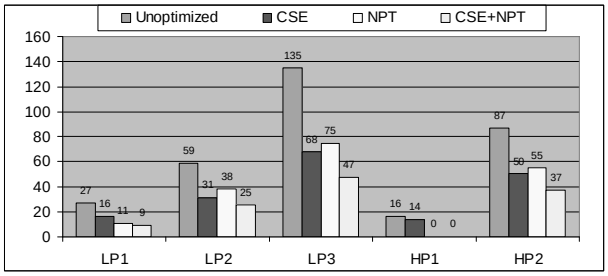


Figure 4. Combination of the two methods for the filters presented in Table 2.

The power consumption reduction stems from three factors: first, the smaller number of adders, second the reduced (and controlled) logic depth from the NPT phase reduces glitching activity, and third shorter paths may

allow for a supply voltage reduction if the same data rate is intended.

7. CONCLUSION

From the results obtained we conclude that using NPT coefficients representation leads to a great reduction in the complexity of the multiplier block of a FIR filter, with a small and controllable distortion in the transfer curve of the filter. This is done by means of selecting the appropriate fixed-point representation for the NPT conversion process by means of scaling the floating-point coefficients before its conversion to fixed point. The scale factor that generates the best results is found by exhaustive search along a discrete range of scale factors.

Table 5. FPGA synthesis results.

Filtro	Optmization Method	Adders	#LE	Delay	Power
LP1	None	27	3017	44	325
	CSE	16	3014	44	325
	NPT	11	2630	35	169
	CSE+NPT	9	2630	35	169
LP2	None	59	3428	66	1078
	CSE	31	3488	72	1224
	NPT	38	2798	55	530
	CSE+NPT	25	2918	59	570
LP3	None	135	8062	86	2990
	CSE	68	7756	76	2588
	NPT	75	5742	65	967
	CSE+NPT	47	6282	59	1138
HP1	None	16	1976	43	258
	CSE	14	2033	45	296
	NPT	0	1573	21	96
	CSE+NPT	0	1573	21	96
HP2	None	87	5572	72	1463
	CSE	50	5926	65	2118
	NPT	55	4649	56	765
	CSE+NPT	37	4882	60	952

The CSE optimization reduces the number of adders in the multiplier block by sharing subexpressions common among different multipliers. It can be combined with NPT, leading to a better reduction in the number of adders, as shown in the "Adders" column in Table 5. Although the CSE phase reduces the number of adders of the filter, the synthesized logic does not present any improvement when we use CSE, as shown in Table 5 (#LE, Delay and Power columns). The reason for that is that we use the FPGA synthesis tool in the standard options, which includes an optimization phase. It eliminates unnecessary logic, including redundant logic (common subexpressions). The optimization makes the task of the CSE algorithm and is applied to all versions of the synthesized filters, including the non-optimized one, leading to similar synthesis results whether CSE optimization is applied or not.

The Delay results in Table 5 are estimated by the synthesis tool. The restriction in the number of non-zero digits in the coefficients reduced significantly the delay, as expected.

Finally the Power results in Table 5 are estimated by the tool by simulating the extracted circuit stimulated by a random input signal. The reduction in the logic and the logic depth by the NPT optimization reduced the circuit subject to transiction and the glitching activity, leading to great reductions in power comsumption.

We conclude that NPT reduces the FIR filter area, delay and power for FPGA applications, while CSE does not lead to any improvement, as explained above.

Future work will investigate the use of Signed Digit representation and pipelined FIR optimization.

8. REFERENCES

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

[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*, (1995), 668-671

[3] 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.

[4] 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.

[5] 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.

[6] K. Hwang, "Computer arithmetic Principles, Architecture and Design": *Wiley*, 1979.

[7] 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.

[8] 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.

[9] 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).

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

[11] V. S. Rosa, S. Bampi, E. Costa, “Coefficient Optimization for High Performance Parallel FIR Filters in FPGA”, XI Workshop IBERCHIP. (Mar. 2005), 86-89.

-