A wavelet core for video processing

Camille Diou, Lionel Torres, Michel Robert

A WAVELET CORE FOR VIDEO PROCESSING

Camille Diou, Lionel Torres, Michel Robert

LIRMM, UMR UM2-CNRS C55060,
161 rue Ada, 34392 Montpellier Cedex 5, FRANCE
Phone: (+33) 4 67 41 85 75, Fax: (+33) 4 67 41 85 00,
Email: <diou, torres, robert>@lirmm.fr

ABSTRACT

The wavelet transform appears to be an efficient tool for image processing. However, the pyramid algorithm remains silicon area costly, essentially because of its memory needs, and depending on the size of filters used. This paper proposes a new implementation of the wavelet transform using the lifting scheme. Using an improved method of managing the data flow and the RAM, this method proposes many improvements such as in-place calculation, small memory needs, and easy inverse transform.

Keywords Image processing, lifting scheme, multi-resolution analysis, wavelet core, wavelet transform.

1. INTRODUCTION

In compression applications using the wavelet transform, in many image processing systems that rely on wavelet algorithms (segmentation, edge detection), the processing time is an important feature and a high speed (or real-time) treatment cannot ever be reached with software implementations. A good solution is to implement a complete system and its memory on a chip, within the data acquisition hardware (System on Chip).

This paper shows a comparison between the MALLAT's pyramid algorithm (PA) [1] and the lifting scheme (LS), a new method for creating wavelets [3, 4, 6]. Next, we present the pyramid algorithm's implementation based on filter banks, and our architecture based on the lifting scheme. This architecture presents many advantages as a good integration, high frequency processing and low memory needs.

2. WAVELET TRANSFORM AND MULTI-RESOLUTION ANALYSIS

The multi-resolution analysis [1] uses two functions to project a signal on two spaces. The wavelet function extracts the details (high frequency signal) and the scaling function keeps the approximation (low frequency signal). By translating and dilating the wavelet, we can analyze the signal all over the time and at different resolution levels.

2.1. Filter banks and pyramid algorithm (PA)

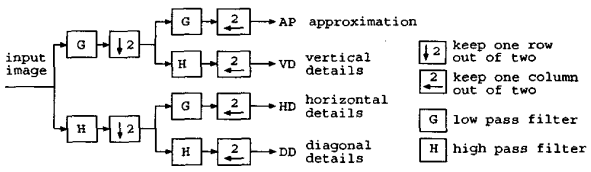

Fig. 1. Wavelet transform image decomposition

The image is filtered by both high-pass and low-pass filters along horizontal direction, then subsampled to give two sub-images that contain, respectively, an approximation of the original image and its horizontal details. This scheme is re-applied on the two sub-images along the vertical direction, giving us the three horizontal, vertical and diagonal details sub-images, and the 2-D approximated sub-image. Figure 1 shows the 2-D MALLAT wavelet transform.

To get a multi-resolution analysis, the approximation signal (low-pass band) is reintroduced into another filter banks set connected in a cascade way with the first one: this is the pyramid algorithm.

2.2. The lifting scheme (LS)

The lifting scheme algorithm [3, 4, 5, 6] presents many advantages compared to the filter banks [5]:

- all calculations can be performed in-place for an important memory saving;
- the algorithm shows an inherent SIMD parallelism at all scales;
- the inverse transform is obtained easily from the direct transform.

2.2.1. Transform

The lifting scheme is composed of several elementary steps that when applied successively construct a predefined wavelet. When using the Cohen-Daubechies-Feauveau 2-2 biorthogonal wavelet [2], the number of steps is reduced to the three following (see Figure 2):

- **split:** separates the signal into 2 parts. The more correlated the sub-signals are, the better the predict;
- **predict:** the first set is used to predict the second one, according to a defined function; the difference between the predicted set and the original one is kept as the detail signal;
- **update:** the detail signal is used to update the untouched set, in order to keep the average value of the signal.

2.2.2. *Inverse transform*

The inverse transform scheme contains the same steps (in this case: undo update, undo predict and merge) which are obtained by reversing the order of the operations and by changing the signs of the operators, as shown in Figure 2, right.

