Fuzzy motion adaptive algorithm and its hardware implementation for video de-interlacing

J. Gutiérrez-Ríos[1], P. Brox[2,3], F. Fernández-Hernández[1], I. Baturone[2,3] and S. Sánchez-Solano[2]

[1] *Dept. Tecnología Fotónica. Universidad Politécnica Madrid. Campus de Montegancedo. 28660 Boadilla del Monte. Madrid (Spain) {jgr@dtf.fi.upm.es}*

[2] *Instituto de Microelectrónica de Sevilla. Centro Nacional de Microelectrónica (CSIC). 41092 Américo Vespucio s/n. Sevilla (Spain) {brox@imse-cnm.csic.es}*

[3] *Dept. Electrónica y Electromagnetismo. Universidad de Sevilla (Spain)*

Abstract

Interlacing techniques were introduced in the early analog TV transmission systems as an efficient mechanism capable of halving the video bandwidth. Currently, interlacing is also used by some modern digital TV transmission systems, however, there is a problem at the receiver side since the majority of modern display devices require a progressive scanning. De-interlacing algorithms convert an interlaced video signal into a progressive one by performing interpolation. To achieve good de-interlacing results, dynamical and local image features should be considered. The gradual adaptation of the de-interlacing technique as a function of the level of motion detected in each pixel is a powerful method that can be carried out by means of fuzzy inference. The starting point of our study is an algorithm that uses a fuzzy inference system to evaluate motion locally (FMA algorithm). Our approach is based on convolution techniques to process a fuzzy rulebase for motion-adaptive de-interlacing. Different strategies based on bi-dimensional convolution techniques are proposed. In particular, the algorithm called 'single convolution algorithm' introduces significant advantages: a more accurate measurement of the level of motion by using a matrix of weights, and a unique fuzzification process after the global estimation, which reduces the computational cost. Different architectures for the hardware implementation of this algorithm are described in VHDL language. The physical realization is carried out on a RC100 Celoxica FPGA development board.

Key words: De-interlacing, Fuzzy Logic, Motion Adaptive, Convolution.

[1] This work was partially supported by MOBY-DIC project FP7-INFSO-ICT-

1 Introduction

Interlaced video is a method to halve video bandwidth by eliminating horizontal lines in successive frames. An interlaced video sequence consists of a set of alternating even and odd fields containing, respectively, the even- and the odd-numbered lines of the original images. Even though this transmission scheme was introduced by the first analogue TV broadcasting systems, it is also used by some of modern digital video standards. However, modern TV sets, computer displays, and LCD displays use progressive scanning, that is, they require complete frames containing all the lines.

De-interlacing algorithms are required to convert interlaced video signals into a progressive format by interpolating the missing lines of each field (see Fig. 1). To perform de-interlacing, many algorithms have been reported in the literature [1]. They can roughly be classified into two categories: non-motion compensated (non-MC) and motion-compensated (MC) algorithms [1]. MC techniques involve a huge computational cost but they offer the most effective results in moving areas [2]. Among non-MC techniques, two categories are distinguished: spatial interpolation or intra-field techniques, and temporal interpolation or inter-field techniques. Spatial interpolation algorithms calculate the lines by interpolating the adjacent lines from the same field. Spatial interpolation may be vertical (only pixels from upper and lower lines in vertical direction are considered), such as *line averaging*, or directional (a higher number of pixels from up and down lines are evaluated in several edge directions) [3]-[8].

Temporal interpolation algorithms interpolate the missing lines by employing pixels from different fields. Among them, the simplest methods are: *field insertion* (where lines from the previous field of the sequence are inserted), and *t-line average* (that performs the average value between lines from the previous and posterior fields of the video sequence).

Inter-field techniques work properly in the static parts of the image. However, if the image contains dynamical areas, these methods produce undesired effects (lines get misaligned) in moving objects. On the other hand, intra-field techniques work much better in the presence of motion, but offer poor results in static regions. This circumstance is the basic idea of motion adaptive de-interlacing algorithms, which were originally proposed in [9]. This kind of algorithms tries to combine an spatial or intra-field method and a temporal or inter-field method according to the presence of motion [10].

248858 (www.mobydic-project.eu) from European Community, TIN2005-08943-C02-01 and TEC2008-04920 projects from the Spanish Government (with support from the European Regional Development Fund), and P08-TIC-03674 project from the Andalusian Regional Government.

The weakest point of motion adaptive de-interlacing algorithms is the correct detection of the motion level. The output signal of a motion detector may be not always null in areas where there is no motion. Fundamentally this is due to the presence of noise, but some systems have also additional problems. For instance, interlacing causes false motion in vertically detailed parts and timing jitter of the sampling clock is particularly harmful in horizontally detailed areas. To avoid these effects, some motion detectors usually include any kind of spatio-temporal filtering [11].

The universal approximation capability of fuzzy systems has been exploited to interpolate images [12]-[14] and video sequences [15]-[17], [18]-[20]. For de-interlacing purpose, several proposals have been reported in the literature. The proposal in [18] uses soft-decision fuzzy logic techniques to remove noise and de-interlace TV video signals. A fuzzy edge-direction detector is introduced in [19] to orient a conventional vertico-temporal filter for de-interlacing. Other possible option is to use fuzzy logic and to apply different heuristic rules with approximate levels of uncertainty, which implicitly performs a non-linear filtering [20]. The algorithm in [15]-[17] uses a fuzzy inference rule base to choose de-interlacing strategy according to the detection of motion. More recently, several proposals based on fuzzy techniques have been reported to perform de-interlacing by enhancing edges in the sequence [3]-[7]. A fuzzy logic-based approach that performs a weighted spatio-temporal de-interlacing is presented in cite[21].

This paper is organized as follows: Section 2 summarizes the starting algorithm of our study. Section 3 describes the proposed algorithms. The evaluation of their performance and its comparison with other de-interlacing techniques are established in Section 4. The hardware implementation of the most efficient proposal called ‘single convolution algorithm’ is detailed in Section 5. Finally, some conclusions are expounded in Section 6.

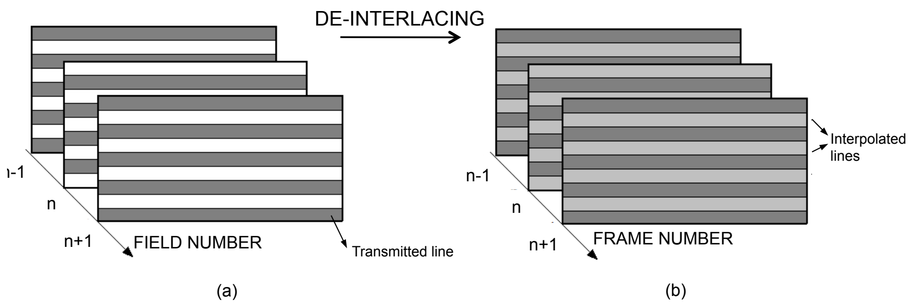

Fig. 1. (a) Interlaced video signal. (b) De-interlaced video signal.

2 Description of the starting algorithm

The algorithm proposed by Van de Ville et al. in [15]-[17] describes heuristic knowledge by means of a fuzzy rule set to look for an efficient trade-off between *line averaging* and *field insertion*, gradually adapted to the level of motion detected in every pixel. The approach presented by Van de Ville et al. will be called Fuzzy Motion Adaptive (FMA) algorithm in this contribution to simplify its citation. FMA achieves a good quality of the de-interlaced sequence but it becomes expensive, in terms of computational cost. A derived fuzzy approach from these contributions was proposed by Sanz et al. in [22]. This proposal was oriented to software implementation and presents an adaptive method that splits the corresponding fuzzy motion detector into 1D filters and linear simulation functions. Additionally, the involved saturation parameters were on-line adjusted taking into account the global frame motion.

