Design and implementation of a high-quality, low-power deinterlacer circuit

Livio Tenze, Stefano Marsi, Sergio Carrato
D.E.E.I., University of Trieste, v. Valerio 10, 34100 Trieste, Italy
e-mail `tenze,marsi,carrato@ipl.univ.trieste.it`

ABSTRACT — **A novel circuit for a simple non motion-compensated deinterlacer is presented, where a motion detection module effectively computes a weighted mean of a temporal and a spatial interpolator. Both the motion detection module and the two interpolators are innovative, simple and effective, so that they are able to provide very good results, as experimental results show, at a low computational complexity. The final circuit, based on a pipeline architecture, has been designed to work on a real time video digital signal with a low power consumption.**

1 Introduction

A common situation which is often encountered in present days is the need to display interlaced video sequences on progressive displays, mainly because of the widespread use of PCs for multimedia applications and of the increasing diffusion of solid state displays for TV sets. In this paper, we consider the implementation of a non motion-compensated deinterlacer [1] where two innovative, simple and effective interpolators are used, and their contributions are weighted by a suitable motion detection module. Very good operation is provided also in "difficult" cases such as static horizontal edges and thin lines, and no image flow artifacts are introduced.

After a brief review of the algorithm, a VLSI implementation is proposed, where the adopted architecture will be described along with the technical features of the realized circuit.

2 The proposed algorithm

As already mentioned, our deinterlacing algorithm is aimed at providing good quality and artifact free images at a low computational complexity and, consequently, with low power consumption. It consists of a motion detector module and of two operators for the reconstruction, one being purely temporal and the other purely spatial. The detector modulates the contribution of the two operators, according to the level of estimated motion. Both the detector module and the two operators, while presenting low computational complexity, have very accurate behaviour, i.e. their output is correct also in non trivial cases such as image flow or thin horizontal, or quasi horizontal, lines.

Let us consider a 5×5 mask around the pixel to be interpolated, on the previous, the present, and the following field. Going into detail, with reference to Fig. 1, we consider 5 pixels on the line above and below the pixel to be interpolated, x_7, in the current field, and one pixel above and one below x_7 in the previous and in the following fields (pixels x_1^p, x_{13}^p and x_1^f, x_{13}^f, respectively).

As already mentioned, the approach consists in finding a good spatial estimate, $y_{spatial}$, and a good temporal one, y_{temp}, and then suitable combining them in the final output y_7 of the deinterlacer.

		x_1		
x_2	x_3	x_4	x_5	x_6
		x_7		
x_8	x_9	x_{10}	x_{11}	x_{12}
		x_{13}		

Figure 1: Pixels considered by the deinterlacer. x_7 is the pixel to be reconstructed; pixels x_2 to x_6 and x_8 to x_{12} belong to the current field, while pixels x_1, x_7, and x_{13} are available in the previous and in the following field, and will be referred to as x_1^p, x_7^p, and x_{13}^p, and x_1^f, x_7^f, and x_{13}^f, respectively.

Let us first consider the spatial interpolator. If an edge is present, it is advisable to interpolate along its direction in order to avoid blurring. The edge can be detected by considering differences along some directions (namely, 45^0, 90^0, and 135^0) and giving more weight to the most similar pixels, i.e. the most "correlated" ones. Consequently, we define the following directional means

$$\begin{aligned} m_{45} &= (x_5 + x_9)/2 \\ m_{90} &= (x_4 + x_{10})/2 \\ m_{135} &= (x_3 + x_{11})/2 \end{aligned}$$

and differences

$$\begin{aligned} d_{45} &= (|x_4 - x_8| + 2|x_5 - x_9| + |x_6 - x_{10}|)/4 \\ d_{90} &= (|x_3 - x_9| + 2|x_4 - x_{10}| + |x_5 - x_{11}|)/4 \\ d_{135} &= (|x_2 - x_{10}| + 2|x_3 - x_{11}| + |x_4 - x_{12}|)/4 \\ d_{max} &= \max(d_{45}, d_{90}, d_{135}) \\ d_{min} &= \min(d_{45}, d_{90}, d_{135}) \end{aligned}$$

A direction θ ($\theta = 45, 90, 135$) with good correlation will have a small d_θ and, consequently, a large $(d_{max} - d_\theta)$ and a small $(d_\theta - d_{min})$. We then compute the weights as

$$w_{s,\theta} = \frac{d_{max} - d_\theta + 1}{d_\theta - d_{min} + 1} \quad \text{for} \quad \theta = 45, 90, 135$$

where the terms $+1$ have been added in order to avoid null numerators and denominators, and finally

$$y_{7,spatial} = \frac{w_{s,45} * m_{45} + w_{s,90} * m_{90} + w_{s,135} * m_{135}}{w_{s,45} + w_{s,90} + w_{s,135}} \tag{1}$$