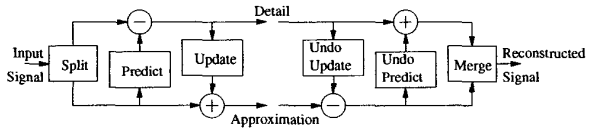

Fig. 2. Signal decomposition and recomposition with lifting

3. ARCHITECTURES

We have seen in the previous sections that the lifting scheme presents many advantages for the implementation of the WT. In this sections, we will point out this advantages by presenting two architectures for the WT. The first one is a filter banks architecture which implementation isn't improved enough for performing the 2D WT. Readers can find more information on this implementation in [7]. The architecture we propose is based on the lifting scheme and allows some improvements for the 2D WT calculation.

3.1. Filter banks (FB) architecture

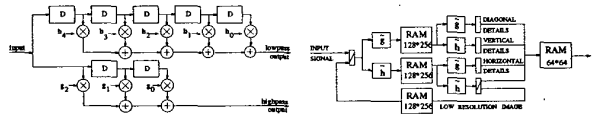

Fig. 3. Implementation of the PA: *a:* basic filtering cells, *b:* whole 2D structure

An implementation of the 2-D wavelet transform using the CDF wavelet is shown in [7]. We see in Figure 3b that the 3 filter banks (Figure 3a) are present, and that an important memory is necessary between the filters to perform the transform. Furthermore, the size of the filters corresponds to those described in section 2.2.1, that is 5 coefficients for the first filter (LP), and 3 coefficients for the second (HP), that implies a total of 6 adders per filter bank.

3.2. Lifting scheme architecture

3.2.1. *1-D structure*

We saw in the section 2.2 that the implementation of the lifting scheme requires few operators. We need, for each step (prediction and update), a total of four adders to perform the complete stage (Figure 4).

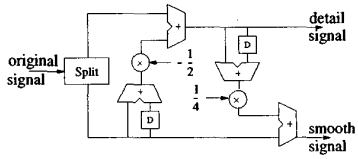

Fig. 4. Structure of the lifting scheme transform block

We can see in Figure 4 that there is no need for memory during the 1-D transform: the transformed coefficients overwrite the original ones during the decomposition. Thus, the transform can be performed in real-time, at the pixel clock. We get the same conclusion for the reconstruction step. Furthermore, unlike the FB method, the subsampling is done before the computations, thus there is no lost of computational time.

3.2.2. *2-D structure*

Unlike the 1-D case, we need, in the 2-D case, some memory to perform the transform. we use a way to perform the 2-D transform that only need very few memory: before starting the transform along the columns, we only wait that a few rows are treated and then start the vertical decomposition on these rows (Figure 5): this method allows the 2-D transform using less than 4 lines of memory. The implementation of this structure is shown in the section 4.

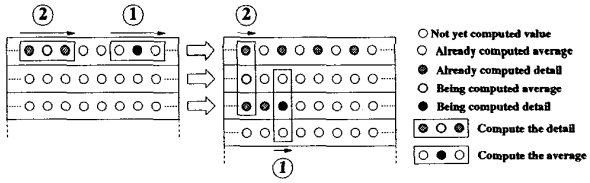

Fig. 5. Vertical decomposition after computation of 3 lines

3.3. Comparisons

3.3.1. *Number of logic blocks*

The lifting is very interesting for integrated systems because of little need for logic blocks. The table 1 shows the differ-

ence between the lifting scheme and the filter banks. This table also shows the number of adders that we need to perform either the transform or the inverse transform, in the 1-D and 2-D cases.

3.3.2. Memory

We saw that we don't need any memory during the transform, because the calculated values replace the original ones. Using the method described in section 4.1.1, we can reduce the memory need for the 2D transform, as shown in table 1.