Our proposal is inspired by it but it improves its performance and reduces its cost considerably. In order to facilitate the understanding of the FMA, let us first describe it briefly.

The luminance function of a video sequence is denoted by a three dimensional function $I(x, y, t)$ where x and y are the cartesian co-ordinates of the pixels in every frame, and t is the field number in the sequence. As images are digitized in pixels, x, y and t are discrete variables defined over natural numbers. Initially, our case of study is a monochromatic video signal but the procedure is easily extended to color image, considering the luminance of the joint RGB signal and applying the obtained correction to each one of the RGB components.

As temporal method, *field insertion* operation is chosen, which can be expressed as follows:

$$I_T(x, y, t) = I(x, y, t-1) \tag{1}$$

while *line averaging* is selected as spatial method, which is expressed as:

$$I_S(x, y, t) = (I(x, y-1, t) + I(x, y+1, t))/2 \tag{2}$$

In this way, the luminance of the pixels of the missing lines will be calculated and adapted to local motion, as follows:

$$I(x, y, t) = (1 - \gamma(x, y, t)) \cdot I_T(x, y, t) + \gamma(x, y, t) \cdot I_S(x, y, t) \tag{3}$$

where $\gamma(x, y, t)$ is a value in the interval [0,1] which represents an estimation of current motion in the pixel (x, y) and is obtained as the consequent of a set of fuzzy rules for motion estimation. Then, as higher is the motion (higher value of $\gamma(x, y, t)$), higher is the weight of spatial interpolation over temporal interpolation and vice-versa.

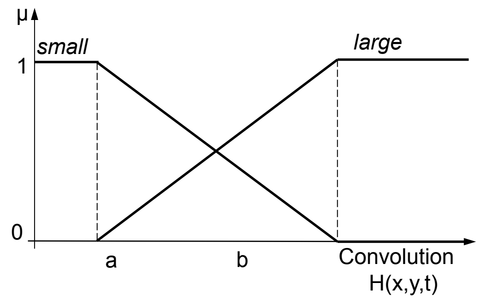

Fig. 2. Fuzzy partition composed by linguistic terms 'small' and 'large'.

The rules of FMA for getting $\gamma(x,y,t)$ take the frame difference signal as the input space. The frame difference signal is defined as:

$$H(x,y,t) = \left|(I(x,y,t+1) - I(x,y,t-1)\right|/2 \tag{4}$$

which denotes the variation in the luminance of every pixel.

The set of rules proposed in FMA for motion detection has the following meaning:

(1) When the frame difference signals in the neighborhood of the current pixel is *small*, the consequent states there is no motion present.
(2) When there are *large* frame difference signal only at left side of the current pixel, the consequent still states there is no motion present.
(3) When there are *large* frame difference signal only at right side of the current pixel, the consequent still states there is no motion present.
(4) When there are *large* frame difference signals at both sides of the current pixel, the consequent states there is motion present.
(5) When the frame difference signal at the current pixel is *large*, the consequent also assumes there is motion present.

where 'frame difference' is a linguistic variable whose values, *small* and *large*, are represented by the fuzzy sets depicted in Fig. 2. Furthermore, the use of piece-wise linear functions will ease the hardware implementation of the algorithm.

These rules are also executed in the upper and lower lines of the previous field in order to consider a rectangular window around the current pixel. The final value of $\gamma(x,y,t)$ is obtained after applying a defuzzification process, which is expressed as follows:

$$\gamma(x,y,t) = \frac{\pi_{(M(x,y,t)=true)}}{\pi_{(M(x,y,t)=true)} + \pi_{(M(x,y,t)=false)}} \tag{5}$$

where $\pi_{M(x,y,t)=true}$ and $\pi_{M(x,y,t)=false}$ are defined as the combined plausibility

Table 1
Fuzzy rule set for 2 neighbors at each side of the current pixel

Rule	Antecedents	Consequent
1.	$(H(x{+}2,y,t)$ *is small*$)$ *AND* $(H(x{+}1,y,t)$ *is small*$)$ *AND* $(H(x,y,t)$ *is small*$)$ *AND* $(H(x{-}1,y,t)$ *is small*$)$ *AND* $(H(x{-}2,y,t)$ *is small*$)$	$M(x,y,t)=false$
2.	$(H(x{+}2,y,t)$ *is small*$)$ *AND* $(H(x{+}1,y,t)$ *is small*$)$ *AND* $(H(x,y,t)$ *is small*$)$ *AND* $((H(x{-}1,y,t)$ *is large*$)$ *OR* $(H(x{-}2,y,t)$ *is large*$))$	$M(x,y,t)=false$
3.	$((H(x{+}2,y,t)$ *is large*$)$ *OR* $(H(x{+}1,y,t)$ *is large*$))$ *AND* $(H(x,y,t)$ *is small*$)$ *AND* $(H(x{-}1,y,t)$ *is small*$)$ *AND* $(H(x{-}2,y,t)$ *is small*$)$	$M(x,y,t)=false$
4.	$((H(x{+}2,y,t)$ *is large*$)$ *OR* $(H(x{+}1,y,t)$ *is large*$))$ *AND* $((H(x{-}1,y,t)$ *is large*$)$ *OR* $(H(x{-}2,y,t)$ *is large*$))$	$M(x,y,t)=true$
5.	$H(x,y,t)$ *is large*	$M(x,y,t)=true$

of the presence or absence of motion respectively, as follows:

$$\pi_{(M(x,y,t)=true)} = max(M_{(x,y,t)=true}, M_{(x,y-1,t-1)=true}, M_{(x,y+1,t-1)=true}) \quad (6)$$

$$\pi_{(M(x,y,t)=false)} = min(M_{(x,y,t)=false}, M_{(x,y-1,t-1)=false}, M_{(x,y+1,t-1)=false}) \quad (7)$$

$M_{(x,y,t)=true}$ and $M_{(x,y,t)=false}$ are defined as the consequents that state 'true' and 'false' motion respectively, and are calculated by combining the activation degree, α_i, of the corresponding rules as follows:

$$M_{(x,y,t)=false} = max(\alpha_1(x,y,t), \alpha_2(x,y,t), \alpha_3(x,y,t)) \quad (8)$$

$$M_{(x,y,t)=true} = max(\alpha_4(x,y,t), \alpha_5(x,y,t)) \quad (9)$$

The complexity of FMA approach depends on the number of the considered pixels at each side of the current pixel in the same line. If this number is denoted with the parameter K, the total number of antecedentes in the five rules are $2K+1$. FMA rules when the parameter K equals two are shown in Table 1.

Van de Ville et al. described the efficiency of their proposal by de-interlacing several sequences [15]-[17]. After analyzing different values of parameters K, a and b (a and b define the transition zone of the membership functions in

Fig. 2), they conclude, and we have corroborated, that the best results and, therefore, the lower errors[2], are obtained for K=2 and $0 \leq a \leq 1$; $6 \leq b \leq 9$.

FMA introduces important advantages over other conventional motion adaptive methods but it implies a high computational complexity since it requires for its implementation a high number of max-min operators, a high number of fuzzification processes (as many as the number of antecedents in the rules), and one division as shown the expression in (5). Our proposal is inspired by FMA, but it considerably reduces its computational cost and introduces two improvements to achieve a superior performance. Firstly, the use of convolution techniques allows the system to evaluate better the presence of motion in contrast to the use of max-min norms in FMA, which forces the system only to consider the extreme values. And the second one is to assign weights to the values of differences matrix. It seems logical to assign a higher value to pixels that are closer to the current pixel location.