Concerning the temporal part, a weighted mean is computed between x_7 in the previous and in the following field, i.e. x_7^p and x_7^f. In order to do so, some new differences are computed, this time between x_7^p or x_7^f and the 6 closest pixels in the current field, i.e. $x_i,\ i \in I = \{3,4,5,9,10,11\}$. In this case we are not looking for edge directions; in turn, the task is to understand if one of the two pixels x_7^p and x_7^f may be considered a reliable estimate for x_7. This is likely to be correct if the latter ones are similar to the $x_i,\ i \in I$. By defining suitable temporal differences [1] we are able to compute two weights w_t^p and w_t^f, similarly to what has been done for the spatial interpolation. Therefore $x_{7,temp}$ can be estimated according to the following expression:

$$y_{7,temp} = \frac{w_t^p x_7^p + w_t^f x_7^f}{w_t^p + w_t^f}. \qquad (2)$$

Finally, we combine the spatial and the temporal parts with

$$y_7 = (1 - \overline{w_{temp}}) \cdot y_{7,spatial} + \overline{w_{temp}} \cdot y_{7,temp} \qquad (3)$$

with a suitable weight $\overline{w_{temp}}$. In simple cases, a weight of the type

$$\overline{w_{temp}} = (1 - \frac{|x_7^f - x_7^p|}{d_{min} + 1})$$

(with again the term +1 added to avoid null denominators) should be sufficient. Indeed, in the fraction two differences, a temporal and a spatial one, appear in the numerator and in the denominator, respectively; in case of good temporal correlation, in fact, $|x_7^f - x_7^p|$ would be smaller than d_{min}, so that $\overline{w_{temp}} \simeq 1$; *viceversa* if temporal correlation is poor.

A more sophisticated scheme has to be used, however, if thin lines have to be correctly reconstructed and image flow has to be avoided. To this purpose, we also consider several differences between pixels in the present field close to x_7, i.e. x_4 and x_{10}, and pixels over and below x_7^p and x_7^f, i.e. x_1^p, x_{13}^p, x_1^f, and x_{13}^f. In [1] we show that a suitable parameter $\bar{\delta}$ can be introduced, defined upon these differences, which is small if the temporal estimate $y_{7,temp}$ is reliable; consequently we compute the final weight for Eq. 3 according to

$$\overline{w_{temp}} = \max\left(1 - \frac{|x_7^f - x_7^p| + \bar{\delta}}{2 * d_{min} + 1}, 0\right)$$

If the background is uniform and similar to x_7^p or x_7^f, both $|x_7^f - x_7^p|$ and $\bar{\delta}$ are small, so that $\overline{w_{temp}} \simeq 1$ and temporal interpolation is used. It has to be noted that this situation also occurs in case of static thin lines, which consequently are correctly reconstructed and appear sharp in the progressive image; an example is provided in the following section. In case of image flow, in turn, $\bar{\delta}$ is large, so that $\overline{w_{temp}} \simeq 0$ and $y_7 \simeq y_{7,spatial}$.

3 Implementation

Even if the proposed algorithm could be considered quite simple from an analytic point of view, it involves a large amount of similar operations. It is apparent that an implementation of the various functions, such as those described in the algorithm equations, will drive to manage a lot of complex operators (e.g., adders, multipliers, minimum and maximum functions, dividers and so on). Therefore, to produce an effective realization of the algorithm on a small, low power, circuit a possible solution can be obtained sharing the various elementary operators and scheduling them in a pipeline sequential architecture.

Considering the algorithm as a loop that repeatedly executes the operations in its body until an exit condition becomes true, a significant improvement can be obtained scheduling consecutive loop iterations to be partially overlapped in time, i.e. a new loop iteration is started before the current iteration has finished. In such a way the rate of the input/output data becomes independent of the time requested to complete all the operations described in the algorithm and also the throughput of the design can be significantly improved.

