A NEW EMBEDDED SYSTEM FOR ADAPTIVE SPATIAL VIDEO DE-INTERLACING ALGORITHMS BASED ON FUZZY LOGIC

[1]Ms. BOYAPATI BHARATHIDEVI, [2]Mr. LAKSHMI PRASAD CHENNAMSETTY,
[1]Assistant professor, [2]Assistant professor
[1]Department of ECE
[1]UNIVERSAL COLLEGE OF ENGINEERING & TECHNOLOGY

ABSTRACT: Video de-interlacing algorithms perform a crucial task in video processing. Despite these algorithms are developed using software implementations, their implementations in hardware are required to achieve real-time operation. This paper describes the development of an embedded system for video de-interlacing. The algorithm for video de-interlacing uses three fuzzy logic-based systems to tackle three relevant features in video sequences: motion, edges, and picture repetition. The proposed strategy implements the algorithm as a hardware IP core on a FPGA-based embedded system. The paper details the proposed architecture and the design methodology to develop it. The resulting embedded system is verified on a FPGA development board and it is able to de-interlace in real-time.

Keywords: Embedded system, video de-interlacing, Fuzzy system, IP core

1. INTRODUCTION

Video de-interlacing is a key task in digital video processing. Some current digital transmission standards use interlaced scan format to halve video bandwidth. However, modern display devices require a progressive scan. Therefore, algorithms to interpolate the missing lines during the transmission have to be implemented at the receiver side. This kind of algorithms are called de-interlacing since they perform the reverse operation of interlacing [1].

Digital video processing chain usually includes the four stages shown in the block diagram of Figure 1:

- Reception: video TV is received by a TV decoder, which is in charge of decoding the analog or digital signal.
- Removal of artifacts: analog signal suffers white gaussian noise whereas digital signal is affected by video compression artifacts, which introduces two kinds of artifacts: 'block' and 'mosquito' noise.
- Conversion of resolution: signal is firstly converted from interlaced to progressive. After performing deinterlacing, an algorithm for down or up scaling is applied accordingly to the format required by the device.
- Picture improvement: the quality of video signal is enhanced by applying several picture enhancement algorithms such as color improvement, sharpness or edge enhancement, and improvements of contrast, details and textures.

Despite video processing algorithms are developed using software implementations, their implementations in hardware are required to achieve real-time operation. Although primitive consumer equipments had several ASICs.

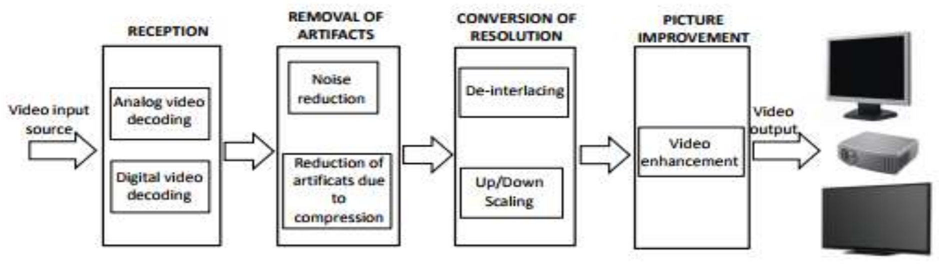


Figure 1. The four typical stages of digital video processing

to perform each of these four tasks, the current high-end products in the market contain a highly integrated chip to perform the last three tasks and even all the tasks in Figure 1. Video processing chips are included in numerous consumer devices, such as Liquid Crystal Display (LCD) TVs, plasma TVs, Audio-Video (AV) receivers, DVD players, High Definition (HD)-DVDs, Blu-ray players/recorders, and projectors. Independently of the video source (during the last years multiple video sources are proliferating), these chips normally perform intelligent and costly de-interlacing methods to obtain a high quality output progressive signal.

Concerning video de-interlacing, several choices for hardware implementations are currently available in the market, such as Application-Specific Integrated Circuits (ASICs) [2]-[9], programmable solutions on Digital Signal Processing (DSPs) [10]-[14] and/or Field Programmable Gate Arrays (FPGAs) [15]-[18], and Intellectual Property (IP) cores [19]-[23] that can be used as building blocks within ASICs or FPGA designs. Recently, application specific instruction-set processors (ASIPs) for high speed computation of intra-field de-interlacing have been also proposed [24]. Each of these alternatives offers advantages and disadvantages regarding high performance, flexibility and easy upgrade, and low development cost.