3 Inference by convolution

This section describes our proposal which applies convolution techniques to perform rule inference. This is not a procedure for this particular application but it is rather a general method specially useful in the case of a high number of antecedents. Furthermore, convolution is a natural and very frequent operation in signal processing, and a great effort has been dedicated to increase efficiency for its execution, both at hardware and software levels. Besides, convolution can be computed as a product in the frequency space, getting great efficiency by using Fast Fourier Transform (FFT) algorithms.

In the case of two-dimensional signals, discrete convolution *c(x,y)* of two functions *w(x,y)* and *s(x,y)* is defined as:

$$c(x,y) = w(x,y) * s(x,y) = \sum_{n=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} w(m,n) \cdot s(x-m, y-n) \qquad (10)$$

If we choose for *w(x,y)* in (10) a rectangular function of height 1, that is:

[2] In order to be able to compare the efficiency of different algorithms and configurations, quantifying image quality is carried out considering the Mean Square Error (MSE) between original image and processed one. Since it is a standard measurement and considers all the image pixels.

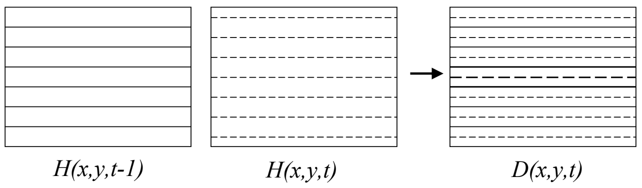

Fig. 3. Line insertion in the differences matrix

$$
\begin{aligned}
&w(x,y) = 1, \quad if \quad \begin{pmatrix} -K_l \leq x \leq K_r \\ -L_u \leq y \leq L_d \end{pmatrix} \\
&else \quad w(x,y) = 0
\end{aligned}
\tag{11}
$$

the convolution of *s(x,y)* and *w(x,y)* in (11) would be the addition of all the elements of *s(x,y)* within the window. Furthermore, if the height of *w(x,y)* is changed to $1/[(K_l{+}K_r{+}1)\times(L_u{+}L_d{+}1)]$, the result of the convolution would be the mean value of *s(x,y)* within the window.

Consequently, all the union operations in the rules may be globally implemented by means of convolution. Intersection operations must be products in this case. To convert products into convolutions is feasible by application of De Morgan's laws, since intersections are converted into unions. That is, convolution of a window on the complementary of the matrix of differences H(x,y,t) will provide an estimation of not-motion within this particular window.

According to this idea, the inference mechanism required by the FMA algorithm may be implemented by a bi-dimensional convolution. As it was described previously, the algorithm estimates motion by evaluating pixel differences in vertical and horizontal directions. Our proposal realizes a bidimensional convolution using the global differences matrix ($D(x,y,t)$) which includes all the lines of the current field ($H(x,y,t)$) plus the lines from the previous field ($H(x,y,t-1)$). This matrix ($D(x,y,t)$) is calculated by interweaving as shown in Fig. 3.

$$D(x,y,t) = H(x,y,t) + H(x,y,t-1) \tag{12}$$

Although our proposals could consider any window size, three lines are used since this size offers a good trade-off between complexity and performance. Furthermore, FMA algorithm works with three lines and the same number of lines has to be used in order to establish a fair comparison.

3.1 The approach based on convolution with five rules

This approach calculates the motion-adaptative parameter $\gamma(x, y, t)$ in equation (5). Let us consider, for example, the second rule of FMA (see Table 1). It is evaluated by using the same neighbors that were used by FMA algorithm, that is, a window size of 3x5. Since the first part of the antecedent is the estimation of motion *small* in the right side of the window, including the current pixel, we will make convolution (properly speaking, correlation) of the global differences complementary matrix (*D'(x,y,t)* that evaluates 'not motion') as follows:

$$C2a = \frac{1}{9}\begin{pmatrix} 0\;0\;1\;1\;1 \\ 0\;0\;1\;1\;1 \\ 0\;0\;1\;1\;1 \end{pmatrix} * D^{'}(x, y, t) \tag{13}$$

where the factor 1/9 is to make averaging instead of addition. Since the values of the global differences matrix are restricted in the interval [0,1], the complementary of the matrix of differences will be made as follows:

$$D^{'}(x, y, t) = TOP - D(x, y, t) \tag{14}$$

where TOP is the maximum value of the differences matrix. For example, if luminance is encoded with eight bits, the maximum value is 255. The other part of the antecedent of the second rule is the estimation of motion *large* in the left part of the convolution window (excluding the current pixel), expressed by using the following equation:

$$C2b = \frac{1}{6}\begin{pmatrix} 1\;1\;0\;0\;0 \\ 1\;1\;0\;0\;0 \\ 1\;1\;0\;0\;0 \end{pmatrix} * D(x, y, t) \tag{15}$$

Now, the consequent of this second rule is obtained by choosing an aggregation operator for making intersection of $C2a$ and $C2b$. For example, minimum, product, averaging in the form of geometric mean, etc... Any case, the membership function to make fuzzification must be suitable for the chosen aggregator. This procedure is extended to the rest of the rules used by the FMA algorithm (see Table 2). For instance, the third rule will be implemented with

Table 2
Fuzzy rule set for 2 neighbors at each side of the current pixel

Rule	Antecedents	Consequent
1.	$(C1 = \frac{1}{15}\ (11111; 11111; 11111)\ {*D}'(x,y,t))\ is\ large$	$M(x,y,t) = false$
2.	$(C2a{*D'}(x,y,t))\ AND\ C2b{*D}(x,y,t))\ is\ large$	$M(x,y,t) = false$
3.	$(C3a{*D}(x,y,t)\ AND\ C3b{*D'}(x,y,t))\ is\ large$	$M(x,y,t) = false$
4.	$(C4 = \frac{1}{12}\ (11011; 11011; 11011)\ {*D}(x,y,t))\ is\ large$	$M(x,y,t) = true$
5.	$(C5 = \frac{1}{3}\ (00100; 00100; 00100)\ {*D}(x,y,t))\ is\ large$	$M(x,y,t) = true$

the following convolutions:

$$C3a = \frac{1}{6}\begin{pmatrix} 0\ 0\ 0\ 1\ 1 \\ 0\ 0\ 0\ 1\ 1 \\ 0\ 0\ 0\ 1\ 1 \end{pmatrix} * D(x,y,t) \tag{16}$$

$$C3b = \frac{1}{9}\begin{pmatrix} 1\ 1\ 1\ 0\ 0 \\ 1\ 1\ 1\ 0\ 0 \\ 1\ 1\ 1\ 0\ 0 \end{pmatrix} * D^{'}(x,y,t) \tag{17}$$

The value of $\gamma(x,y,t)$ is calculated by applying the expression in equation (5), but the values of plausibility are evaluated by combining the activation degree, α_i, of the rules in Table 2 as follows:

$$\pi_{(M(x,y,t)=true)} = \alpha_4 \cdot \alpha_5 \tag{18}$$

$$\pi_{(M(x,y,t)=false)} = \alpha_1 \cdot \alpha_2 \cdot \alpha_3 \tag{19}$$

3.2 Single Convolution algorithm

Instead of considering uniform weights within the matrix to perform convolution, as shown in expressions (13), (15)-(17), it is possible to use variable weights without additional computational cost. This provides to give more significance to some positions with respect to the others, for example, it seems logical to give more weight to those positions closer to the current pixel.

