FPGA design and implementation of a wavelet-domain video denoising system

Mihajlo Katona[1], Aleksandra Pižurica[2],
Nikola Teslić[1], Vladimir Kovačević[1], and Wilfried Philips[2]

[1] University of Novi Sad, Chair for Computer Engineering,
Fruškogorska 11, 21000 Novi Sad, Serbia and Montenegro
mihajlo.katona@krt.neobee.net
[2] Ghent University, Dept. Telecommunications and Information Processing,
Sint-Pietersnieuwstraat 41, B-9000 Ghent, Belgium
Aleksandra.Pizurica@telin.UGent.be

Abstract. Multiresolution video denoising is becoming an increasingly popular research topic over recent years. Although several wavelet based algorithms reportedly outperform classical single-resolution approaches, their concepts are often considered as prohibitive for real-time processing. Little research has been done so far towards hardware customization of wavelet domain video denoising. A number of recent works have addressed the implementation of critically sampled orthogonal wavelet transforms and the related image compression schemes in Field Programmable Gate Arrays (FPGA). However, the existing literature on FPGA implementations of overcomplete (non-decimated) wavelet transforms and on manipulations of the wavelet coefficients that are more complex than thresholding is very limited.
In this paper we develop FPGA implementation of an advanced wavelet domain noise filtering algorithm, which uses a non-decimated wavelet transform and spatially adaptive Bayesian wavelet shrinkage. The standard composite television video stream is digitalized and used as source for real-time video sequences. The results demonstrate the effectiveness of the developed scheme for real time video processing.

1 INTRODUCTION

Recently, several promising multiresolution (wavelet domain) video noise filters have been proposed. These can be categorized in *non-separable* spatio-temporal approaches utilizing a three-dimensional (3-D) wavelet representation [1], [2] and *separable* approaches that combine 1-D temporal filtering and 2-D spatial denoising in the wavelet domain [3, 4]. Although these wavelet domain video filters were reported to outperform the more classical, single-resolution techniques, little research has been done so far towards their customization for hardware implementations and consequently, they are often considered as prohibitive for real-time applications.

Modern hardware solutions for digital signal processing algorithms increasingly employ Field Programmable Gate Arrays (FPGA). FPGAs accelerate the

execution of algorithms and offer a tremendous potential to improve performance through parallelization. While FPGA design of the orthogonal wavelet transform and related image compression tools (JPEG2000) has been well studied [5–7], only a few publications address FPGA design of other types of wavelet transforms or wavelet coefficient manipulations other than simple thresholding.

In this paper we efficiently customize one of the latest wavelet domain denoising filters [8] and implement it in FPGA's for real-time video denoising. Some additional details of the developed architecture can be found in [9], where we described the preliminary results of this research.

This paper is organized as follows. In Section 2 we describe the implemented algorithm and present its customization for real-time implementation. The real-time environment that is used in this study is described in Section 3. The conclusions are in Section 4.

2 DEVELOPED FPGA DESIGN

Fig. 1 depicts the implemented video denoising scheme, which consists of the non-decimated 2-D wavelet transform, Bayesian wavelet shrinkage followed by the inverse wavelet transform and selective recursive temporal filtering.

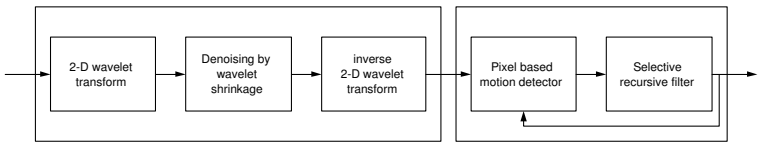

Fig. 1. The implemented denoising scheme.

An important issue is whether to implement the floating-point arithmetic in FPGA and to use the original algorithm arithmetic or to convert the algorithm to the integer/fixed-point arithmetic. We use the fixed-point arithmetic which is less complex for a hardware implementation.

2.1 Non-decimated wavelet transform in FPGA's

While the implementations of the orthogonal wavelet transform have been extensively studied in literature [5–7] much less research has been done towards hardware implementations of the non-decimated wavelet transform. We design an FPGA implementation of the non-decimated wavelet transform using the algorithm *à trous* as it is described by Mallat and Zhong [10]. This algorithm replaces sub-sampling of the filtered signal by up-sampling the filters, where 2^{j-1} zeros ("holes", i.e., *trous* in French) are inserted between the filter coefficients at the decomposition level j.