	Lifting scheme	Filter banks
# adders (1D)	4	6
# adders (2D)	12	18
Mem. need (dec.)	<1 kB	84 kB
Mem. need (rec.)	<1 kB	112 kB

Table 1. Resources needed by lifting vs. pyramid algorithm

4. HARDWARE IMPLEMENTATION

4.1. Overview

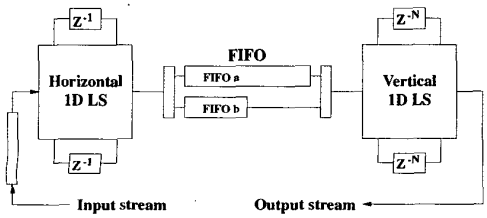

Fig. 6. 2D implementation of the lifting scheme

Figure 6 shows the implementation of the 2-D lifting scheme. The input image is first decomposed horizontally. Then, using some improved techniques, we will decompose it along vertical axis using only 3.5 memory lines. There is no noticeable difficulty in implementing the first 1-D horizontal decomposition block, thus, we'll point out the vertical decomposition.

4.1.1. Vertical decomposition of the sub-images : the line-buffer method

Figure 7 shows the different steps of the 2-D transform using lifting. We consider that the memory a contains the previous already computed detail line. memory b contains the previous approximation. When the 1-D decomposed detail lines is computed, we store it in the memory c. We can then start the vertical decomposition when the approximation coefficient comes in (Fig.7-1). Once the detail is computed, we can re-use it with the previous detail to compute the average (Fig.7-2). Next, we extract the previous detail and average (Fig.7-3) and replace them respectively with the computed detail and average (Fig.7-4). We re-apply this scheme to the next incoming value (Fig.7-5), and once the whole line is treated, we can reload the memories c and d with the incoming lines (Fig.7-6) and repeat the scheme from the beginning.

We can figure out which type of memory we will use. two FIFO of sizes N and $(\frac{N}{2}+1)$, to keep a constant data rate at the output of the two FIFO. For memories a and b we just have to use two delay lines of size N (where N is the size of the line)

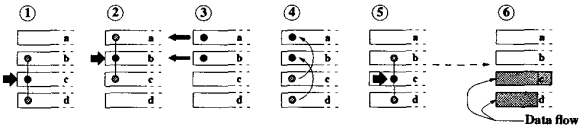

Fig. 7. The line-buffer method

4.2. Implementation results

The table 2 below shows the performances of our system compared to filter banks [7] and ADV 601. Notice that the needed memory in this table is extended to the maximum size of the images that the system can treat.

	Lifting scheme	Filter banks [7]	ADV 601 [8]
Max. pixel rate	66 MHz	1 MHz	14.75 MHz
Mem. need (dec.)	3 kB	84 kB	Unknown
Mem. need (rec.)	3 kB	112 kB	Unknown
Max. image size	1024 × 2048	256×256	768×288
Can process video ?	yes	no	yes

Table 2. Performances of lifting vs. PA and ADV601

5. ARCHITECTURE FOR THE MULTI-RESOLUTION ANALYSIS

5.1. 2D multi-resolution decomposition architectures

There are two easy-to-implement multi-resolution architectures:

- the first one consists in using a picture memory used to store the whole approximation sub-image, as the Figure 3.b shows.
- the second solution is to reproduce as many transform blocks as necessary (*i.e.* one block per level).

The first solution needs too many memory for an on-chip implementation (several kilobytes of memory). The second solution increases the hardware resources needed (the transform blocks and their associated memory). Thus, we have to develop a method to perform the multi-resolution 2D analysis that will re-use the same wavelet transform block, and minimize the necessary memory.

5.2. The proposed architecture

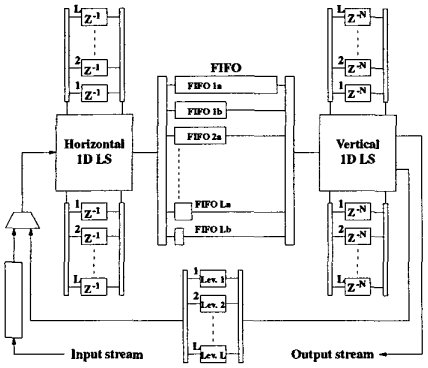

Fig. 8. Architecture for the multi-resolution analysis

The architecture we propose is based on the Recursive Pyramid Algorithm. The RPA consists of re-arranging the order of the N outputs such that an output is scheduled at the earliest possible instance [9, 10]. Figure 9 shows how the recursive pyramid algorithm produces a regular data flow with samples from different resolutions.

