Polyphase Structures for Multiplierless Biorthogonal Filter Banks

K.A. Kotteri
Virginia Tech
kkotteri@vt.edu

A. E. Bell
Virginia Tech
abell@vt.edu

J. E. Carletta
University of Akron
carlett@uakron.edu

***Abstract*— Design techniques for high performance, fixed-point, multiplierless filter banks are presented. A technique is developed for designing a fixed-point polyphase filter structure such that the hardware is highly efficient and the image compression quality is minimally impacted by the use of fixed-point mathematics. Image compression using the biorthogonal 9/7 discrete wavelet transform provides a motivating example. The results show that implementation structure has a significant impact on the resulting hardware cost, throughput, and compression quality. We achieve a polyphase structure with twice the throughput rate of non-polyphase structures and image compression fidelity within 0.2dB of the unquantized, floating-point filters.**

I. Introduction

This work investigates fast hardware designs of biorthogonal perfect reconstruction filter banks. We demonstrate our new algorithms with the biorthogonal 9/7 wavelet filters that are employed in the JPEG2000 lossy image coder [1]. High-performance multiplierless implementations of the biorthogonal 9/7 discrete wavelet transform (DWT) on a field programmable gate array (FPGA) are described.

We employ several hardware and compression performance metrics. *Hardware cost* is measured in terms of the number of logic elements used. *Throughput*, or the number of data outputs per second that the filter can generate, relates to how long it takes to process an image. *Peak signal-to-noise ratio* (PSNR) is the quantitative measure used to compare the fidelity of a compressed image with its original.

We derive a new polyphase, cascade structure and perform a comprehensive hardware and compression comparison of the direct polyphase, cascade polyphase, direct non-polyphase, and cascade non-polyphase structures. Section II provides some background on the biorthogonal 9/7 DWT filters and on the role of the filter structure in an implementation's performance. Section III derives new polyphase structures that preserve compression performance while doubling the throughput. Section IV compares the various implementations in terms of hardware cost, throughput and PSNR. The implications of our results are presented in Section V.

II. Background

The structure of one stage of a two-channel biorthogonal filter bank is shown in Fig. 1. The symmetric, unquantized lowpass filter (LPF) coefficients, $h(n)$ and $f(n)$, for the biorthogonal 9/7 filters are listed in Table I [2].

The ultimate goal of a filter bank is perfect reconstruction (PR); the synthesis section should exactly invert the analysis section, so that the reconstructed image, $\hat{X}$, will equal the original image, X (to within an integer shift l). This is possible when aliasing and distortion are avoided. Aliasing is routinely avoided by deriving the highpass filters from the lowpass filters: $G(z) = F(-z)$ and $J(z) = -H(-z)$. PR then reduces to satisfying the no-distortion condition [3]:

$$F(z)H(z) - F(-z)H(-z) = 2z^{-l}. \quad (1)$$

The 9/7 filters meet the no-distortion condition given infinite precision; however, filters implemented in fixed-point hardware do not satisfy this condition. The key to perfect reconstruction in fixed-point hardware is to design a quantized magnitude response of the cascaded LPFs, $F'(z)H'(z)$, as close to the unquantized case as possible.

Fast, fixed-point filter approximations are multiplierless representations where multiplication is performed by shifting and adding. Each filter coefficient is represented as sums or differences of powers of two (SPT). In multiplierless filter design, the number of non-zero SPT digits T can be used to control the trade-off between hardware cost and image compression performance. T corresponds roughly to hardware cost; it represents the number of terms that must be added to perform the filter computation. In general, the higher the T, the closer the quantized PSNR values are to the unquantized PSNR values; conversely, smaller T implies less hardware but worse PSNR. Methods for the effective allocation of SPT terms to the coefficients of direct and cascade form filters have been recently examined [4]. They are summarized here.

Direct form implementations of the two LPFs, $H(z)$ and $F(z)$, require that sixteen coefficients be quantized; this alters

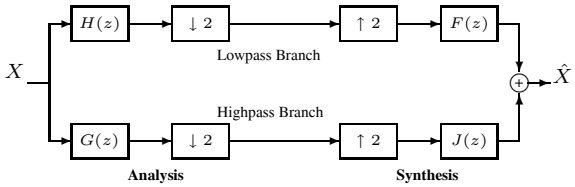

Fig. 1. Two-channel biorthogonal scalar wavelet filter bank.

