DSP PERFORMANCE COMPARISON BETWEEN LIFTING AND FILTER BANKS FOR IMAGE CODING

S. Gnavi, B. Penna, M. Grangetto, E. Magli, G. Olmo

Dipartimento di Elettronica - Politecnico di Torino
Corso Duca degli Abruzzi 24 - 10129 Torino - Italy
Ph.: +39-011-5644195 - Fax: +39-011-5644099
gnavi(penna)@mail.tlc.polito.it
grangetto(magli,olmo)@polito.it

ABSTRACT

The lifting scheme is a very well-known computationally efficient alternative to the filter bank scheme for evaluating the discrete wavelet transform of signals and images. However, the actual computational saving is still a matter of debate. On one hand, theoretical results in the literature report an asymptotic upper-bound of two for very long wavelet filters. On the other hand, it is worth wondering to what extent the architecture of the processor used can actually bias this gain. In this paper we tackle this problem from an implementation perspective, and profile the execution time of the two algorithms on a digital signal processor. Both the real-valued and the integer versions of the wavelet transform are considered. The quantitative results are used to gain some insight on the way the processor architecture affects the algorithms.

1. INTRODUCTION

The discrete wavelet transform (DWT) [1] is a widely used signal processing tool in many fields, mainly due to its ability to capture most of signal energy within few transform coefficients. In particular, a large body of research has been devoted to the application of the DWT to the image compression problem (see e.g. [2, 3]). This effort has been noteworthily successful; as a consequence, the DWT has been selected as the transform coding kernel for the forthcoming JPEG2000 image coding standard [4].

The DWT has been traditionally implemented by means of the filter bank scheme (FBS) [1]. However, Sweldens has proposed an alternative framework, called Lifting Scheme (LS) [5], to compute the DWT. The LS has at least two advantages with respect to the FBS. The former is that, depending on the type and length of the wavelet filter, the LS requires down to one half of the number of operations needed by the FBS [5]. The latter is that it allows to compute a (non-linear) integer wavelet transform (IWT) that maps integers to integers, thus permitting to attain lossless compression even in finite precision arithmetic [6].

The actual computational saving of the LS with respect to the FBS is still a matter of debate. In his paper [5], Sweldens derives a bound of two on the LS gain under fairly general assumptions. It is worth remarking that, even though this bounds holds for a large number of filters, including all the filters dealt with in this article, Reichel has recently proved [7] that, for filters designed under less restrictive assumptions that in [5], the actual LS gain can reach up to four. On the other hand, the LS gain in a real scenario may heavily depend on the architecture of the processor used for the DWT computation.

This paper addresses the signal processing issue of LS and FBS complexity, which is investigated from a mere implementation perspective. In particular, it presents a performance comparison between the LS and the FBS using several filters of practical interest, addressing both the DWT and the IWT. For all filters considered, the experimental results are compared with the theoretically derived figures, which foresee a bound of two on the maximum LS gain. The addressed implementation is realized on a digital signal processor (DSP), which is very often employed in both multimedia and scientific applications. This implementation matches the strong interest in the development of reconfigurable hardware by means of mixed DSP/FPGA architectures, for reusable multimedia intellectual property blocks [8]. The DWT implementation on a DSP can thus be thought of as part of a multimedia library upon which complex image and video processing systems can be built.

This paper is organized as follows. In Sect. 2 we briefly review the FBS and LS, and outline our reference implementation. In Sect. 3 experimental results are shown on the comparison between LS and FBS. Finally, in Sect. 4 conclusions are drawn.

The authors are with the *Signal Analysis and Simulation* group at *Politecnico di Torino.* URL: www.helinet.polito.it/sasgroup

2. WAVELET TRANSFORM

The two main algorithms used to compute the DWT are the LS and the FBS, which are briefly reviewed in Sect. 2.1 and 2.2 respectively.

2.1. Filter bank scheme

The block diagram of Fig. 1 shows two iterations of the FBS used to compute the DWT of a one dimensional signal. The input sequence is highpass and lowpass filtered by the analysis filters $\widetilde{H}(z)$ and $\widetilde{G}(z)$; then the two resulting sequences are downsampled by a factor two. Iterating the procedure on the lowpass branch yields the signal decomposition on more than one level. To compute the two-dimensional transform of an image, these operations should be performed both rowwise and columnwise.

