Performance Evaluation of Parallel FIR Filter Optimizations in ASICs and FPGA

Vagner S. Rosa
Informatics Inst. - UFRGS
PO Box. 15064
Porto Alegre, RS, Brazil
+55(51)3316-6165
vsrosa@inf.ufrgs.Br

Eduardo Costa
Univ. Católica de Pelotas
Felix da Cunha, 412
Pelotas, RS, Brazil
+55(53)2848287
ecosta@atlas.ucpel.tche.br

Jose C. Monteiro
IST/INESC
Alves Redol, 9
Lisbon, Portugal
+351(21)3100283
jcm@inesc-id.pt

Sergio Bampi
Informatics Inst. - UFRGS
PO Box. 15064
Porto Alegre, RS, Brazil
+55(51)3316-6165
bampi@inf.ufrgs.Br

***Abstract*—This paper addresses constant coefficients Parallel FIR filter optimizations. The optimizations proposed use a combination of two approaches: the reduction of the coefficients to a limited number of Power-of-Two (PT) terms, where the maximum number of non-zero bits is set as a constraint, followed by Common Subexpression Elimination (CSE) among multipliers. Implementation results and the optimization effects in area, delay, and power for FPGAs and CMOS standard cells designs are presented. We show that it is possible to achieve area savings for both, ASIC or FPGAs implementation. Additional results for ASIC area are compared when timing constrained physical synthesis from a 0.35um CMOS cell library is performed.**

I. 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 multipliers are the most expensive blocks in terms of area, delay, and power in a FIR filter when considering custom implementation. In fact, even for a dedicated implementation of constant-coefficient multipliers, the amount of hardware needed is very high as we have several multipliers in the entire filter. In this work we are addressing optimizations in the multiplier block of FIR filters by changes in the coefficients and architectural level improvements. The goal is to optimize multiplier-less implementations and test the impact of the optimizations in these two architectural targets: FPGA and ASICs. The results for optimizations in both coefficient and architectural levels, both separately and combined, are compared against classical 10-bit, 12-bit and 16-bit arithmetic parallel multiplier-less FIR filters.

This paper is organized as follows. The next section briefly describes FIR filter design. We present a brief review of the related work on power-of-two coefficients and common subexpression elimination for FIR filters in section III. In section IV we present our FIR filter design flow. Section V shows the synthesis results obtained by Altera Quartus FPGA mapping. Synthesis results obtained by standard cell ASIC physical synthesis software from Cadence are also discussed in this section. Section VI summarizes the conclusions and presents our proposals for future work.

II. FIR FILTER DESIGN

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 by the Discrete Fourier Transform (DFT) of the required frequency transfer function, applying some known windowing method. In the sequential implementation a set of multiply-and-add (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

This work was partially supported by CNPq

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.

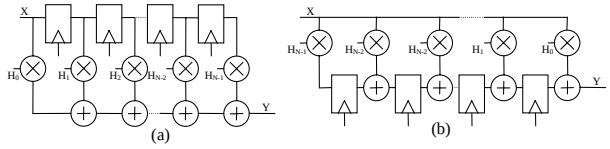

Figure 1. Parallel FIR filters in (a) direct form or (b) transposed direct form.

Both the direct form and transposed architectures of the FIR filter have the same complexity [9], but for some multiplier block optimization algorithms, the transposed form is preferred [1,2,3].

III. Related Work

Several techniques for optimizing the multiplier block of parallel FIR filters were proposed in the literature. All of them consider the use of fixed-point representation and most of them [1-3] consider the transposed form implementation. This is due to the easier way to obtain common sub expressions to be shared along two or more multipliers in this form. In terms of signed representation, some work present in literature consider the use of some kind of signed digit (SD) representation, mainly the canonical signed digit (CSD) representation [2,3]. These representations result in fewer non-zero digits in each coefficient, usually resulting in a smaller multiplier block. It 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]. This 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 present when we have just one power-of-two term in each coefficient, eliminating additions in the multiplier block at all. This requires 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 combine these approaches and evaluate the results in terms of the optimized number of architectural adders specified in the VHDL output that our technique generates. Additionally, this work shows the area and delay gains resulting from each optimization step, and proposes an optimum combination of these techniques by targeting both FPGA and ASIC physical synthesis. The key result is the possibility of gains that the NPT and CSE optimizations allow in the two different technology mapping paths.