We use the SystemC library [11] and a previously developed simulation environment [12, 13] to develop a real-time model of the wavelet decomposition and composition [9]. The input value is 8 bit integer. We use the 16 bit arithmetic for wavelet decomposition and composition. The input 8 bits are placed at bit positions from 14 to 7. The output bits occupy the same positions (see Fig. 2).

F	E	D	C	B	A	9	8	7	F	5	4	3	2	1	0
0	INPUT								0	0	0	0	0	0	0
0	OUTPUT								X	X	X	X	X	X	X

Fig. 2. Input/Output data format.

Our extensive simulations and tests demonstrate that this implementation results in a perfect reconstruction and gives practically the same results as a referent MATLAB code of the algorithm *à trous* [10]. At a number of input frames there were more than 97.13% errorless pixels with mean error of 0.0287. Analysis of those images at the level of bit representation, reveals that maximally 1 bit out of 16 was wrong. Moreover, the wrong bit may occur only on the least significant bit (LSB) position. If we take into account that input and output pixels are 8 bit places above first 6 LSB bits, we can ignore this error. This is depicted in Fig. 2.

2.2 FPGA design of a spatially adaptive wavelet shrinker

We design FPGA architecture for a spatially adaptive wavelet denoising method of [8], which shrinks each wavelet coefficient according to the probability of presenting a "signal of interest" given the observed coefficient value and given a local spatial activity indicator (LSAI). In our implementation LSAI is the locally averaged coefficient magnitude within a 3x3 window and the signal of interest is defined as the noise-free component that exceeds in magnitude the noise standard deviation σ.

The analyzed denoising algorithm can be summarized as follows. Let y_l denote the noise-free wavelet coefficient and w_l its observed noisy version at the spatial position l in a given wavelet subband. For compactness, we suppress here the indices that denote the scale and the orientation. The locally averaged coefficient magnitude is denoted by $z_l = \sum_{k \in N_l} |w_k|$, where N_l is a square window centered at the position l. Further on, let H_1 denote the hypothesis "*the signal of interest is present*: $|y_l| > \sigma$ " and let H_0 denote the opposite hypothesis "*the signal of interest is absent*: $|y_l| \leq \sigma$". The shrinkage estimator from [4] is

$$\hat{y}_l = \frac{\rho \xi_l \eta_l}{1 + \rho \xi_l \eta_l} w_l, \tag{1}$$

where

$$\rho = \frac{P(H_1)}{P(H_0)}, \quad \xi_l = \frac{p(w_l|H_1)}{p(w_l|H_0)} \quad \text{and} \quad \eta_l = \frac{p(z_l|H_1)}{p(z_l|H_0)}. \tag{2}$$

and where $p(w_l|H_0)$ and $p(w_l|H_1)$ denote the conditional probability density functions of the noisy coefficients given the absence and given the presence of a signal of interest. Similarly, $p(z_l|H_0)$ and $p(z_l|H_1)$ denote the corresponding conditional probability density functions of the local spatial activity indicator. Under the Laplacian prior for noise-free data $p(y) = (\lambda/2)\exp(-\lambda|y|)$ we have [8] $\rho = \exp(-\lambda T)/(1 - \exp(-\lambda T))$. The analytical expressions for ξ_l and η_l seem too complex for the FPGA implementation. Based on an extensive experimental study, as we explain later in this Section, we efficiently implement the two likelihood ratios ξ_l and η_l as appropriate *look-up tables*, stored in two "Read-Only" Memories (ROM).

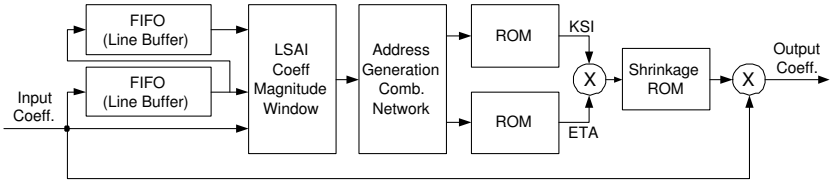

Fig. 3. Block schematic of implemented denoising architecture.