Taking into account their linguistic meaning, all the FMA rules can be summarized in only one sentence: *if there is a large frame difference at both sides*

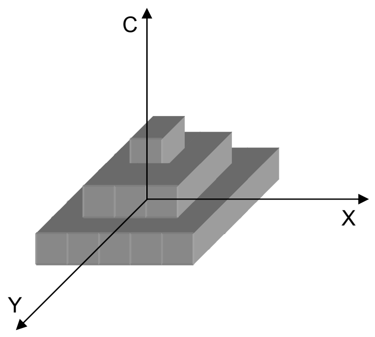

Fig. 4. Model of matrix C for the algorithm of single convolution for single convolution

Table 3
Fuzzy rule set for 2 neighbors at each side of the current pixel

Rule	**Antecedents**	**Consequent**
1.	$(C * D(x,y,t))$ *is small*	$M(x,y,t) = false$
2.	$(C * D(x,y,t))$ *is large*	$M(x,y,t) = true$

of the current pixel or there is a large frame difference in the current pixel then motion is present. As a consequence, only the 4th and 5th FMA rules are evaluated. To perform both rules by using a unique bi-dimensional convolution, a new convolution mask is proposed. Fig. 4 shows a model to illustrate the shape of the proposed matrix called C. This matrix considers neighbors all around the current pixel and the weight of each neighbor depends on the distance to the current pixel.

With this new algorithm the number of rules is reduced to only two as shown the Table 3, but only the second rule that corresponds to the presence of motion is calculated to evaluate $\gamma(x, y, t)$ since, as mentioned above, the fuzzy concepts *small* and *large* are complementary (see Fig. 1) :

$$\gamma(x, y, t) = \pi_{(M(x,y,t)=true)} = \alpha_2 \tag{20}$$

The improvement in terms of computational efficiency is indubitable since the number of antecedents and rules of the algorithm have been considerably reduced. The improvement in execution time is difficult to quantify since it always depends on the degree of quality of the designed software, the efficiency of the used compilers, and, of course, the employed hardware. At this moment, all our computer programs have been coded in Matlab. However, our measurement about execution time are extremely cautious, since it is very hard to estimate real computing power when a high level language is being used. Any case, for a sequence of twelve frames, Matlab executes the algorithm

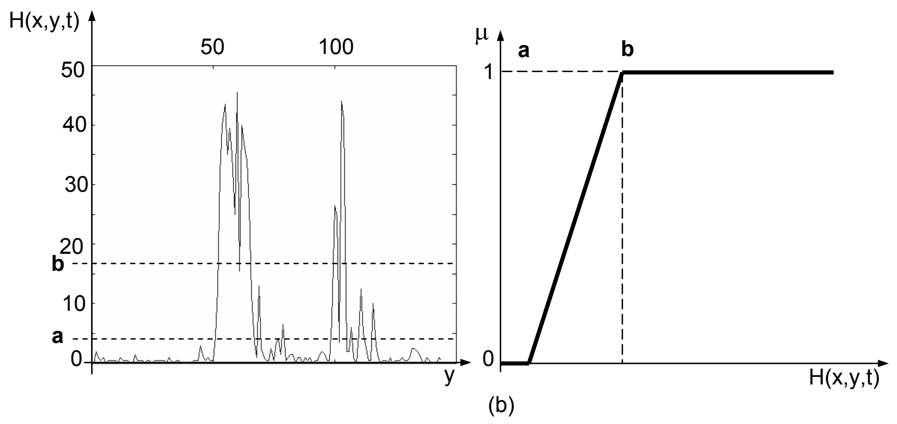

Fig. 5. Loss of information with the process of fuzzification

of Van de Ville et al. in 138 s (seconds), the algorithm that uses convolution techniques with five rules in 4.8 s, and the single convolution algorithm in 1.4 s. The elimination of loops by working on a two dimensional matrix and the efficiency of convolution makes the developed algorithms to give rise additional improvements. Concerning to the employed hardware, if execution is made by general purpose processors, the execution time is directly related to the computational power of the processor. However, efficiency can be drastically improved by means of specific hardware as it is detailed in Section 5.

One important aspect of the procedure is the step in which the fuzzification is applied. In FMA algorithm, fuzzification is made every time an input variable is introduced in the antecedents of any rule (1-5). That is, each value of the matrix of differences is initially evaluated with certain fuzzy sets to determine the membership degree of that value to *small* or *large* (see Fig 2). However, each time a fuzzification operation is made, certain loss of information is produced as illustrated in Fig. 5. In principle, there is no restriction to select the shape of the membership functions and other nonlinear fuzzy partitions could be considered. Although the hardware complexity of our system is considerable reduced if the analysis is limited to the linear ones. This is a reasonable hypothesis in low-complexity motion detection systems for real-time video applications

A more accurate result can be achieved if the fuzzification process is performed after a global estimation of all the rules, that is, after calculating the convolution, as employed by our proposals. Therefore, the final value of $\gamma(x, y, t)$ is the result of fuzzifying the global estimation and, in general terms, we can say that the input variables of the fuzzy system is not the global differences matrix, but the global differences matrix once convolved with C.

The quality of both proposals have been tested with the video sequence *Salesman*, also used as a benchmark sequence in [15]-[17]. Fig. 6 shows a small portion of a de-interlaced frame of this sequence after applying the procedures of