TABLE I
UNQUANTIZED BIORTHOGONAL 9/7 WAVELET COEFFICIENTS.

n	$h(n)$	n	$f(n)$
0,8	0.03782845550726	0,6	-0.06453888262870
1,7	-0.02384946501956	1,5	-0.04068941760916
2,6	-0.11062440441844	2,4	0.41809227322162
3,5	0.37740285561283	3	0.78848561640558
4	0.85269867900889		

This material is based upon work supported by the National Science Foundation under Grants 9876025, 0218672, and 0217894.

the magnitude response of the unquantized cascaded LPFs, $F(z)H(z)$. In most images energy is concentrated at low frequencies; this implies that the product of the quantized LPFs, $F'(z)H'(z)$, must closely resemble the product of the unquantized LPFs, $F(z)H(z)$ at low frequencies. The "direct form with gain compensation" method [4] for allocating SPT terms to the filter coefficients accounts for this by reserving a few terms for a compensating gain G. Table II(a) lists the quantized coefficients for $T = 32$ using this method.

In addition to the magnitude response, the filter's zero locations are also important. $H(z)$ and $F(z)$ each have four zeros at $z = -1$ that are critical to image compression performance. Perturbation in the zeros at $z = -1$ causes DC leakage through the analysis highpass filter $G(z)$ and the subjective quality of the compressed and reconstructed images is degraded by the *checkerboarding* artifact. In the direct form the quantization of each coefficient moves *all* of the zeros and checkerboarding is difficult to avoid.

The cascade form implementation represents each of the two 9/7 LPFs as a cascade of two sections. The first section with coefficients $(1, 4, 6, 4, 1)$ implements the four zeros at $z = -1$, while the second implements the remaining zeros. Now the 9/7 LPFs can be written as

$$\begin{aligned} H(z) &= k_9(z^4 + 4z^3 + 6z^2 + 4z + 1) \\ &\quad \cdot(z^4 + a_1 z^3 + a_2 z^2 + a_1 z + 1) \\ F(z) &= k_7(z^4 + 4z^3 + 6z^2 + 4z + 1) \\ &\quad \cdot(z^2 + a_3 z + 1) \end{aligned}$$

where, $a_1 = -4.630463620$, $a_2 = 9.597484376$, $a_3 = -3.369536379$, $k_9 = 0.037828455$ and $k_7 = -0.064538883$. Placing the zeros at $z = -1$ in a section of their own ensures that these zeros are unaffected by quantization in the rest of the filter. Only six SPT terms are required to implement these four zeros *exactly*. The remaining SPT terms are allocated to coefficients a_1, a_2, a_3, and gain factors k_9 and k_7.

The "cascade form with z_1 compensation" method [4] for designing the cascade sections for the LPFs quantizes a_1, a_2 and a_3 such that the perturbation in zeros of the second section of the synthesis LPF offsets the perturbation in zeros of the analysis LPF. This method has been shown to significantly outperform the direct form implementation in terms of PSNR and image quality [4]. A set of quantized coefficients for this method with $T = 32$ is shown in Table II(b).

TABLE II

QUANTIZED FILTER COEFFICIENTS IN DECIMAL AND SPT: SPT IS LIKE BINARY EXCEPT $\overline{1}$ INDICATES THAT THE POWER OF TWO BE NEGATED.

	$H'(z)$		$F'(z)$	
0	0.03515625	0.00001001	-0.0625	0.000$\overline{1}$0000
1	-0.0234375	0.00000$\overline{1}$10	-0.03125	0.0000$\overline{1}$000
2	-0.125	0.00$\overline{1}$00000	0.4375	0.100$\overline{1}$0000
3	0.375	0.01100000	0.78515625	0.11001001
4	0.8125	0.11010000	0.4375	0.100$\overline{1}$0000
5	0.375	0.01100000	-0.03125	0.0000$\overline{1}$000
6	-0.125	0.00$\overline{1}$00000	-0.0625	0.000$\overline{1}$0000
7	-0.0234375	0.00000$\overline{1}$10		
8	0.03515625	0.00001001		
K	-		-	-
G	-		1.0166015625	1.0000010001

(a) direct with gain compensation, $T = 32$.

	$H'(z)$		$F'(z)$	
S	1	001	1	001
E	4	100	4	100
C	6	110	6	110
	4	100	4	100
1	1	001	1	001
S	1	0001.0000	-1	0$\overline{1}$.0000
E	-4.625	0$\overline{1}$00.$\overline{1}$0$\overline{1}$0	3.3125	11.0101
C	9.6875	1010.0$\overline{1}$0$\overline{1}$	-1	0$\overline{1}$.0000
	-4.625	0$\overline{1}$00.$\overline{1}$0$\overline{1}$0		
2	1	0001.0000		
K	0.0390625	0.0000101	0.0625	(simple shift)
G	-		-	

(b) cascade with z_1 compensation, $T = 32$.