The developed architecture is presented in Fig. 3. One ROM memory, containing the look-up table ξ_l, is addressed by the coefficient magnitude $|w_l|$, and the other ROM memory, containing the look-up table $\rho\eta_l$ is addressed by LSAI z_l. For calculating LSAI, the coefficient values from the current line and from the previous two lines are averaged within a 3x3 window. The read values from ROM's are multiplied and the product r is used to address another look-up table $r/(1+r)$, denoted as "shrinkage ROM". Its output (the shrinkage factor) is multiplied with the input coefficient to produce the denoised coefficient value.

The generation of the appropriate look-up tables for the two likelihood ratios resulted from our extensive experiments on different test images[3] and different noise-levels. Fig. 4 illustrates the likelihood ratio ξ_l calculated from one test image at different noise levels. These diagrams show another interpretation of the well known threshold selection principle in wavelet denoising: a well chosen threshold value for the wavelet coefficients increases with the increase of the noise level. The maximum likelihood estimate of the threshold T (i.e., the value for which $p(T|H_0) = p(T|H_1)$) is the abscissa of the point $\xi_l = 1$. Fig. 5 displays the likelihood ratio ξ_l, in the diagonal subband HH at third decomposition level,

[3] We used standard test images such as "Lena" and "Barbara", and frames from different standard test video sequences, such as "flower garden", "Miss America", "salesman", etc.

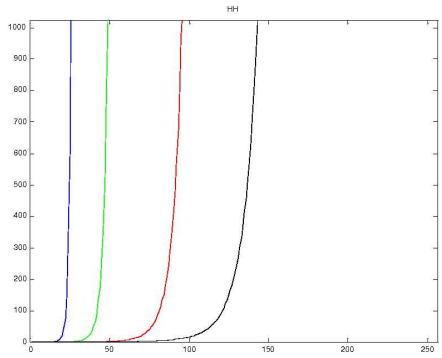

Fig. 4. Likelihood ratio ξ_l for one test frame and 4 different noise levels (σ=5,10,20,30).

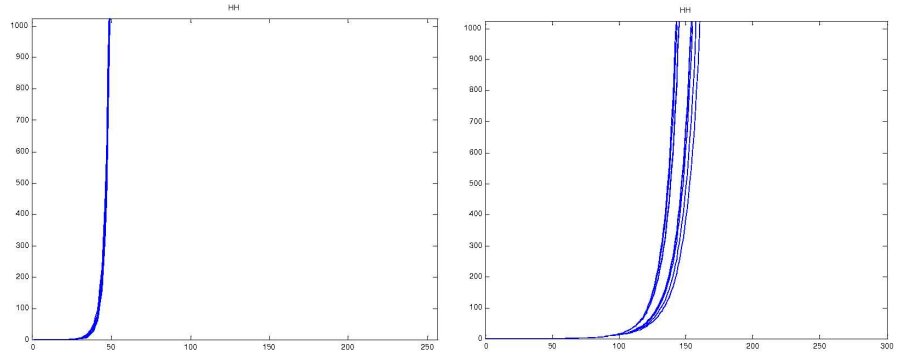

Fig. 5. Likelihood ratio ξ_l displayed for 10 frames with fixed noise levels: $\sigma = 10$ (left) and $\sigma = 30$ (right).

for 10 different frames with fixed noise standard deviations ($\sigma = 10$ and $\sigma = 30$). From a practical point of view, the difference between the calculated likelihood ratios for different frames is minor, especially for lower noise levels (up to $\sigma = 20$). Therefore we average the likelihood ratios over different frames and store these values as the corresponding look-up tables for several different noise levels ($\sigma = 5, 10, 15$ and 20). In the denoising procedure, the user selects the input noise level, which enables addressing the correct set of the look-up tables. The performance loss of the algorithm due to simplifications with the generated look-up tables for different input noise levels is shown in Fig. 6. These results represent peak signal to noise ratio (PSNR) values averaged over frames of several different video sequences. For σ=10 the average performance loss was only 0.13dB (and visually, the differences are difficult to notice) while for σ=20 the performance loss is 0.55dB and is on most frames becoming visually noticeable, but not highly disturbing. For higher noise levels, the performance loss increases.