In commercial equipments, ASICs offer the most efficient solution justified by the huge demand of video products. In the other side, FPGAs are a good solution when there is a low volume of consumer products or in the case that the aim is to develop a prototype. Furthermore, FPGAs address with the requirement of flexibility and easiness to upgrade. Taking into account these considerations, FPGA implementation is the option chosen herein to implement the de-interlacing algorithm on an embedded system.

There are many proposals of fuzzy logic-based embedded systems for control applications [25]-[26]. Recent advances in de-interlacing algorithms propose the inclusion of soft computing techniques to increase the picture quality [27]-[31]. An embedded system that implements the algorithm of [27] in real-time is proposed in this work. To the best of our knowledge this is the first hardware implementation of a fuzzy logic-based video de-interlacing algorithm. This paper details the implementation strategy that consists of an IP core on a FPGA-based embedded system. The paper is organized as follows. Section 2 summarizes a description of the algorithm and the proposed architecture to develop its hardware implementation. Section 3 explains the design methodology to build the IP core for video de-interlacing. The development of the embedded system and its validation on a development board from Xilinx is detailed in Section 4. Finally, the conclusions of this work are expounded in Section 5

II. ARCHITECTURE OF FUZZY IP CORE FOR VIDEO DE-INTERLACING.

The algorithm for video de-interlacing is the result of combining three fuzzy logic-based systems, each of them tackling a relevant feature: motion, edges, and possible repetition of areas in fields. The edge-adaptive system is called spatial interpolator since it performs a non-linear interpolation among pixels in the spatial neighborhood. The system that is capable of detecting repeated areas in the fields is called temporal interpolator since it interpolates pixels in the temporal neighborhood. The third system combines the outputs of the spatial and temporal interpolators according to a motion measurement in the current pixel. A block diagram of the complete system is shown in Figure 2.

The three fuzzy systems are simple since they contain a low number of inputs and a low number of rules in their rulebases. The spatial interpolator (denoted as FS1 in Figure 2) is an edge-adaptive algorithm that uses five potential

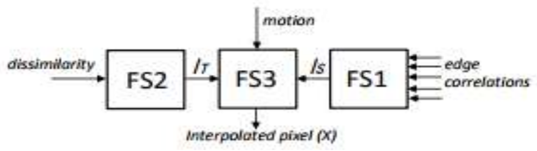


Figure 2. Block diagram of the video de-interlacing algorithm that employs the three fuzzy logic-based system

Table 1. Analysis of storage resources and POs of several state-of-the-art de-interlacers

De-interlacing Algorithm	No. of field memories	No. of Primitive Operations (POs)
MC Field Insertion [32]	1	529
MC VT Filtering [32]	1	536
MC TBP [32]	2	535
MC TR [32]	2	529
MC AR [32]	2	544
GST [32]	1	543
Robust GST [32]	1	555
GST-2D [32]	2	559
Robust GST-2D [32]	2	571
Algorithm in [27]	3	153

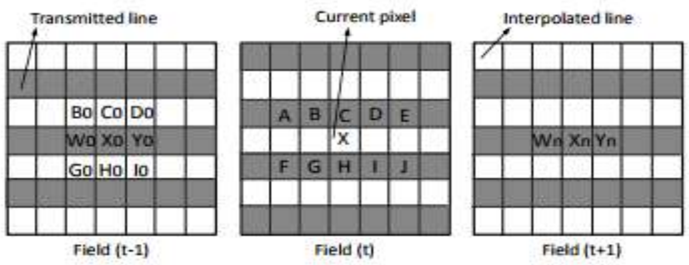


Figure 3. Pixels used for de-interlacing

edge directions and six fuzzy rules to adapt the interpolation to the presence of edges. The second interpolator (FS2) is a fuzzy area-repetition-dependent temporal interpolator that uses a simple convolution to measure the dissimilarity between consecutive fields. It employs two simple fuzzy rules to adapt interpolation to repetition of areas in fields. Despite its simplicity, it reduces considerably annoying artifacts such as feathering. The feature of motion is considered by a fuzzy motion-adaptive interpolator (FS3) that uses a simple convolution to measure the motion at each pixel. This third system evaluates how this motion measurement should influence on the interpolation decisions.