III. POLYPHASE STRUCTURES

The filter bank structure shown in Fig. 1 is inefficient in terms of data throughput. In the analysis stage, the downsampling operation follows the filtering, and half the samples just computed by the filters are discarded. Thus, the output rate of the analysis stage is half the input rate. In the synthesis stage, the upsampling operation precedes the filtering operation. Here, the filters must operate at double the input rate; however, at any given time, half of the multipliers in the synthesis filters are multiplying zeros. Thus, half of the mathematical operations are wasted in a *non-polyphase structure*. A *polyphase structure* does not perform the wasted operations; consequently it doubles the throughput of a non-polyphase structure.

The basic block of the analysis part of a filter bank is a filter $h[n] = (h_0, h_1, h_2, h_3, h_4, h_5, \ldots) \leftrightarrow H(z)$ followed by a downsampling operation, as shown in Fig. 2a. Separating the odd and even powers in the transfer function we have

$$\begin{aligned} H(z) &= h_0 + h_1 z^{-1} + h_2 z^{-2} + h_3 z^{-3} + h_4 z^{-4} + \ldots \\ &= (h_0 + h_2 z^{-2} + h_4 z^{-4} + \ldots) + \\ &\quad z^{-1}(h_1 + h_3 z^{-2} + h_5 z^{-4} + \ldots) \\ &= H_e(z^2) + z^{-1} H_o(z^2) \end{aligned} \quad (2)$$

where,

$$\begin{aligned} H_e(z) &\leftrightarrow (h_0, h_2, h_4, \ldots) \quad \text{(even phase of } h[n]\text{)} \\ H_o(z) &\leftrightarrow (h_1, h_3, h_5, \ldots) \quad \text{(odd phase of } h[n]\text{).} \end{aligned}$$

Once the even and odd powers have been separated, the order of the downsampling operation and the filter can be exchanged, according to the first Noble identity [3]. The resulting polyphase filter structure, shown in Fig. 2b, is computationally equivalent to the original structure in Fig. 2a, but much more efficient: it downsamples the input stream so that only the necessary computations are performed.

A similar transformation applies to the synthesis side of the filter bank. In the following subsections, we describe polyphase structures for the direct form, the cascade form and the hybrid direct-cascade form.

A. Direct Polyphase Structure

The polyphase idea can be applied to the direct implementation of the 9/7 filter bank simply by splitting each filter into its even and odd phases. Applying the polyphase idea does not change the coefficients, so a direct form polyphase implementation (direct-poly) requires the same total number of SPT terms as the direct form without polyphase (direct-no-poly). The advantage of a direct-poly implementation, then, is that the throughput is doubled with no significant change in hardware cost. The direct-poly and direct-no-poly structures have the same effect on image compression quality. In particular, the zeros at $z = -1$ will not be preserved and reconstructed images will exhibit the checkerboard artifact.

B. Cascade Polyphase Structure

Fig. 3a shows the cascade structure (two sections $H_1(z)$ and $H_2(z)$ and a gain of K) followed by the downsampling operation. In applying the polyphase idea to the cascade structure, a problem is encountered in the polyphase expansion of $H_1(z)$, since a downsampling operation cannot exchange places with a unit delay. We derive a polyphase structure for the cascade form by instead considering the simultaneous polyphase representation of both cascaded sections, $H_1(z)$ and $H_2(z)$. By rearranging the equations, the delays can be factored out, and then the Noble identity can be employed:

$$\begin{aligned} H(z) &= H_1(z)H_2(z) \\ &= [H_{1e}(z^2) + z^{-1}H_{1o}(z^2)][H_{2e}(z^2) + z^{-1}H_{2o}(z^2)] \\ &= H_{1e}(z^2)H_{2e}(z^2) + \\ &\quad z^{-1}[H_{1o}(z^2)H_{2e}(z^2) + H_{2o}(z^2)H_{1e}(z^2)] + \\ &\quad z^{-2}H_{1o}(z^2)H_{2o}(z^2). \end{aligned} \tag{3}$$

The result is the cascade-poly form in Fig. 3b. The cascade-poly form doubles the throughput of the original cascade-no-poly form while still preserving the zeros at $z = -1$. However, since each polyphase filter appears twice in the structure, the cascade-poly implementation requires almost double the hardware of the original non-polyphase cascade form. For this reason, we do not consider this implementation further; we look instead for an alternative implementation, capable of maintaining the zeros at $z = -1$ and doubling the throughput without doubling the hardware.