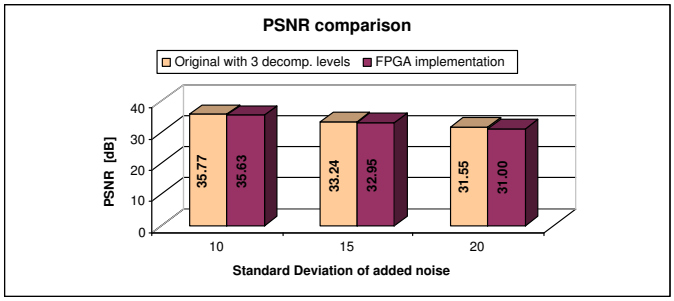

Fig. 6. Performance of the designed FPGA implementation in comparison with the original software version of the algorithm, which employs exact analytical calculation of the involved shrinkage expression.

2.3 FPGA design of a selective recursive temporal filter

A pixel based motion detector with selective recursive temporal filtering is quite simple for hardware implementation. Since we first apply a high quality spatial filtering the noise is already significantly suppressed and thus a pixel based motion detection is efficient. In case the motion is detected the recursive filtering is switched off.

Two pixels are needed for temporal filtering: one from the current field and another from the same spatial position in the previous field. We store the two fields in the output buffer and read the both required pixel values in the same cycle. If the absolute difference between these two pixel values is smaller than the predefined threshold value, *no motion* case is assumed and the two pixel values are subject to a weighted averaging, with the weighting factors defined in [4]. In the other case, when motion is detected, the current pixel is passed to the output. The block schematic in Fig. 7 depicts the developed FPGA ar-

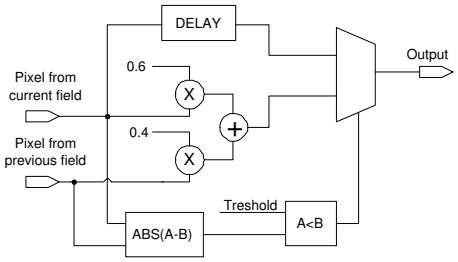

Fig. 7. Block schematic of implemented temporal filter.

chitecture of the above described selective recursive temporal filter. In terms of

computation accuracy, the only required adaptation of the original filter is the conversion from floating-point arithmetic to the integer arithmetic. We use the 8 bit arithmetic because the filter is located in the time domain where all the pixels are represented as 8 bit integers.

3 EVALUATION IN A REAL-TIME ENVIRONMENT

In our implementation we use the standard television broadcasting signal as a video signal source. A common feature of all standard TV broadcasting technologies is that the video sequence is transmitted in the analog domain (this excludes the latest DVB and HDTV transmission standards). Thus, before digital processing of television signals the digitalization is needed. Also, after digital processing the sequence has to be converted back to the analogue domain in order to be shown on a standard tube display. This pair of A/D and D/A converters is well known as a codec. The 8 bit codec, with 256 levels of quantization per pixel, is considered sufficient for preserving all details in the sequence. We use the PAL-B broadcasting standard and 8 bit YUV 4:2:2 codec. The hardware platform set-up consists of three separate boards: analog front-end (A/D conversion), processing board and analog back-end (D/A conversion) Each board corresponds to one of the blocks presented in Fig. 8:

The processing board consists of two Xilinx Virtex II FPGAs (XC2V6000-5) [14] and is equipped with plenty of SDRAM memory (6 banks with 32 bit access made with 256Mbit ICs). Additional implementation details are in [9].

An important practical issue is the specification of the following two parameters: estimated noise standard deviation σ and the motion detection threshold. Currently we keep the motion detection threshold fixed and allow the choice of σ from a set of predefined values. A future work will concentrate on estimating these parameters adaptively from the video sequence and on measuring the sensitivity of the scheme to these parameters.