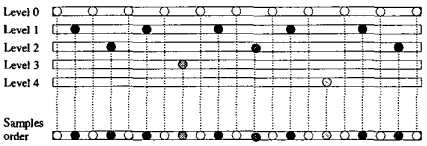

Fig. 9. Recursive pyramid algorithm (RPA)

To compute the multi-level WT, we have to store the low resolutions samples in memory cells until we can compute them, and interlace them with the original samples in the input data flow. Therefore, we use some small FIFO and a multiplexer at the input of the system. A context switching procedure will allow us to store this samples in the appropriate buffers (latches or FIFO). Each time the system will switch its working context, all the corresponding data will be loaded from the memory cells, and the previous ones stored.

Figure 8 shows the modified architecture to perform the WT. The WT begins with the original input data flow. The computed 2D-coefficients are stored in their corresponding input FIFO memories. As soon as two samples from next resolution are ready in this FIFO, we can switch the working context to this level. This will save the current samples in their appropriate memory cells (latches) in the horizontal 1D transform block, and read the data in the latches corresponding to the current level. Thus, all the old data are stored and the new ones are read in one clock cycle. After the computations, the result is sent to the corresponding FIFO. The vertical 1D transform block works in the same manner. The total memory of the modified architecture remains smaller than $7Nb$, where b is the size of the samples in bits, and is equivalent to others architectures [9, 10].

In order to validate this method, we are currently working on the implementation of the architecture for the multi-resolution in a Flex 10k100 FPGA.

6. CONCLUSION

In this paper, we have shown a new way of implementing the wavelet transform. This method combines an efficient processing rate with low area cost and memory use. It can easily be adapted to perform the inverse transform.The described architecture provides a good integration and can be easily prototyped on a FPGA (Field Programmable Gate Array), due to the small memory and the few operators needed. The system and its memory (latches or FIFO) are designed to allow an optimal working frequency and is not hardware dependent. The inherent SIMD parallelism and the pipeline techniques used allow the system to work at very high frequency. We intend to extend the architecture to allow different wavelet to be be implemented, by using programmable coefficients and user defined lifting steps number.

7. REFERENCES

[1] Stéphane G. MALLAT, "A Theory for Multi-Resolution Signal Decomposition: The Wavelet Representation", *IEEE Transactions On Pattern Analysis And Machine Intelligence*, vol. 11, N 7, July 1989

[2] A. COHEN, I. DAUBECHIES and J. FEAUVEAU, "Bi-orthogonal bases of compactly supported wavelets", *Comm. Pure Appl. Math.*, vol. 45, pp. 485-560, 1992

[3] Wim SWELDENS, "The Lifting Scheme: A Custom-design Construction of Biorthogonal Wavelets", *Appl. Comput. Harmon. Anal.*, vol. 3, N 2, pp. 186-200, 1996

[4] Wim SWELDENS, "The Lifting Scheme: A Construction of Second Generation Wavelets", Technical Report 1995:6, Industrial Mathematics Initiative, Department of Mathematics, University of South Carolina, May 1995

[5] Wim SWELDENS and Peter SCHRÖDER, "Building your Own Wavelet at Home", *Wavelets in Computer Graphics*, ACM SIGGRAPH Course notes, PP. 15-87, 1996

[6] Gabriel FERNÁNDEZ, Senthil PERIASWAMY and Wim SWELDENS, "LIFTPACK: A Software Package for Wavelet Transforms using Lifting", *Proc. SPIE* 2825, pp.396-408, 1996

[7] Frédéric TRUCHETET and André FORYS, "Implementation of still-image compression-decompression scheme on fpga circuits", *SPIE*, vol. 2669, pp. 66-75, January 1996

[8] Analog Device, "Ultra Low Cost Video Codec", *Datasheet*, 1997, http://products.analog.com

[9] Chaitali CHAKRABARI, Mohan VISHWANATH, Robert M. OWENS, "Architecture for Wavelet Transforms: A Survey", *Journal of VLSI Signal Processing* 14, pp. 171-192, 1996

[10] Mohan VISHWANATH, Robert Michael OWENS, Mary Jane IRWIN, "VLSI Architectures for the Discrete Wavelet Transform", *IEEE Transactions on Circuits and Systems – II: Analog and Digital Signal Processing*, vol. 42, N 5, May 1995