The algorithm for video de-interlacing employs an off-line tuning process to obtain the values of the parameters in the fuzzy systems as detailed in [27]. The tuning stage has been successfully performed by using a supervised learning algorithm that minimizes the mean square error between a set of data corresponding to progressive and de-interlaced results of different standard sequences.

A detailed description of the complete system and its comparison (in terms of PSNR and visual inspection) with other state-of-the-art de-interlacers are presented in [27]. The algorithm is able to improve the results obtained by several Motion-Compensated (MC) algorithms in areas of the images with small and large motion, with clear and unclear edges, and with film and video material mixed. Concerning embedded system solutions, the hardware implementation of this algorithm is advantageous over the considered MC algorithms, as depicted in Table 1. The number of Primitive Operations (OPs) required is quite low. The number of field memories is three since it is necessary to store the values of the interpolated pixels in the previous field (as shown in Figure 3).

Table 2. Rule base of the fuzzy system for motion-adaptive interpolator

Rule	Antecedents	Consequent
1.	$motion(x,y,t)$ is $SMALL$	$I_T(x,y,t)$
2.	$motion(x,y,t)$ is $SMALL-MEDIUM$	$\gamma_1\ I_T(x,y,t)+\lambda_1\ I_S(x,y,t)$
3.	$motion(x,y,t)$ is $MEDIUM$	$\gamma\ I_T(x,y,t)+\lambda\ I_S(x,y,t)$
4.	$motion(x,y,t)$ is $MEDIUM-LARGE$	$\gamma_2\ I_T(x,y,t)+\lambda_2\ I_S(x,y,t)$
5.	$motion(x,y,t)$ is $LARGE$	$I_S(x,y,t)$

2.1. Description of fuzzy logic-based systems for video de-interlacing

2.1.1. Fuzzy logic-based system for motion adaptation (FS3)

Motion-adaptive approaches are based on the fact that linear temporal interpolators are perfect in the absence of motion, whereas linear spatial methods offer a most adequate solution in case that motion is detected [1]. Motion detection can be implicit, as in median-based techniques, or explicit, using a motion detector. Explicit motion-adaptive algorithms calculate a new pixel value by interpolating between a spatial and a temporal de-interlacing method:

$$I_p = (1 - \alpha)I_T + \alpha I_S \quad (1)$$

where IS is the output of a spatial interpolator, and IT is the output of a temporal interpolator. The variable α, which is the output of a motion detector, ranges from 0 to 1 and determines the level of motion. The performance of explicit motion-adaptive algorithms relies on the quality of the motion detector, since it is strongly dependent on the combination of both de-interlacing algorithms.

The fuzzy system (FS3) performs the non-linear interpolation between a spatial and a temporal interpolator according to the presence of motion. This fuzzy system for motion adaptation has one input, which is a motion measurement based on the use of bi-dimensional convolution between a matrix of picture differences, M, and a matrix of weights, C. For a current pixel (X), the input of this fuzzy system is as follows:

$$motion = \frac{\Sigma_{a=1}^{3}(\Sigma_{b=1}^{3} M_{a,b}C_{a,b})}{\Sigma_{a=1}^{3}\Sigma_{b=1}^{3} C_{a,b}} \quad (2)$$

where M(i, j) are the elements of the following matrix, M, of difference values among three consecutive fields or pictures (see Figure 3):

$$M = \begin{pmatrix} M_{(-1,-1)} & M_{(0,-1)} & M_{(1,-1)} \\ M_{(-1,0)} & M_{(0,0)} & M_{(1,0)} \\ M_{(-1,1)} & M_{(0,1)} & M_{(1,1)} \end{pmatrix} = \begin{pmatrix} \frac{|B-B_0|}{2} & \frac{|C-C_0|}{2} & \frac{|D-D_0|}{2} \\ \frac{|W_n-W_0|}{2} & \frac{|X_n-X_0|}{2} & \frac{|Y_n-Y_0|}{2} \\ \frac{|G-G_0|}{2} & \frac{|H-H_0|}{2} & \frac{|I-I_0|}{2} \end{pmatrix} \quad (3)$$

And Ca,b are the elements of the following weight matrix (their values have been adjusted to ease hardware implementation):

$$C = \begin{pmatrix} 0\,1\,0 \\ 1\,2\,1 \\ 0\,1\,0 \end{pmatrix} \quad (4)$$

The first step of the fuzzy inference process is called fuzzification. It consists of determining the degree to which the input belongs to each of the appropriate fuzzy sets via the chosen membership functions. The shape of the membership functions is piece-wise linear since this selection eases the hardware implementation of the algorithm (see Figure 4). Because of linguistic coherence, the degree of overlapping between two consecutive sets is two.