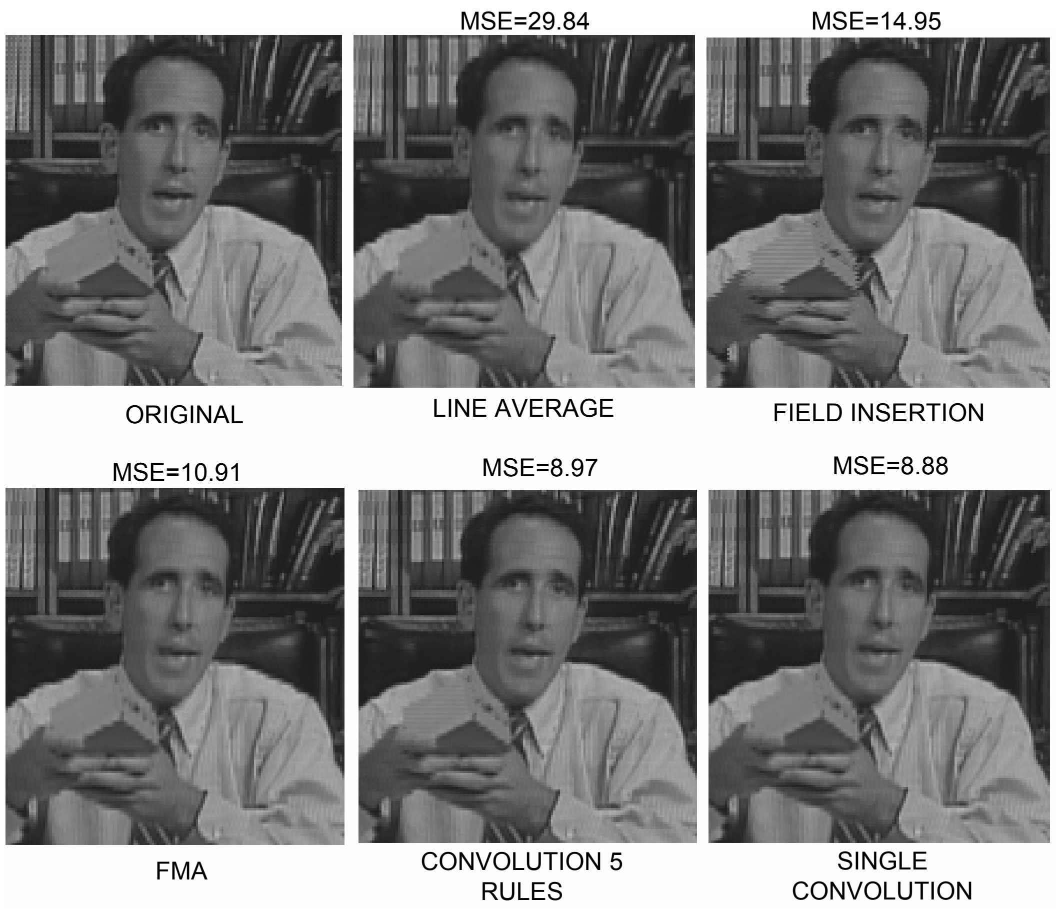

Fig. 6. Comparative results of the developed algorithms

line average, *field insertion*, FMA algorithm, and the developed algorithms of convolution techniques with five rules and single convolution. Measurements by means of mean squared error (MSE) have been indicated over the images in Fig. 6 for this field of the sequence. *Line average* gives a MSE of 29.84, *field*

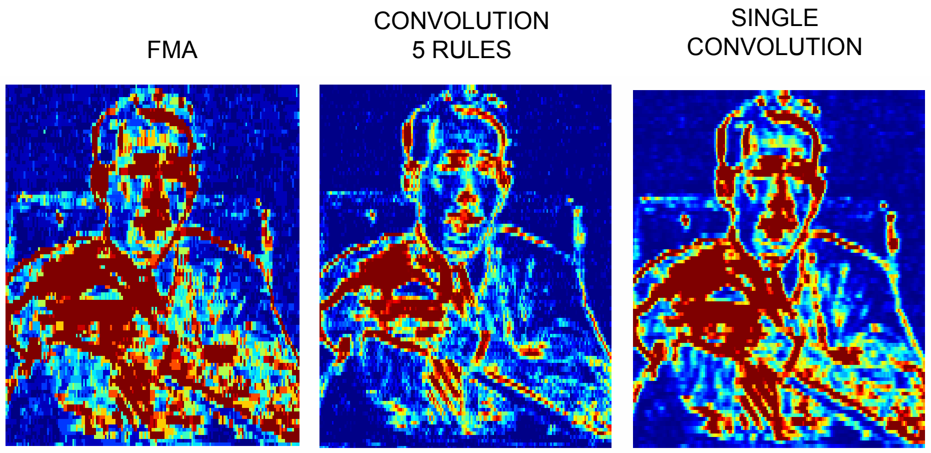

Fig. 7. Comparison about motion estimation

insertion 14.95, while the FMA algorithm gives 10.91, convolution technique with 5 rules 8.97, and single convolution technique 8.88. Finally, the diagrams of motion estimation in Fig. 7 show the major discrimination capacity of the proposed algorithms. From the simulation results obtained by both proposals, we conclude that the single convolution algorithm offers the most attractive solution between the quality of the resultant interpolated images and complexity, and this is why it has been selected to extend our study in Section 4.

4 Comparison with conventional de-interlacing algorithms: quality and complexity

Before presenting data, it is necessary to indicate that the quality of results depends on the kind of image, specially in what concerns to dynamical features. Other subject is that all the motion adaptive algorithms under comparison have a great deal of possible variants, as those relative to logical fuzzy operators used in the inference process previously mentioned. At the same time, all of them are dependent on adjusting parameters, as the shape of fuzzy sets *large* and *small*, size of window, location and weighting in the matrices to make convolution and so on.

Table 4 shows the average MSE obtained when de-interlacing 50 fields of seven video sequences with three different formats: TV(720x576), CIF(352x288) and QCIF(176x144). These are standard video sequences that have been widely used as benchmarks in video processing applications. The results shown in Table 4 correspond to the proposed algorithm using the single convolution with the following matrix of coefficients:

$$C = \begin{pmatrix} 1 & 2 & 3 & 2 & 1 \\ 1 & 3 & 5 & 3 & 1 \\ 1 & 2 & 3 & 2 & 1 \end{pmatrix} \tag{21}$$

The proposed algorithm has been compared with other conventional algorithms [1]: four spatial methods such as line doubling, *line averaging* and the conventional ELA are used. ELA was reported in [23] and it is a well-known edge adaptive de-interlacing algorithm, which consists of applying the average of the luminance values along the direction with the maximum correlation. The most adequate direction is selected by evaluating the absolute value of luminance differences in 3+3 (ELA 3+3) or in 5+5 (ELA 5+5) taps. A recent modified version of the ELA algorithm is presented in [24] to enhance de-interlacing in horizontal edges. Furthermore, the following de-interlacing techniques are selected: the simplest temporal de-interlacing algorithm (*field insertion*), two vertico-temporal methods with two and three fields [1] [3]; an implicit median-based technique that uses a three-point VT median lter; a

[3] These algorithms are implemented in many consumer equipments since they offer a good trade-off between complexity and performance.

Table 4
Average MSE for different de-interlacing methods

Sequence	**Tokyo**	**Paris**	**Trevor**	**Salesman**	**News**	**Mother**
Format	**TV**	**CIF**	**CIF**	**CIF**	**QCIF**	**QCIF**
Line doubling	123.33	283.19	51.05	68.87	197.27	42.86
Line averaging	46.46	139.98	20.37	28.84	77.28	16.56
ELA 3+3	64.72	182.01	25.23	40.01	141.27	18.79
ELA 5+5	91.21	223.39	30.34	62.52	166.37	37.42
Enhanced ELA [24]	61.95	178.68	25.23	37.85	123.33	16.41
Field insertion	14.59	67.15	23.82	15.706	31.62	15.81
VT 2fields	13.46	54.96	14.19	14.42	18.49	7.11
VT 3fields	19.95	47.43	12.504	13.12	17.62	5.29
Median technique [25]	20.94	61.11	18.62	14.19	28.97	9.21
Van de Ville et al. (FMA)	21.13	31.71	18.83	11.24	21.88	7.31
MC field insertion [1]	4.69	23.17	12.64	9.57	14.86	6.93
S.Convolution	11.01	19.27	13.93	9.64	11.53	4.22

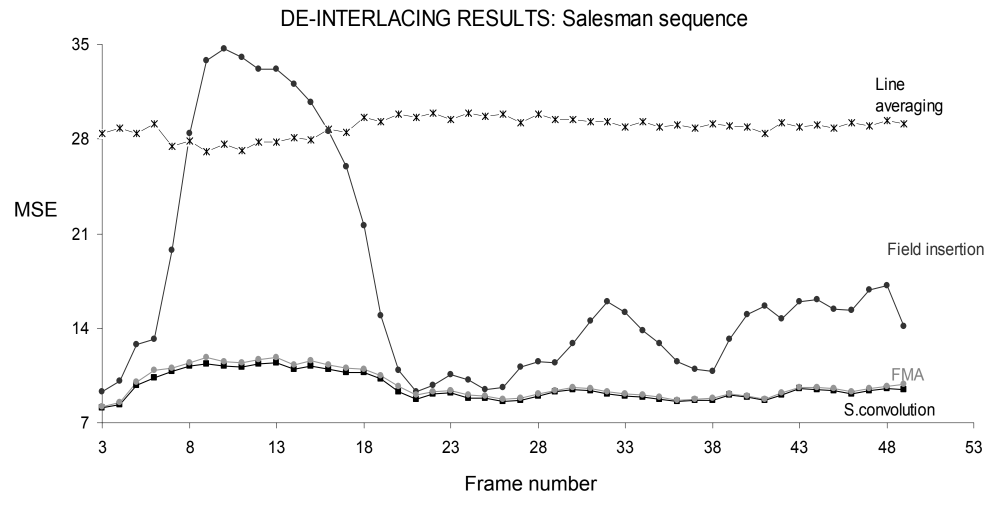

Fig. 8. MSE values obtained by the Salesman sequence

MC algorithm called ‘MC field insertion’; and, finally, the proposal of Van de Ville et al.