The underlying idea here is to use a direct-poly structure, because it allows for double the throughput without an increase in hardware. At the same time, we want to preserve the zeros at $z = -1$. This we achieve by using coefficients that can be represented exactly in SPT format, and for which the original zeros at $z = -1$ are unperturbed.

We start with the quantized coefficients from the "cascade with z_1 compensation" method (shown in Table II(b)), and convolve the two sections to make a single section. This single section is mathematically equivalent to the original cascade, but can be implemented in a direct form polyphase implementation. The normalizing gain factor, K, remains a separate stage. The resulting coefficients are shown in Table III. We refer to this structure as the direct-cascade-poly form,

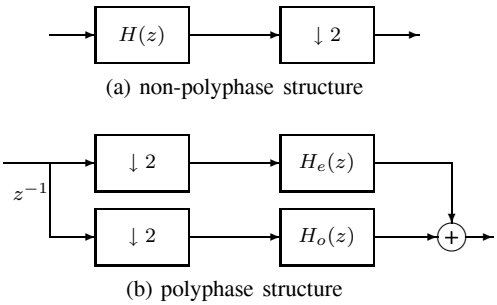

Fig. 2. Direct form filter, analysis side.

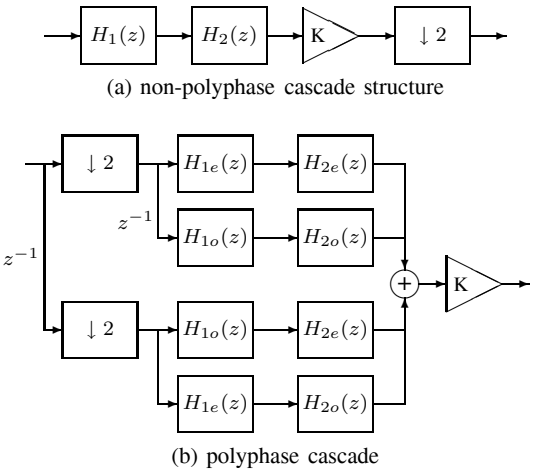

Fig. 3. Cascade form filter, analysis side.

since it is equivalent in compression performance to the cascade, but is a direct polyphase structure.

The number of SPT terms required for the direct-cascade-poly form depends on the particular coefficients; in general, the coefficients for the direct-cascade-poly structure will require more terms than the original cascade-no-poly coefficients, but significantly fewer terms than the cascade-poly structure. For our example, the cascade-no-poly structure required $T = 32$ SPT terms, the direct-cascade-poly structure required $T = 45$, and the cascade-poly structure required $T = 62$. The direct-cascade-poly design method results in double the throughput of a polyphase structure, with only a moderate increase in hardware cost, while preserving image compression quality.

IV. Results

The results represent four combinations of filter coefficient quantization technique, form and structure: a direct form with gain compensation in both non-polyphase and polyphase structures (direct-no-poly and direct-poly), a cascade form with z_1 compensation in a non-polyphase structure (cascade-no-poly), and a direct form that effectively implements cascade with z_1 compensation in a polyphase structure (direct-cascade-poly).

We first compare image processing quality. The two quantized filter coefficient sets from Table II were used to compute a 5-level, non-expansive symmetric extension DWT of 3

TABLE III

DIRECT-CASCADE-POLY COEFFICIENTS OBTAINED BY CONVOLVING THE COEFFICIENTS IN TABLE II(B), $T = 45$.

	$H'(z)$		$F'(z)$	
0	1	00001.0000	-1	$000\overline{1}.0000$
1	-0.625	$00000.\overline{1}0\overline{1}0$	-0.6875	$0000.\overline{1}0\overline{11}$
2	-2.8125	$000\overline{11}.010\overline{1}$	6.25	0110.0100
3	10.375	01010.0110	11.875	$1100.00\overline{1}0$
4	23.125	$1100\overline{1}.0010$	6.25	0110.0100
5	10.375	01010.0110	-0.6875	$0000.\overline{1}0\overline{11}$
6	-2.8125	$000\overline{11}.010\overline{1}$	-1	$000\overline{1}.0000$
7	-0.625	$00000.\overline{1}0\overline{1}0$		
8	1	00001.0000		
K	0.0390625	0.0000101	0.0625	(simple shift)
G	-		-	