The rule base of the inference system is strongly related to the selection of the number of membership functions (see Table 2). The first rule states that 'if motion(x, y, t) is SMALL then the interpolated pixel is calculated by applying a temporal interpolator (IT)'. On the contrary, the rule number five asserts that 'if motion(x, y, t) is LARGE the interpolation is performed by applying a spatial interpolator (IS)'. The other three rules consider intermediate situations that, when they are activated, perform different linear combinations of the spatial and the temporal interpolators.

The final step to calculate the value of the interpolated pixel is the defuzzification process. Among the defuzzification methods, the Fuzzy Mean (FM) has been chosen since it is a simplified method that allows hardware simplicity. It

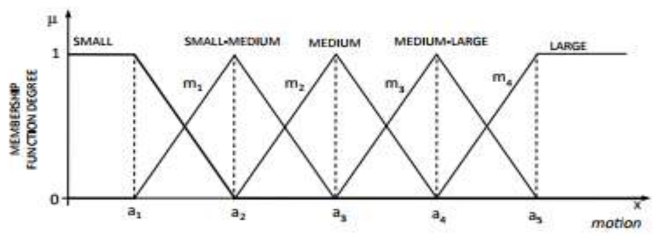


Figure 4. Normalized piecewise linear functions used for the membership functions. The parameters that define the Z, triangular, and S functions are break points (ai) and slopes (mi)

Table 3. Rule base of the fuzzy system for edge-adaptive interpolation

Rule	Antecedents	Consequent
1.	*b* is *SMALL* and *c* is *LARGE* and *d* is *LARGE*	$X = \frac{B+I}{2}$
2.	*b* is *LARGE* and *c* is *LARGE* and *d* is *SMALL*	$X = \frac{D+G}{2}$
3.	*b* is *VERY_SMALL* and *c* is *LARGE* and *d* is *VERY_SMALL*	$X = \frac{B+D+G+I}{4}$
4.	*a* is *SMALL* and *b* is *LARGE* and *c* is *LARGE* and *d* is *VERY_LARGE* and *e* is *VERY_LARGE*	$X = \frac{A+J}{2}$
5.	*a* is *VERY_LARGE* and *b* is *VERY_LARGE* and *c* is *LARGE* and *d* is *LARGE* and *e* is *SMALL*	$X = \frac{E+F}{2}$
6.	*Otherwise*	$X = \frac{C+H}{2}$

consists of a weighted average of the rule consequents, c r (see Table 2), where the weights are the activation degrees, α r , of the corresponding rules:

$$FM = \frac{\sum_r \alpha^r \cdot c^r}{\sum_r \alpha^r} \qquad (5)$$

2.1.2. Fuzzy logic-based system for edge adaptation (FS1)

The fuzzy system (FS1) for spatial interpolation takes inspiration from well-known conventional edge-adaptive de-interlacing algorithms. This kind of techniques explores a neighborhood of the current pixel to extract information about the edge orientation [33]. Among them, Edge-based Line Average (ELA) algorithm interpolates the new pixel value by analyzing the luminance differences in the upper and lower lines. ELA looks for the most possible edge direction and then applies 'line average' along the selected direction. The pseudo-code of the ELA algorithm with 5+5 taps is as follows (see Figure 3):

```
$if\ min(a,b,c,d,e) = b \rightarrow X = (B+I)/2$
$elseif\ min(a,b,c,d,e) = d \rightarrow X = (D+G)/2$
$elseif\ min(a,b,c,d,e) = a \rightarrow X = (A+J)/2$          (6)
$elseif\ min(a,b,c,d,e) = e \rightarrow X = (E+F)/2$
$else\ \rightarrow X = (C+H)/2$
$where:\ a = |A-J|,\ b = |B-I|,\ c = |C-H|,\ d = |D-G|,\ e = |E-F|$
```

ELA performs well when the edge direction agrees with the maximum correlation, but otherwise introduces mistakes and degrades the image quality. The fuzzy inference system proposed in [27] is inspired by the ELA scheme but it uses a rule-based inference system to overcome ELA limitations.