IV. Proposed Flow

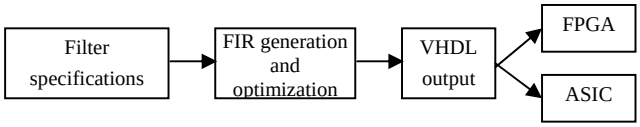

Figure 2. Synthesis Flow

The proposed filter design flow is presented in Fig. 1. The FIR generation and optimization phase was developed in this work to automatically generate a VHDL output file, starting from a parameterized filter specification. The input specification of the filter includes, for example, the transfer function requirements and the maximum number of PT terms to be used when generating the NPT. The two optimization steps employed can be turned on/off separately in the filter specification. The effects of the NPT and CSE optimizations employed separately and combined are finally tested in the FPGA and ASIC mapping done with Altera and Cadence software tools.

The algorithms employed in the NPT and CSE phases are described in [11]. They can be summarized as follows. Firstly, a floating-point coefficient set is generated using classical methods for FIR filters. Optimization by coefficient scaling is done first. The floating-point coefficient set is multiplied by several different scale factors (in 0.01 unit steps in a range from 0.5 to 1, for example) generating one new set for each scale factor. Each of these sets are converted to fixed point and then to NPT, limiting to N the number of non-zero digits (PT, Power-of-two terms) in the fixed-point binary representation. An algorithm we developed for selecting the best NPT coefficient set based on the frequency response, for a given filter specification, is then processed. The NPT phase can be bypassed, simply converting the original non-scaled coefficient set to fixed point and using it as the selected set. The selected set is then used as an input to a CSE algorithm, which eliminates all common sub-expressions among the multipliers. This optimization phase can also be bypassed. In all cases, a multiplier-less implementation of the parallel FIR filter is done. This multiplier-less non-optimized version uses only the symmetry property of FIR filter and implement the symmetrical taps (n and N-n) only once. Finally a hardware description of the entire filter is automatically generated in VHDL.

V. RESULTS

The behavior of the NPT rounding technique was analyzed for several filter specifications. The experiments we developed have shown that it is very unlikely to obtain more than 20dB attenuation from pass-band to stop-band per PT digit (a similar result was stated in [5] for CSD coefficients). This is very dependent on the frequency response characteristics and on the number of taps. In [11] we show a graphical representation of the transfer function before and after the NPT phase.

The last optimization phase of the algorithm is the common subexpression elimination. It searches the multiplier block for identical partial products (subexpressions) among two or more multipliers, generating hardware for it only once and sharing among all the multipliers needing this partial product.

Table 1 presents the general parameters used to synthesize the filters and the tools employed. Table 2 presents the FIR specifications, where low pass (LP) and high pass (HP) filters are 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 case, and the delay and area needed for FPGA implementation and ASIC implementations.

TABLE I. PARAMETERS FOR THE SYNTHESIS RESULTS OBTAINED.

Filter Form	Transposed Direct Form
Input	16 bits samples
Output	32 bits samples
Coefficients	16 bits
FPGA Chip	EP20K200E
FPGA Synthesis tool	Quartus II 2.2
ASIC Technology	AMS 0.35μm
ASIC Synthesis tool	Cadence PKS

TABLE II. SOME FILTERS 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

Looking at the results of architectural adders in Table 3 we notice significant reduction in the number of adders is achieved by applying either CSE or NPT optimization separately. Combining both optimization methods improves the results even more. The advantage of using the NPT phase is that it greatly simplifies the complexity of the multipliers by controllably modifying the coefficients, at the cost of small and acceptable changes in the filter transfer function. Limiting the number of power-of-two (PT) terms in each coefficient also reduces the summation tree needed to implement each multiplier [6], reducing the logic depth and the delay. Notice the filter HP1, where we used a 1PT representation (only one PT term) such that no adders in the multiplier block are needed.