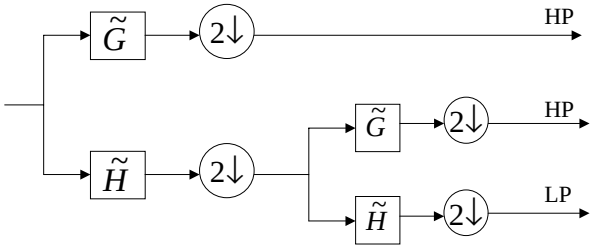

Fig. 1. Block diagram of the FBS

2.2. Lifting Scheme

The LS performs a sequence of so-called primal and dual lifting steps, as reported in the block diagram of Fig. 2. Each lifting step consists in a filtering operation on the even or odd signal samples. From the mathematical standpoint, this cascade of lifting steps is obtained by factorizing the polyphase matrix representing the discrete time filter used to compute the DWT (see [1] and [5]). The LS can be easily modified to compute an IWT, by simply rounding off the output of the $s_i(z)$ and $t_i(z)$ filters before adding or subtracting, as shown in Fig. 2 [6].

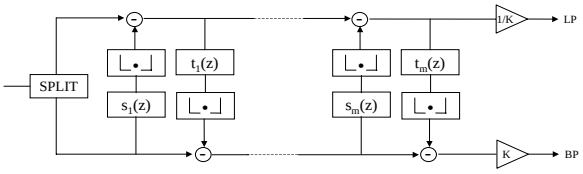

Fig. 2. Block diagram of the LS

Even though the LS and FBS yield the same results in the DWT computation, their computational cost is different. In order to compare the two algorithms one can use the number of multiplications and sums required to output a pair of samples, one on the lowpass and one on the highpass branch. As shown in [5], the cost of LS tends, asymptotically for long filters, to one half of the cost of the standard algorithm. The theoretical gains for the filters considered in this work are reported in Sect. 2.3.

2.3. Reference implementation

In order to profile the execution time of the two different algorithms and compare them, the FBS and LS have been implemented on a floating-point Texas Instruments TMS320 C6711 DSP board. This device is efficiently programmed in C language, because the compiler is able to perform the assembler optimization task very efficiently. In particular, in order to perform a fair comparison, the same convolution routine has been used for the FBS and for the lifting steps. The filters employed in the implementation are:

- LeGall I(5,3) (LEG(5,3) in the following);
- Daubechies (9,7) (DB(9,7) in the following);
- Sweldens (13,7) (SWE(13,7) in the following).

It is worth noticing that the first two filters are explicitly supported by JPEG2000 Part I [4] for the reversible and non-reversible decomposition respectively. The third filter is of interest because it has a larger number of taps, and can hence be used to test the asymptotical LS gain with respect to the FBS. The selection of the factorization of the wavelet filters, to be used in the LS, has also been made following the directives of the JPEG2000 standard for the LEG(5,3) and DB(9,7) filters; as to the SWE(13,7) filter, the factorization reported in [5] has been used.

The formulae [5] used to compute the cost of the two algorithms for the filters used in this work are reported in Tab. 1. Here $|h|$ and $|g|$ are the degree of the lowpass and highpass filter; in the case that $|h|$ and $|g|$ are even, we set $|h| = 2N$ and $|g| = 2M$. Note that the SWE(13,7) filter, being an interpolating one, has a different formula, which also involves the number of vanishing moments $\tilde{N}$.

Filter	Standard algorithm	Lifting scheme
LEG(5,3)	$4(N+M)+2$	$2(N+M+2)$
DB(9,7)	$4(N+M)+2$	$2(N+M+2)$
SWE(13,7)	$3(N+\tilde{N})-2$	$3/2(N+\tilde{N})$

Table 1. Computational cost (number of multiplications plus sums) of lifting versus filter banks

In order to achieve perfect reconstruction, it is necessary to perform boundary extension at the borders of the

input signal. In this work we have considered symmetric extension, which mirrors the signal samples outside the signal support. If used with biorthogonal symmetric filters, it allows to exactly reconstruct the image also at its borders.

3. EXPERIMENTAL RESULTS

The results reported in this section refer to the time needed to perform one level of transform on one image row, by using the LS and the FBS respectively. The results have been parameterized on the length of the input data vector. It is worth noticing that performing the transform on an image also involves columnwise filtering. However, it has been found that the time needed for the columnwise filtering is the same as for the rowwise one. This behavior is justified by the efficient memory access allowed by the cache memory. When one datum is accessed in the external memory, the DSP loads into the on-chip cache eight consecutive floating-point numbers. This implies that, in order to load one complete row, the number of accesses is only one eighth of the number of coefficients of that row. As for columnwise filtering, once the first column has been loaded, the subsequent seven ones are already present in the cache memory, provided that this memory is large enough. Therefore the memory access time to image rows and columns turns out to be the same on average.