As it can be seen in Table 4, the proposed algorithm achieves the lowest MSE errors in three of the six sequences (Paris, News, Mother). The MC algorithm works better in two sequences (Tokyo and Salesman), whereas the VT technique with three fields slightly improves the results in the Trevor sequence.

For the Salesman sequence, the MSE value for each de-interlaced frame is shown in Fig. 8. This graph also proves the advantage of motion adaptive proposals that considerably reduces the error values by combining the spatial (*line average*) and the temporal (*field insertion*) techniques.

Complexity is analyzed using two figures of merit: 1) hardware resources to store luminance values of the pixels involved in the calculation; and 2) primitive operations (POs). Table 5 shows the storage devices that are required by each de-interlacing algorithm. The motion adaptive algorithms require field memories to evaluate the presence of motion. Specifically, FMA approach uses two field memories to evaluate the five rules for the current pixel and another more to store the consequents of the pixels in the previous field. To calculate the convolution, the single convolution proposal requires three fields memories. The second figure of merit is estimated by evaluating the number of POs to calculate an interpolated value. Addition, subtraction, shifting, absolute difference and sign function are considered as operation of complexity 1 PO, while multiplication and division are considered with complexity 2 POs. As can be seen in Table 5, the proposal considerably reduces the number of POs in comparison with the FMA approach, and slightly increases the POs of vertico-temporal algorithms. In terms of computational cost, the heaviest technique is the MC algorithm. The calculation of the number of POs is not included in Table 5 since it depends on the strategy used to calculate the motion vector and the size of the block processing. For instance, the implemented version of the MC field insertion used the 3-DRS algorithm described in [26] and it requires 529 POs.

Between simple linear de-interlacing algorithms and complex motion-compensated ones, motion-adaptive algorithms represent a suitable midpoint. The analysis of de-interlacing results shows the efficiency of the proposed motion- adaptive algorithm. Furthermore, it reduces the complexity of the FMA algorithm up to a percentage of 56% in terms of POs.

Table 5
Analysis of storage resources and primitive operations (POs) required by each de-interlacing algorithm

	Resources			Complexity
De-interlacing Algorithm	**Line buffer**	**Field memory**	**Register**	**POs**
Line doubling	-	-	-	-
Line averaging	1	-	-	2
ELA 3+3	1	-	2	11
ELA 5+5	1	-	4	20
Enhanced ELA [24]	1	-	2	16
Field insertion	-	1	-	-
VT 2fields	2	1	-	16
VT 3fields	2	2	-	12
Median technique [25]	1	1	-	11
Van de Ville et al. (FMA)	2+1	2	4	103
S.convolution	1	3	4	45

5 Hardware implementation

Hardware implementation of the single de-interlacing algorithm has been developed with the tool *XSG* (Xilinx System Generator) [27]. This tool consists of a Simulink library, called *Xilinx blockset*, and software to translate a Simulink model into a hardware realization of the model described in VHDL language. *XSG* maps the system parameters (defined like mask variables in Xilinx blockset blocks) into entities, architectures, ports, signals, and attributes in the hardware realization.

Three different system architectures have been considered to implement the single convolution de-interlacing algorithm. They differ at the level of parallelism employed to implement the convolution. The first one is a completely parallel architecture in which thirty values of luminance, stored in memory, are necessary to calculate the luminance of each pixel in the new inserted line (Fig. 9(a)). As a result, a value is obtained for the pixel in a clock period. The second design (see Fig. 9(b)) employs a mixed architecture which carries out the operations in sequential form for the five values of each row of the differences matrix, whereas operations from different columns of the difference

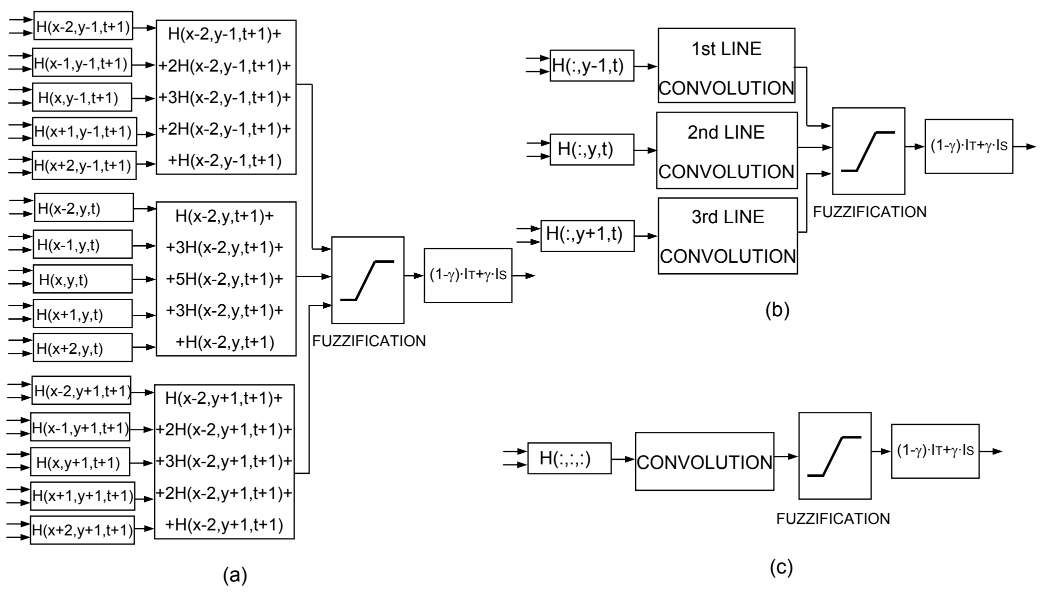

Fig. 9. (a) Parallel, (b) mixed and (c) sequential implementation architectures developed with XSG.

matrix are realized in parallel. A result is obtained after five clock periods in this case. The third design uses a totally sequential architecture (Fig. 9(c)). In this case a new value of luminance is calculated after fifteen clock periods.

Fuzzification process is performed after convolution in these three architectures. This strategy provides two advantages: firstly, fuzzification always implies a lost of information, and the idea is to process convolution without information loss. Secondly, this strategy simplifies the architectures since a unique fuzzification block is used independently of the selected architecture (see Fig. 9).

To give a physical medium to the different hardware implementation architectures of the algorithm, a RC100 Celoxica board has been employed. This board is specially suitable to implement algorithms for video processing applications. It includes a medium size FPGA from Xilinx called Spartan II. Since the algorithm does not involve a huge computational cost, this FPGA should be adequate to achieve real-time applications. Furthermore, the RC100 includes a XCR3128XL CPLD and two 36-bit x 256k location independent synchronous RAM banks, in which three correlative fields are stored, which are necessary for the calculus of the algorithm.