The inputs of this fuzzy system are the edge correlations in the five directions (a, b, c, d, e). As the starting point to design the rules of the fuzzy system heuristic knowledge expressed linguistically has been applied (see Table 3):

1. An edge is clear in direction b not only if b is small but also if c and d are large.

2. An edge is clear in direction d not only if d is small but also if b and c are large.

Table 4. Rule base for the temporal interpolator

Rule	Antecedents	Consequent
1.	*dissimilarity*(x, y, t) is *SMALL*	X_0
2.	*dissimilarity*(x, y, t) is *LARGE*	X_n

3. If b and d are very small and c is large, neither there is an edge nor vertical linear interpolation performs well; the best option is a linear interpolation between the neighbors with small differences: B, D, G, I.

4. If three antecedents are large, that is, no edge is found in the three directions (b, c, and d). An edge is clear in direction a , if a is very small and e is very large.

5. If three antecedents are large, that is, no edge is found in the three directions (b, c, and d). An edge is clear in direction e, if e is very small and a is very large.

6. Otherwise, a vertical linear interpolation would be the most adequate.

This heuristic knowledge is fuzzy since the concepts of 'small', 'large', and 'very small' are not understood as threshold values but as fuzzy ones. Fuzzy sets SMALL, LARGE, VERY SMALL, and VERY LARGE are again represented by fuzzy sets with a piecewise linear functions. The overlapping of the membership functions has been selected to ensure that no more than two rules are simultaneously activated. This strategy allows the use of the minimum operator as connective 'and' since a positive value for the activation degree of the sixth rule is always obtained. FM defuzzification method is used to calculate the output value. The output of the FS1 is the spatial interpolator, (IS), which is used in the complete algorithm (see Figure 2).

2.1.3. Fuzzy logic-based system for picture-repetition adaptation (FS2)

The simplest temporal interpolator is named 'field insertion', and it consists of repeating the pixel with same spatial coordinates in the previous picture. This technique introduces annoying artifacts such as feathering when the previous pixel is not a good option. This is especially noticeable in film sequences when the previous field does not correspond to the same but to a different frame, which has motivated an active research field on film-mode detectors.

Since de-interlacing should be able to cope with any kind of material: video, film and hybrid (currently very popular due to the explosion of multimedia market), the idea developed in [27] is to adapt locally temporal interpolator to the presence of repeated areas in the fields as spatial interpolator was adapted locally to the presence of edges. In this sense, area repetition, as edges and motion, is considered, in general, a local (as happens to video and hybrid sequences) and fuzzy feature of the image and, hence, two simple fuzzy rules are proposed to implement a pixel-bypixel fuzzy selection between pixels in the previous (t-1) and posterior (t+1) pictures. The input of these rules is a measure of dissimilarity between consecutive fields. Taking advantage of previous experience, dissimilarity between two consecutive fields, which is calculated by using a bi-dimensional convolution (as in the case of motion):

$$dissimilarity = \frac{\Sigma\Sigma M_{(i,j)} C'_{(i,j)}}{\Sigma\Sigma C'_{(i,j)}} \qquad (7)$$

where M(i, j) are the elements of the matrix, M, defined in (3) and C 0 (i, j) are the coefficients of the convolution mask:

$$C' = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} \qquad (8)$$

The influence of dissimilarity in selecting the kind of temporal interpolation is evaluated by considering the following fuzzy rules (see Table 4):

1. If dissimilarity between the fields (t-1) and (t) is SMALL, the most adequate interpolated value is obtained by selecting the pixel value in the previous field at the same spatial position (I(x, y, t − 1)).

2. On the contrary, if dissimilarity is LARGE, the pixel value in the previous field is not a good choice and is better to bet on the pixel in the next field (I(x, y, t + 1)).

The shape of membership functions to model the fuzzy concepts SMALL and LARGE are also piece-wise linear functions. The output of this fuzzy system is given by applying the Fuzzy Mean defuzzification method.