The absolute running time of the LS (both in the DWT and IWT mode) and the FBS, related to the computation of the one-level transform on one row, is shown in Fig. 3, 4 and 5. On the basis of these figures some considerations can be made.

First of all, it is possible to estimate the number of images per second that can be processed by the algorithms. The 2-D one level transform on a 256x256 gray-scale image, employing the LEG(5,3) filter and symmetric extension, can support a rate of between 7 and 8 images per second with the LS. On the other hand, the filter bank scheme only sustains between 4 and 5 images per second. Note that the time needed to compute the integer transform is from 10% to 25% larger than that required for the DWT using the LS. This is not surprising, since rounding off each filter output is an operation which can hardly be optimized by the compiler.

In Sect. 2.2 it has been stated that the LS requires asymptotically half the number of operations with respect to the FBS. In order to facilitate the comparison, the absolute running times have been reorganized in Tab. 2, which reports the ratios between the measured running times of the two algorithms. Moreover, the expected theoretical ratios according to the formulas in Tab. 1 are also shown. It can be noticed that the measured values are slightly different from the theoretical ones.

In order to explain the discrepancy between theoretical and practical gain, the effect of the architecture has been analyzed. It has been found that the architecture of this, as well as other DSPs, can have a strong impact on the execution of the two algorithms. In particular, the DSP used in this work has a very efficient pipeline, which can dispatch 8 parallel instructions per cycle. Parallel instructions proceed simultaneously through each pipeline phase, whereas serial instructions proceed through the pipeline with a fixed relative phase difference between instructions. Every time a jump to an instruction not belonging to the pipeline occurs, the pipeline must be emptied and reloaded. This means that the convolution with long filters turns out to be more easily optimized than several convolutions with short filters. In fact, in this latter case the pipeline contents need to be frequently updated. Thus the theoretical gain that comes from the fewer number of operations in the LS may be partially lost. As for the wavelet filters considered in this work, the following remarks can be made. The DB(9,7) filter is fac-