The system includes a CVBS video input and a video decoding chip (SAA7111 from Philips), enabling the FPGA to capture and decode NTSC and PAL video sources. All control signals are directly mapped with the pins of the FPGA. Finally, the board has the ability to generate 24-bit color VGA output to be displayed on a monitor. The board features a 24-bit Video DAC to convert the digital output from the FPGA to the appropriate analog signals on the VGA connector.

Celoxica provides a development environment which uses the Handel-C language for hardware description. Thus, it incorporates a library of Handel-C macros and functions designed to make it possible to start producing Handel-C designs right away on the RC100 Celoxica board [28]. This library includes macros for accessing each bank of SRAM and video drivers for the D/A converter which provides the VGA output and for the video decoder SAA7111.

The high level of our design is a description in Handel-C of the behavior of the system. The generated code describes the capture of video signal through the CVBS video input and the writing process of data stored in memories. The reading process from SRAM provides the input to a black box defined in Handel-C which integrates the design of the de-interlacing algorithm described by XSG. Data of the new lines are ordered in a process to be able to display the data on a monitor through the VGA output. Fig. 10(a) shows a block diagram with the processes implemented on the FPGA.

Both HDLs descriptions are synthesized individually. The VHDL description obtained from XSG is synthesized with FPGA Express Compiler II from Synopsys. On the other hand, the Handel-C description is synthesized with $DK1$, the software tool provided by Celoxica. Both designs are incorporated at a post-synthesis level when the implementation of the global system is carried out. Fig. 10(b) shows a block diagram of the tools used in the design.

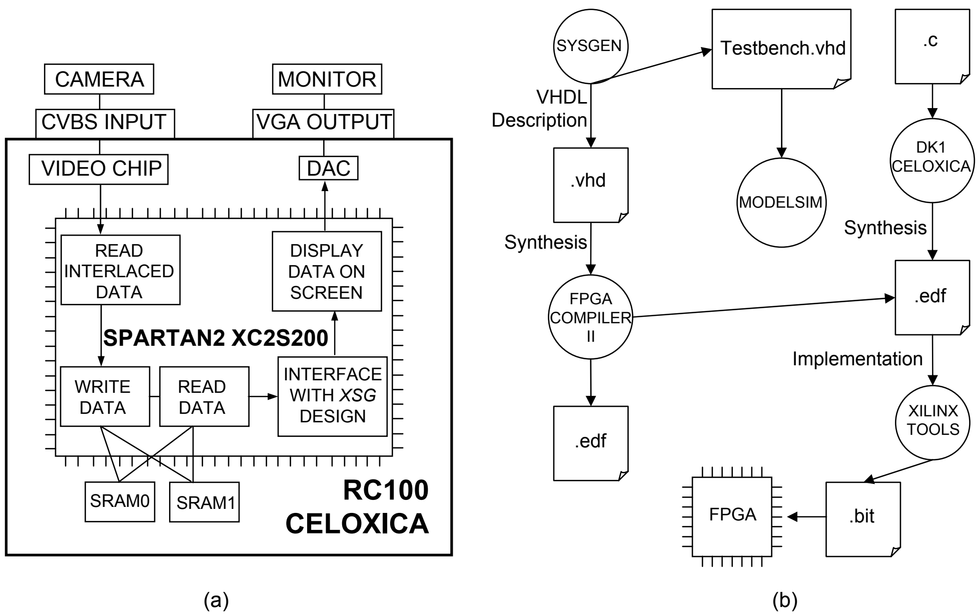

Fig. 10. (a) Block diagram of processes implemented in the FPGA. (b) Tools used in the development of the design.

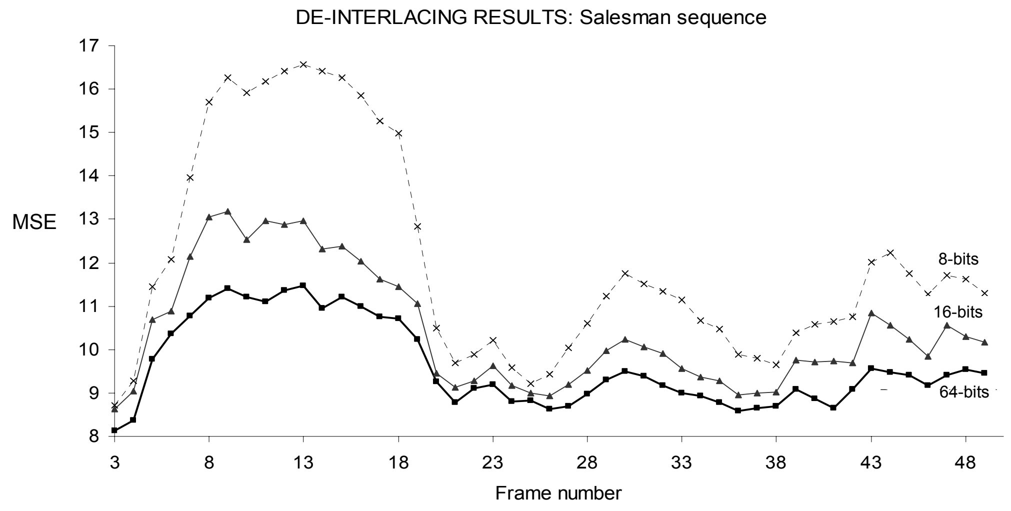

Fig. 11. MSE values obtained by several implementations of the single convolution algorithm.

5.1 Implementation results

The different alternatives for implementing the single convolution de-interlacing algorithm can be evaluated from results obtained in the Simulink environment or by modeling the VHDL description generated by XSG with the ModelSim tool from Mentor Graphics. MSE results achieved by the XSG designs for the Salesman sequence is shown in Fig. 10.

As can be seen from the results in Fig. 10, MSE value is higher for the hardware implementations since the algorithm described in Matlab works with double-precision numbers (64-bits) whereas XSG models has been implemented with

Table 6
Implementation results of the designs generated with XSG

	FPGA	**SPARTAN2 xc2s200**	
XSG **Design**	**No. slices** *XSG* **design**	**No. slices Global design**	**Processing time (ns) / (MHz)**
Parallel 8 bits	588(25%)	1906(81%)	36/27.7
Parallel 16 bits	894(38%)	2216(94.2%)	39/25.6
Mixed 8 bits	196(8.3%)	1514(64.4%)	115/8.7
Mixed 16 bits	381(16.9%)	1698(72.2%)	125/6.9
Sequential 8 bits	98(4.16%)	1424(60.5%)	345/2.8
Sequential 16 bits	233(9.9%)	1550(65.9%)	390/2.6

integer numbers of 8 and 16 bits. Obviously, the implementation with 16-bits achieve lower errors than the implementation with 8-bits.

Implementation results in terms of device utilization are shown in Table 6. Analyzing these results, the number of slices for the sequential design is obviously lower than the others and the parallel design requires the highest number of devices. Table 6 also includes the results obtained in terms of the occupational level of the FPGA when the global design written in Handel-C is implemented.

The Philips SAA7111 video decoding chip on the RC100 board is controlled using the I^2C bus in-system communication protocol. It provides a data rate of 13.5 MHz and thus, this forces the system to compute a new interpolated pixel value at 27 MHz. According to the timing results in Table 6, the suitable architecture for a real-time implementation in this hardware platform would be the 8-bit parallel one.

6 Conclusions

A fuzzy de-interlacer for video sequences proposed by Van de Ville et al. has been the inspiration to propose other algorithms for video de-interlacing based on convolution techniques.

Motion estimation is crucial to adjust the interpolation technique. The level of motion is evaluated with a fuzzy system and its performance depends on the quality of the input variables. A global matrix that considers inter and intra-field pixel differences is used to obtain robust input variables.