Table 3 also shows area and delay results for FPGA and ASIC implementations. Area reduction in this case does not track the same trend as the number of architectural adders in the multiplier block, mainly because we are considering the whole filter, not just the multipliers block, as in the case of the adders results. Registers and the bottom row of adders shown in Fig. 1 are not reduced or optimized in this work. For FPGA synthesis, area was significantly reduced in terms of logic cells, when applying NPT reduction. However, no area reduction occurs (sometimes logic cells usage actually increases) when applying CSE in the VHDL generation phase. This occurs due to the fact that the commercial FPGA synthesis tool includes a logic minimization before mapping the circuits to the target FPGA. Then, this minimization includes the CSE optimization step internal to the vendor CAD. On the other hand, delay values were reduced in all examples presented. For ASIC synthesis results, using a 0.35 μm CMOS standard cell library, physical area (cell and routing areas estimated by the PKS Cadence tool) was reduced slightly, when different optimization methods were employed. Similar results were verified for delay values, except for CSE, where the delay actually increased. In fact, the logic depth of the multiplier block is increased, which arguably turns the routing task harder for the physical synthesis. This can be explained by the fact that the synthesis tool employed to map the generated VHDL description to the target chip already makes a considerable effort to optimize the code. Thus, redundant paths are eliminated and fine-tuning timing is done, as we described earlier for FPGA. We also presented in Table 3 area and delay ASIC synthesis results by considering timing constrained applications. We tried to verify the relationship between area and the filter optimization procedures, by using a timing-constrained ASIC synthesis. This required to set to all filters a delay constraint which is less than the one obtained by the fastest version of the unconstrained filter. As an example, we have LP1 filter in Table III presenting the shortest delay of 17,99ns in the NPT optimized version; we then chose 17ns as a timing-constraint for the ASIC synthesis for the same filter which was optimized with other algorithms. This value is close to the best result obtained in the unconstrained synthesis version for the LP1 filter.

TABLE III. FIR FILTER SYNTHESIS RESULTS.

Filter	Opt. Method	VHDL Description		FPGA Synthesis				ASIC unconstrained				ASIC time-constrained			
		Adders		Area		Delay		Area		Delay		Area		Delay (ns)	
		#	%	#LE	%	ns	%	mm^2	%	ns	%	mm^2	%	Obt.	Targ.
LP1	Non opt	27	100	3017	100	44	100	0.96	100	19.18	100	0.98	100	17.17	17
	CSE	16	59	3014	100	44	100	0.89	93	21.10	110	0.93	95	17.42	17
	NPT	11	41	2630	87	35	80	0.83	86	17.99	94	0.83	85	16.94	17
	NPT+CSE	9	33	2630	87	35	80	0.81	84	18.05	94	0.81	83	16.94	17
LP2	Non opt	59	100	3428	100	66	100	0.99	100	23.78	100	1.01	100	19.90	20
	CSE	31	53	3488	102	72	109	0.83	84	25.07	105	0.88	87	20.51	20
	NPT	38	64	2798	82	55	83	0.81	82	23.05	97	0.82	81	19.83	20
	NPT+CSE	25	42	2918	85	59	89	0.74	75	22.15	93	0.75	74	19.92	20
LP3	Non opt	135	100	8062	100	86	100	2.22	100	28.71	100	2.29	100	22.90	23
	CSE	68	50	7756	96	76	88	1.91	86	30.14	105	1.97	86	22.96	23
	NPT	75	56	5742	71	65	76	1.77	80	25.85	90	1.78	78	22.69	23
	NPT+CSE	47	35	6282	78	59	69	1.62	73	24.87	87	1.65	72	19.95	23
HP1	Non opt	16	100	1976	100	43	100	0.59	100	19.46	100	0.60	100	17.00	17
	CSE	14	88	2033	103	45	105	0.57	97	20.42	105	0.62	103	17.27	17
	NPT	0	0	1573	80	21	49	0.45	76	17.08	88	0.46	77	16.83	17
	NPT+CSE	0	0	1573	80	21	49	0.45	76	17.08	88	0.46	77	16.83	17
HP2	Non opt	87	100	5572	100	72	100	1.58	100	26.93	100	1.60	100	21.96	22
	CSE	50	57	5926	106	65	90	1.38	87	27.16	101	1.44	90	21.97	22
	NPT	55	63	4649	83	56	78	1.31	83	24.18	90	1.32	83	21.66	22
	NPT+CSE	37	43	4882	88	60	83	1.21	77	25.30	94	1.22	76	21.86	22

VI. CONCLUSION

From the obtained results we conclude that using NPT coefficients representation is very effective for reducing the complexity of the multiplier block of a FIR filter, keeping the side effects in the filter transfer function under control. 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 generated the best results was found by exhaustive search along a discrete range of scale factors. The NPT phase was the only effective way to reduce the area for FPGA synthesis. Comparing the FPGA and the ASIC results we notice that the optimization we developed provides larger gains in area for the ASIC synthesis. The FPGA synthesis tool does a better job finding duplicated paths (common subexpressions) than the ASIC tool employed to generate these results. This implies in a greater difference between the non-optimized version and the optimized version using both the NPT and CSE techniques. Note that we have not employed CSD (canonical signed digit) in the results yet. We hope to demonstrate in future works that all the algorithms presented apply equally well for that signed encoding.

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, J. C. Monteiro, "An Improved Synthesis Method for Low Power Fir Filters", Proc. of the 17th symposium on Integrated circuits and system design, 237-241, (Sep, 2004).