The two fundamental pipeline parameters, which greately affect the throughput, are the "initiation interval", i.e. the number of clock cycles between the start of two consecutive loop iterations, and the "latency", i.e. the number of clock cycles required to execute all the operations in a single loop iteration. While the latency simply corresponds to the input-to-output delay, the initiation interval is tightly related to the throughput rate; in fact, the higher the number of clock cycle in the initiation interval, the larger the re-use of the operator in the algorithm. For example, if the initiation interval is composed by 8 clock cycles, an operator that requires less than two clock cycles to complete the operation can be used up to four times by different parts of the algorithm along the entire latency; similarly a faster operator which requires less than one clock cycle can be re-employed up to eight times.

In the implementation of the proposed algorithm we focused our attention on a particular application, i.e. the real time deinterlacing of digital video samples composed by 720×576 pixels at 50 field (25 frames) per second (i.e the input samples presents a 96.4 ns cadence). The developed architecture exploits a pipeline with 64 latency cycles and an initiation period of 8 cycles, while the clock cycle has been kept under 12 ns. Thus the entire initial period of the pipeline composed by 8×12 ns can be completed before a new input sample appears.

A complete scheme of the pipeline is represented in Fig. 2, where however, for the sake of clarity, the transitions have not been shown; in this figure, every operator is represented by a column, while the time cadence has been reported along the vertical axis, and the gray cells represent the period during which every operator is used in the pipeline cycle.

In this scheme we can note that all the operators are characterized by a delay smaller than the input sample rate, and that most of them are re-employed several times during different clock cycles to perform similar tasks, obtaining in such a way a high throughput degree architecture. Moreover, considering that any point of intersection between a column and a row in the scheme represents a time-slot where every operator can be scheduled, it can be noted that the entire scheduling has been designed under the following constrain: by shifting vertically the whole scheme of pipeline by a factor which is multiple of the initiation cycles, every operator has to be put on a free time-slot.

It has to be noted that this constrain is not fundamental for the inputs, since the input samples can be reused several times while the pipeline loop goes. Moreover, the overlap of inputs among different pipeline loops makes it possible to limit the number of input pins.