The developed techniques consist on making convolution of the input variables with a selected function that gets a weighted averaging among them, carrying out a kind of fuzzy union. Fuzzy intersection may be computed in a similar way and with similar level of complexity, by the application of convolution techniques.

The developed de-interlacers are essentially two: one of them makes use of the same set of rules of the starting algorithm and applies convolution in the inference. The other takes advantage from the fact that the function to make convolution is able to configure a weighting scheme among the input variables and simplifies the set of rules in only one convolution (single convolution algorithm). This single convolution approach has been implemented on a FPGA development board using the Xilinx's System Generator tool in combination with the Handel-C development environment provided by Celoxica. Three different system architectures, which differ at the level of parallelism employed to implement the convolution, have been considered. Timing results prove that an 8-bit parallel implementation of the algorithm is capable of provide real-time operation for this hardware platform.

The quality of the results are better than these obtained by the starting algorithm. However, the most significant improvement has been the reduction of the computational complexity.

References

[1] G. de Haan - De-interlacing. Chapter book of Digital Video Post Processing, pp. 185-201, University Press Eindhoven, Sep. 2006.

[2] Y-L. Chang, S-F. Lin, C-Y. Chen, L-G. Chen - Video de-interlacing by adaptive 4-field global/local motion compensated approach. IEEE Trans. on Circuits and Systems for Video Technology, vo. 15, no.12, pp 1569-1582, 2005.

[3] G. Jeon, M. Anisetti, V. Bellandi, J. Jeong - Fuzzy rule-based edge-restoration algorithm in HDTV interlaced sequences. IEEE Trans. on Consumer Electronics, vol. 53, no.2, pp 725-731, May 2007.

[4] G. Jeon, R. Lee, D. Kim, J. Lee, J. Jeong - Weighted fuzzy filter on interlaced-to-progressive conversion. Proc. IEEE International Conference on Multimedia and Expo, pp 173-176, Apr. 2008.

[5] G. Jeon, Y. Fang, K. Lee, M. Y. Jung, R. Lee, J. Jeong - Video deinterlacing algorithm based on fuzzy reasoning with angle extraction approach. Studies in Computational Intelligence, vol. 226, pp. 369-379, 2009.

[6] G. Jeon, M. Anisetti, V. Bellandi, E. Damiani, J. Jeong - Designing of a type.2 fuzzy logic filter for improving edge-preserving restoration of interlaced-to-progressive conversion. Information Sciences, vol. 179, no. 13, pp. 2194-2207, Jun. 2009.

[7] G. Jeon, M. Anisetti, J. Lee, V. Bellandi, J. Jeong - Concept of linguistic variable-based fuzzy ensemble approach: Application to interlaced HDTV sequences. IEEE Trans. on Fuzzy Systems, vol. 17, no. 6, pp. 1245-1258, Dec. 2009.

[8] S.-J. Park, G. Jeon, J. Jeong - Deinterlacing algorithm using edge direction from analysis of the DCT coefficeint distribution. IEEE Trans. on Consumer Electronics, vol. 55, no. 3, pp.1674-1684, 2009.

[9] A. M. Bock - Motion adaptive standards conversion between formats of similiar field rates. Signal Processing: Image Communication, vol. 6, no. 3, pp. 275-280, Jan. 1994.

[10] C. Stiller, S. Kammel, J. Horn and T. Dang - The computation of motion. Digital Image Sequence Processing, Compression and Analysis. Todd R. Reed Ed. CRC Press, 2005.

[11] D. Wang, A. Vincent, P. Blanchfield - Hybrid de-interlacing algorithm based on motion vector reliability. IEEE Trans. on Circuits and Systems for Video Technology, vol.15, no. 8, pp. 1019-1025, 2005.

[12] H. C. Ting, H. M. Hang - Spatially adaptive interpolation of digital images using fuzzy inference. Proc. SPIE, vol. 2727, pp 1206-17, Mar. 1996.

[13] N. Shezaf, H. Abromov-Segal, I. Sutskoner, R. Bar-Sella - Adaptive low complexity algorithm for image zooming at fractional scaling ratio. Proc. 21st IEEE Convention of the Electrical and Electronics Engineers, pp 253-256, Tel Aviv (Israel), Apr. 2005.

[14] T. Aso, N. Suetake, T. Yamakawa - A code-reduction technique for an image enlargement by using a sum-based fuzzy interpolation. Proc. 9th Int. Conf. on Neural Information Processing (ICONIP), vol. 3, pp 711-721, Torino (Italy), Aug. 1989.

[15] D. Van de Ville, B. Rogge, W. Philips, I. Lamahieu - De-interlacing using fuzzy-based motion detection. Proc. Int. Conf. on Knowledge-Based Intelligent Information Engineering Systems, 1999.

[16] D. Van de Ville, B. Rogge, W. Philips, I. Lamahieu - Evaluation of several operators for fuzzy-based motion adaptive de-interlacing. Proc. of the PRORISC IEEE Benelux Workshop on Circuits, Systems and Signal Processing, pp 535-544, Nov. 1999.

[17] D. Van de Ville, R. Van de Wall, W. Philips and I. Lamahieu - Motion adaptive de-interlacing using fuzzy logic. Proc. of International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems(IPMU), pp 1989-1996, Jul. 2002.

[18] M. Mancuso, V. D'Alto, R. Poluzzi - Fuzzy edge-oriented motion-adaptive noise reduction and scanning rate conversion. Proc. IEEE Asia-Pacific Conference on Circuits and Systems (APCCAS), pp 652-656, Dec. 1994.

[19] F. Michaud, C. T. Le Dinh, G. Lachiver - Fuzzy detection of edge-direction for video line doubling. IEEE Trans. on Circuits and Systems for Video Technology, vo. 7, no. 3, pp 539-542, Jun. 1997.

[20] H. Jiang, D. Huu, E. Tinyork and M. Vasquez - Motion Adaptive De-interlacing. United States Patent (US 6,459,455), Oct. 2002.

[21] G. Jeon, J. You, J. Jeong - Weighted fuzzy reasoning scheme for interlaced to progressive conversion. IEEE Trans. on Circuits and Systems for Video Technology, vol. 19, no. 6, pp 842-855, Jun. 2009.

[22] A. Sanz, F. Fernández, J. Gutiérrez-Ríos, G. Triviño, A. Sánchez, J.C. Crespo, A. Mazadiego - Video deinterlacing using adaptive fuzzy filters. Applied Computational Intelligence - Proc. of the 6th International FLINS Conference, pp 397-402, 2004.

[23] T. Doyle, M. Looymans - Progressive scan conversion using edge information. Signal Processing of HDTV, II. L. Chiariglione ED., Elsevier Science Publishers, pp.711-721, 1990.

[24] H. Y. Lee, J. W. Park, T. M. Bae, S. U. Choi, Y. H. Ha - Adaptive scan rate up-conversion system based on human visual characteristics. IEEE Trans. on Consumer Electronics, vol.46, no.4, pp 999-1006, Nov. 2000.

[25] P. Haavisto, Y. Neuvo - Motion adaptive scan rate up-conversion. Multidimensional Systems Signal Processing, no. 3, pp 113-130, 1992.

[26] G. de Haan - Motion estimation, chapter book of Video Processing, pp 221-269, University Press Eindhoven, 2004.

[27] Xilinx - Xilinx System Generator v2.1 for Simulink User Guide. (http://www.mathworks.com/applications/dsp_comm/xilinx_ref_guide.pdf), 2004.

[28] Celoxica - RC100 Function Library Manual. 2002