An alternative real-time implementation of this algorithm may be based on commercially available DSP processors instead of FPGA. Indeed the approximation of the algorithm based on ROM tables as we proposed and speed-optimized programming in languages like C or C++ should significantly accelerate the software version of the algorithm. In this case, the profiling of the software implementation would be required to determine the DSP parameters, like the needed MIPS performance (MIPS - *Million Instructions Per Second*) and the ROM size, which are needed for real-time program running. However, it is not

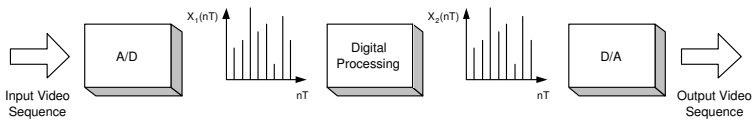

Fig. 8. A digital processing system for television broadcasting video sequences.

certain that a general purpose DSP processor could perform the non decimated wavelet transform of a television stream in real time due to a number of needed memory accesses for reading and writing the wavelet coefficients parallel with the accesses to the input and output buffers.

4 CONCLUSION

New trends in video technology and emerging wavelet domain video denoising methods require development of the appropriate real-time hardware architectures with FPGA's. This paper revealed technical details of one of such developments which has resulted in a real-time implementation of one of the latest wavelet domain denoising methods. We believe that some architectural design aspects presented in this paper should be interesting for future FPGA design of other, related wavelet domain denoising methods.

References

1. Roosmalen, P., Westen, S., Lagendijk, R., Biemond, J.: Noise reduction for image sequences using an oriented pyramid thresholding technique. In: IEEE Conf. on Image Process. Lausanne, Switzerland (Sep. 1996) 375378
2. Selesnick, I., Li, K.: Video denoising using 2D and 3D dual-tree complex wavelet transforms. In: Wavelet Applications in Signal and Image Processing. Volume 5207 of SPIE Conf. (Aug. 2003) San Diego
3. Zlokolica, V., Pižurica, A., Philips, W.: Video denoising using multiple class averaging with multiresolution. International Workshop VLBV03 ((Madrid, Spain, Sep. 2003))
4. Pižurica, A., Zlokolica, V., Philips, W.: Noise reduction in video sequences using wavelet-domain and temporal filtering. In: Wavelet Applications in Industrial Processing. Proc. SPIE (2003)
5. Nibouche, M., Bouridane, A., Murtagh, F., Nibouche, O.: FPGA-based discrete wavelet transforms system. In Brebner, G., Woods, R., eds.: Field-Programmable Logic and Applications, Springer-Verlag (2001) 607–612
6. Wu, B.F., Hu, Y.Q.: An efficient VLSI implementation of the discrete wavelet transform usng embedded instruction codes for symetric filters. IEEE Transaction on Circuits and Systems for Video Technology **13 no. 9** (September 2003)
7. Dillen, G., Georis, B., Legat, J.D., Cantineau, O.: Combined line-based architecture for the 5-3 and 9-7 wavelet transform of JPEG2000. IEEE Transaction on Circuits and Systems for Video Technology **13 no. 9** (September 2003)
8. Pižurica, A., Philips, W.: Estimating the probability of the presence of a signal of interest in multiresolution single- and multiband image denoising. (IEEE Trans. Image Processing (in press))
9. Katona, M., Pižurica, A., Teslić, V.Z.N., Philips, W.: Real-time wavelet domain video denoising implemented in FPGA. In: Wavelet Applications in Industrial Processing II. Volume 5607 of Proc. SPIE. (2004) 63–69
10. Mallat, S., Zhong, S.: Characterization of signals from multiscale edges. IEEE Trans. Pattern Anal. and Machine Intel. **14** (1992) 710–732
11. : SystemC Version 2.0 Users Guide. SystemC Inc., www.systemc.org (2002)

12. Katona, M., Teslic, N., Kovacevic, V., Temerinac, M.: Test environment for bluetooth baseband inegrated circuit development. In Milovanovic, B.D., ed.: TELSIKS 2001. Volume 2 of Telecommunication Technologies. (Septmeber 2001) 405–408
13. Katona, M., Teslic, N., Krajacevic, Z.: FPGA design with SystemC. In Napieralski, A., ed.: Mixed Design og Integrated Circuits and Systems. Volume 1 of MIXDES 2003. (Jun 2003) 220–223
14. : Virtex II Platform FPGA: Complete Data Sheet. www.xilinx.com. (2004)