For the sake of clarity, referring to figure 1 and con-

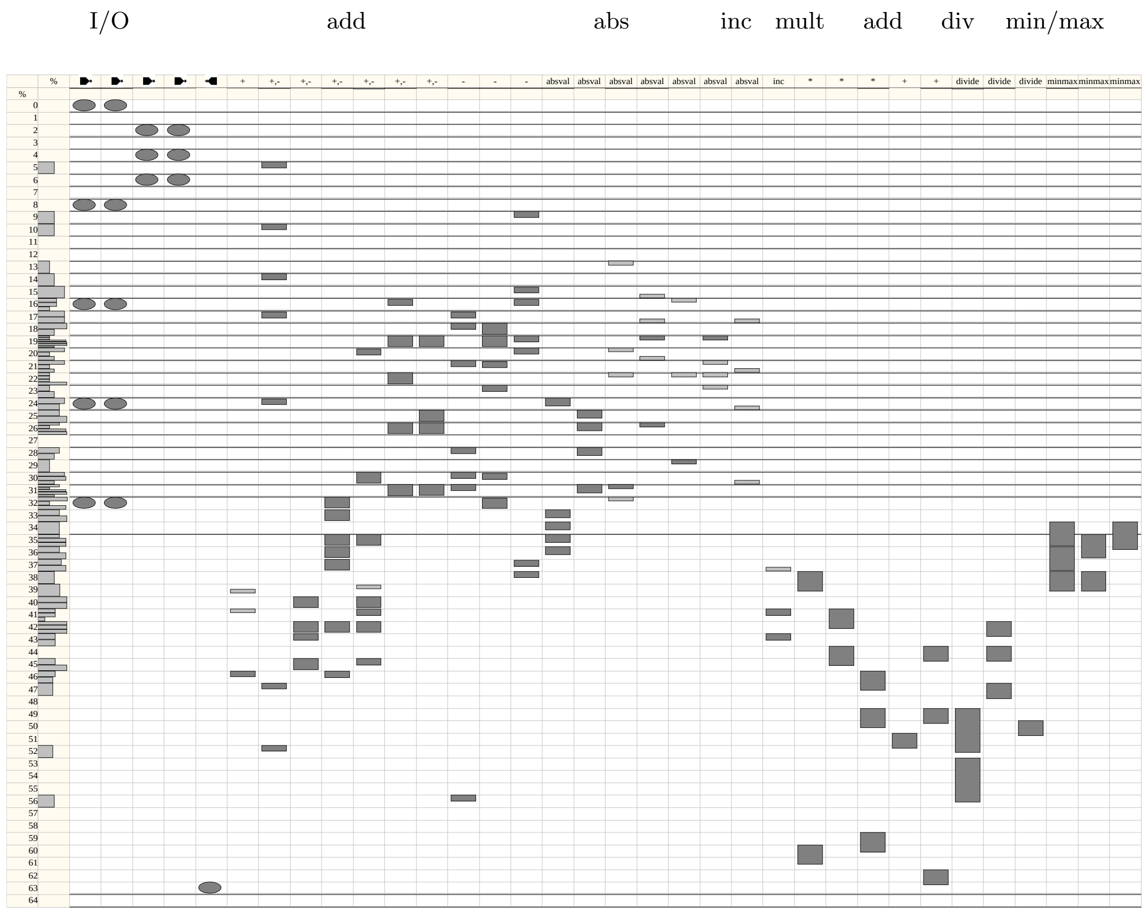


Figure 2: Pipeline scheme for the circuit architecture

sidering that several pixels are involved in evaluating the output, it is possible to constrain a single input pixel to be interpreted as x_6 in the actual pipeline cycle, as x_5 in the previous one, and so on. In such a way it is sufficient to use, for the whole circuit, just 4 different input data buses. Two of them are devoted to the inputs of the horizontal pixels, belonging to the two picture rows involved in the current field ($x_2 \dots x_6$ and $x_8 \dots x_{12}$) where their rate is equal to that of the input data. Indeed, the other two input buses are used for the acquisition of the vertical pixels of the two columns (x_1, x_7, x_{13}) both in the previous and next field. Since all these data have to be acquired before a new pipeline cycle starts, a higher rate has to be considered to feed the circuit with these data.

In the physical implementation of the circuit all of the elementary operators (e.g., adders, multipliers, dividers) have been developed in a combinatorial form, using fixed point arithmetic with a suitable number of bits. In order to implement these operators we exploited the features of architectures such as carry look ahead and Wallace tree.

The final circuit has been realized with a 0.35 μm standard cell CMOS technology; a simplified layout is reported in Fig. 5. The core occupies a total area of 1.4 mm^2 and is composed by about 6000 standard cells, while the simulated power consumption is approximately 40 mW.

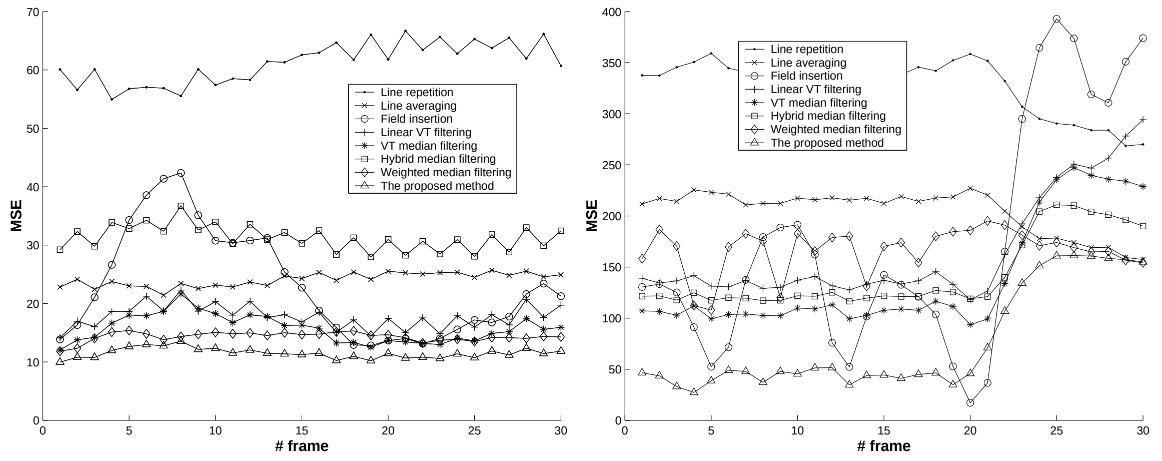
4 Experimental results

The presented algorithm has been tested with several real world sequences; as an example, we show the results obtained with “Salesman” and “Tennis”. These original sequences are progressive; in order to numerically evaluate the behaviour of the deinterlacing algorithms, the mentioned movies have been artificially interlaced. In this way it is possible to estimate, using the MSE, the difference between the original progressive frames and the reconstructed ones.