TABLE IV

PSNR VALUES FOR IMAGES COMPRESSED USING THE QUANTIZED FILTERS IN TABLE II.

		unquantized	quantized	
Image Name	Comp. Ratio	(floating point)	Direct w/ gain comp.	Cascade w/ z_1 comp.
Mandrill	8:1	28.52	28.47	28.49
	16:1	24.96	24.93	24.96
	32:1	22.81	22.76	22.79
	64:1	21.36	21.26	21.47
Boat	8:1	38.21	36.98	38.22
	16:1	33.66	33.08	33.57
	32:1	30.28	29.95	30.12
	64:1	27.70	27.52	27.51
Peppers	8:1	37.61	36.29	37.73
	16:1	35.54	34.48	35.50
	32:1	33.03	32.30	32.98
	64:1	30.17	29.59	30.12

different grayscale images. Four compression ratios (8:1, 16:1, 32:1, and 64:1) were examined. The PSNR values are shown in Table IV. Note that the direct-no-poly and direct-poly are mathematically equivalent; both give the results noted in the column marked "direct with gain compensation". Similarly, the cascade-no-poly and direct-cascade-poly systems both give the results in the "cascade with z_1 compensation" column. The cascade-no-poly and direct-cascade-poly structures result in quantized PSNR values close to the unquantized PSNR values and significantly better qualitative results since there is no checkerboarding.

The four structures were implemented in hardware using filter synthesis software, written in C, to generate synthesizable VHDL filter descriptions. The software automatically chooses appropriate bit widths for internal signals such that errors due to truncation and overflow are completely avoided. Hardware performance was evaluated by synthesizing the filters for an Altera FPGA. The Altera Quartus version 2.1 software package is used for logic synthesis, placement and routing on an Altera EPF20K30EFC144-1X FPGA, a member of the APEX family. The Quartus software is used to analyze critical path delays, and to determine hardware size, measured here in terms of number of Altera logic elements.

TABLE V

HARDWARE PERFORMANCE FOR THE LOWPASS BRANCH OF FIGURE 1.

	Non-Polyphase		Polyphase	
	direct-no-poly	cascade-no-poly	direct-poly	direct-cascade-poly
SPT terms	32	32	32	45
size (logic cells)	999	1125	1474	1465
full rate (MHz)	80.09	74.94	124.26	158.80
half rate (MHz)	40.05	37.47	62.13	79.40

Table V summarizes the hardware performance of the low-pass branch of the filter bank: $H'(z)$ and $F'(z)$. For the non-polyphase filters, the cascade form filter is slightly larger than the direct form filter. Even though both systems have $T = 32$, breaking a filter into sections implies a larger total number of filter coefficients—and therefore more registers. Both systems operate at approximately the same rate.

The polyphase systems have roughly twice the throughput of the non-polyphase systems. The direct-cascade-poly system, with $T = 45$, is both smaller and faster than the direct-poly system, with $T = 32$. This is because T (essentially the number of word-level adders in the system) is not a direct measure of hardware; the bit widths of the word-level adders must also be considered. A significantly larger number of bits must be carried in the direct-poly system to avoid truncation effects. The direct-cascade-poly structure is clearly the superior hardware implementation in terms of throughput and compression quality.

V. CONCLUSIONS

The underlying structure of a biorthogonal filter bank plays a crucial role not only in the filter bank's performance and cost from the hardware perspective, but also in its performance from the image compression perspective. A polyphase filter bank structure is highly desirable from a hardware performance perspective because of its high throughput. The cascade form is highly desirable from an image compression perspective, but it results in exorbitant hardware cost if represented as a cascade polyphase structure. The proposed direct-cascade-poly design technique shows how to obtain a polyphase structure that requires only a moderate amount of hardware while preserving compression quality.

REFERENCES

[1] *ITU T.800: JPEG2000 Image Coding System Part I.* ITU Standard, July 2002.

[2] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, "Image coding using wavelet transform," *IEEE Trans. Image Processing*, vol. 1, no. 2, pp. 205–220, April 1992.

[3] G. Strang and T. Nguyen, *Wavelets and Filter Banks*, 1st ed. Wellesley MA: Wellesley-Cambridge Press, 1996.

[4] K. A. Kotteri, A. E. Bell, and J. E. Carletta, "Design of multiplierless, high-performance, wavelet filter banks with image compression applications," *IEEE Trans. Circuits Syst. I*, vol. 51, no. 2, to appear in Feb 2004.