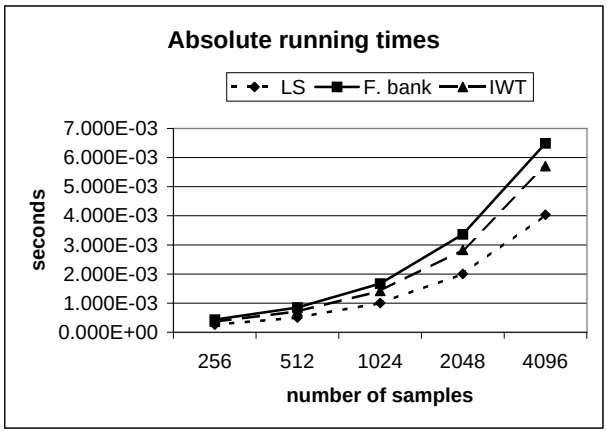

Fig. 3. LEG(5,3): absolute running times

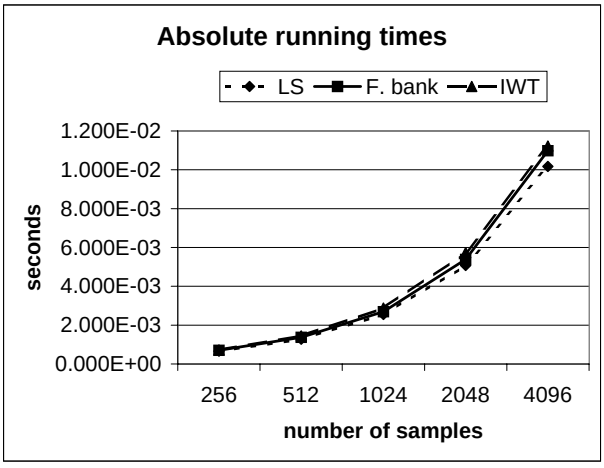

Fig. 4. DB(9,7): absolute running times

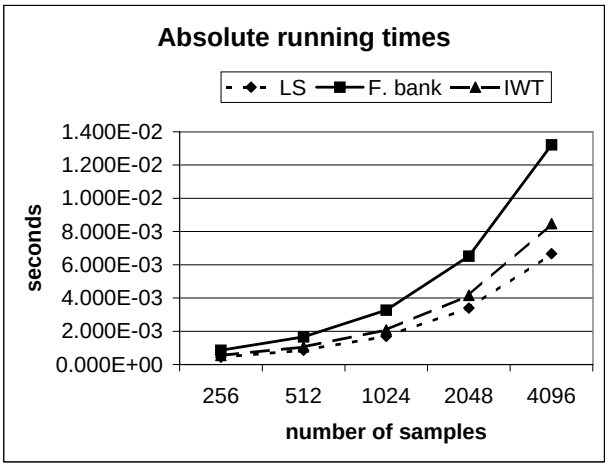

Fig. 5. SWE(13,7): absolute running times

FBS running time / LS running time			
Samples	LEG(5,3)	DB(9,7)	SWE(13,7)
256	1.650	1.069	1.969
512	1.678	1.077	1.937
1024	1.657	1.058	1.929
2048	1.679	1.062	1.917
4096	1.607	1.080	1.982
Expected theoretical value			
	1.4	1.666	1.833

Table 2. Ratios between the running times of FBS and LS

torized into four 2-tap filters (plus scaling), which make the compiler optimization less effective; this explains why the actual performance of this filter with the LS is significantly worse than expected. Conversely, the SWE(13,7) filter allows to obtain the best results: even though the filter is long, its factorization consists of only 2 filters with 4 taps each (plus scaling)[1].

In summary, on the basis of these results, it can be inferred that the practical LS gain is generally in accordance with the results in [5]. However, it turns out that the DSP architecture can heavily bias the results, privileging the FBS in the case of long filters which are factorized by the Euclidean algorithm into a long sequence of short convolution kernels.

[1] It must be noticed that the bound of two is computed in [5] under the worst case assumption of the longest factorization, which does not hold for the factorization of the SWE(13,7) filter used in our tests. This means that the bound might be exceeded, as we have found with regard to this filter in a fixed-point implementation which is not presented in this paper.

4. CONCLUSIONS

In this paper we have addressed the comparison between two algorithms used to compute the wavelet transform, i.e. the FBS and the LS, considering also the integer-transform option. We have run the algorithms on a TI TMS320C6711 DSP, reporting comparative results on their execution times, and analyzing their behavior with respect of the architectural features of the processor. We have found that the LS always runs faster than the FBS. However, the gain differs from the theoretical results in [5]. In fact, the pipelined architecture of the DSP tends to optimize more efficiently convolutions with long filters, which are typical of the FBS. On the other hand, the LS gain is higher for long filters. Thus the actual gain in a real-world scenario heavily depends on the number and length of the factorized filters used in the LS to reproduce the wavelet kernel.

5. REFERENCES

[1] M. Vetterli, J. Kovačević, "Wavelets and subband coding", *Prentice Hall PTR, Englewood Cliffs, New Jersey*, 1995

[2] M. Antonini, M. Barlaud, P. Mathieu, I. Daubechies, "Image coding using wavelet transform", *IEEE Transactions on image processing*, vol. 1, Apr. 1992, pp. 205-230

[3] D. Lazar, A. Averbuch, "Wavelet-based video coder via bit allocation", *IEEE Transactions on Circuits and Systems for Video Technology*, vol.11, n.7, July 2001, pp. 815-832

[4] Document ISO/IEC 15444-1 "JPEG2000 Final Committee Draft Version 1.0, 16 March 2000", available at URL `www.jpeg.org`

[5] I. Daubechies, W. Sweldens, "Factoring Wavelet Transforms into Lifting Steps", *J. Fourier Anal. Appl.*, Vol. 4, Nr. 3, pp. 247-269, 1998

[6] R. C. Calderbank, I. Daubechies, W. Sweldens, and B. Yeo, "Wavelet Transforms that Map Integers to Integers", *Applied and Computational Harmonic Analysis*, vol. 5, n. 3, pp. 332-369, 1998

[7] J. Reichel, "On the arithmetic and bandwidth complexity of the lifting scheme", *Proc. of ICIP 2001 - IEEE International Conference on Image Processing*, Thessaloniki, Greece, Oct. 2001

[8] J. Eyre, J. Bier, "The evolution of DSP processors", *IEEE Signal Processing Magazine*, vol. 17, n. 2, Mar. 2000, pp. 43-51