Each of these sequences is characterized by different features: in “Salesman” slow motion and some thin lines are present, while in “Tennis” a large amount of fast motion and thin features show up. Following [2], our method is compared with several well-known non motion compensated methods of comparable complexity, namely line repetition, line averaging, field insertion, linear VT filtering , VT median filtering, hybrid median filtering and weighted median filtering.

In Fig. 3a we compare the results obtained applying the proposed method to “Salesman”. It may be seen that the lowest value of Mean Square Error (MSE) is achieved by the proposed deinterlacer. Moreover, direct inspection of the sequences shows that our approach is able to preserve the thin lines which are present, without introducing flickering, as most of the other methods do.

In Fig. 3b and 4 we propose the results obtained with “Tennis”. Also in this case, our approach produces the



a) b)

Figure 3: MSE for "Salesman" (a) and "Tennis" (b) for several well-known non motion-compensated deinterlacers.

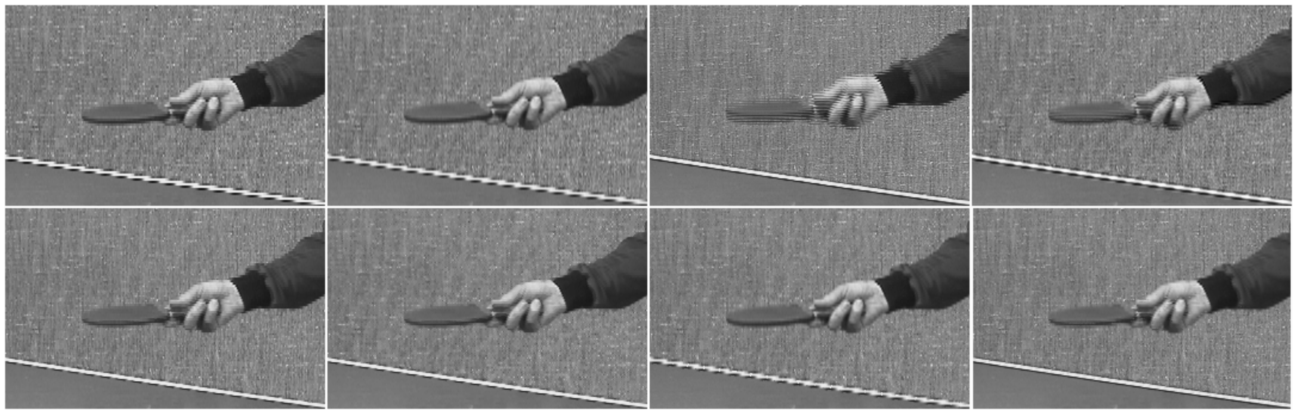


Figure 4: Image number 3 of the "Tennis" sequence. From top-left to bottom-right: line repetition, line averaging, field insertion, linear VT filtering, VT median, hybrid median, weighted median, proposed deinterlacer.

best results: the moving hand of the table tennis player is well reconstructed and, most notably, the white diagonal line (the border of table) is in our case almost perfectly interpolated, while line repetition, line averaging, linear VT filtering and weighted median produce visible artifacts. The field insertion technique correctly reconstructs the diagonal line, due to the lack of motion in that region, but introduces large artifacts while interpolating the moving hand. Median-based methods perform somehow better than the linear techniques but the diagonal line is smoothed or jagged in some parts; moreover, they are not able to maintain the background details when there is no motion.

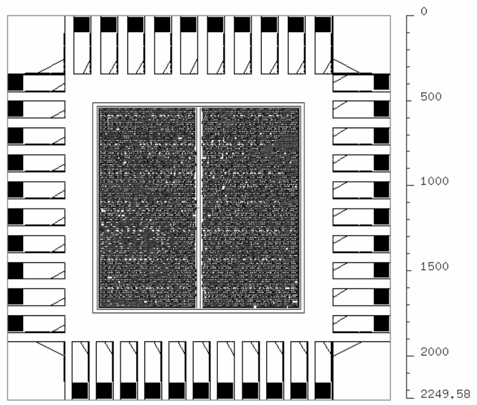


Figure 5: Simplified layout of the final chip. Core size is about 1.4 mm^2.

References

[1] L. Tenze, A. Fermo, and S. Carrato, "A high-quality edge and motion sensitive deinterlacer," in *Proc. 1st COST 276 Workshop on Information and Knowledge Management for Integrated Media Communication*, (Leganes/Madrid, Spain), Nov. 2001. In print.

[2] G. De Haan and E. B. Bellers, "Deinterlacing—An overview," *Proceedings of the IEEE*, vol. 86, pp. 1839–1857, Sept. 1998.