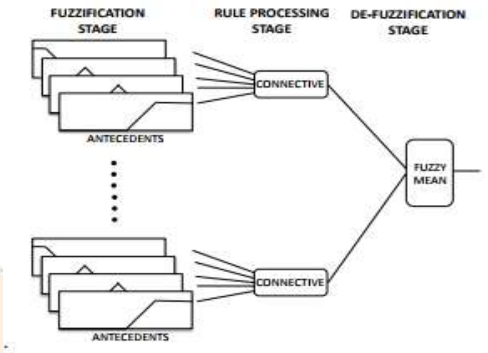


Figure 5. Parallel rule processing architecture

2.2. Hardware implementation of the fuzzy inference systems

The proposed architecture to implement the fuzzy inference systems provides a data path for each rule. This strategy is very adequate when the rule bases contain a low number of rules and it provides a high inference speed since the rules are processed in parallel. The modules that compose the proposed architecture are a memory, an inference unit, a control circuit, and a defuzzification block. The memory stores the values to define the membership functions of the antecedents and the consequents. Since the fuzzy systems have a reduced number of rules, the memory size required is not large. The connective AND in the antecedents of the rules is represented by minimum operator. The inference can be performed in parallel since there is not a high number of the rule antecedents. The architecture has multi-input operators dedicated to aggregate the antecedents of the rules. These operators are designed by the cascade connection of two-input operators. The architecture introduces pipeline stages, which separate independent operations, and allow an overall improvement of the inference speed.

The general structure of the architecture consists of three fundamental stages: the fuzzification stage, which calculates the membership degree for each of the inputs; the rule processing stage, in which the activation degrees of the rules are computed, and the defuzzification stage, which provides the system output. A block diagram of the proposed rule processing architecture is illustrated in Figure 5.

2.2.1. Fuzzification stage

There are two well-known strategies to implement the membership functions: memory-based approach and arithmetic-based approach. Memory-based approaches store the membership degrees of every possible input value into a memory. The input value acts as the address of the memory to retrieve the fuzzy labels that are activated and the corresponding membership functions. The main advantage of this kind of approaches is that there is no restriction in the shape of the functions. The main drawback is the memory size, which can be very large to implement high-resolution systems, since the size of memory increases exponentially with the bit number of the inputs and the number of the resolution bits to code the membership functions. The second approach is particularly appropriated if the shapes of the membership functions are piecewise linear. The antecedent memory only stores the parameters required to carry out the calculation of the functions: the break points and the slopes. The proposed architecture implements the membership functions by using the arithmetic-based approach in parallel. From a hardware point of view, this strategy requires more resources but it achieves a high inference speed for real-time purposes.

Three shape of membership functions are implemented: triangular, Z, and S functions (see Figure 4). From a hardware point of view, this selection has two advantages. The first one is that the parameters stored for each function are a point and a slope. The second one is that only one Membership Function Circuit (MFC) is required to calculate the two membership degrees per input because the maximum overlapping degree is two and the functions are normalized, so that calculating one of the degrees, μn, the other is obtained as 1- μn. The flow chart to implement

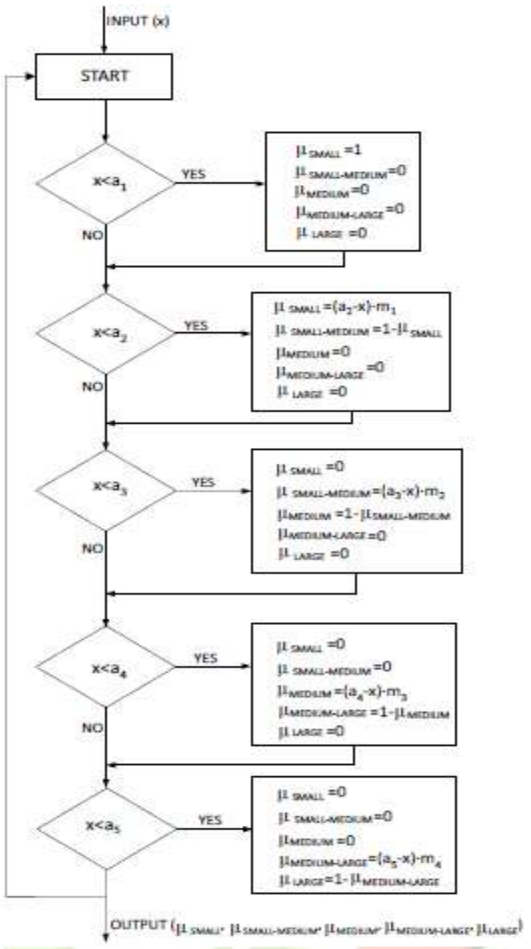


Figure 6. Flow chart used to implement the membership functions that are shown in Figure 4

the membership functions in Figure 4 is illustrated in Figure 6. The hardware resources required to implement these membership functions are a multiplier, an adder, and registers to store intermediate results.

2.2.2. Rule processing stage

A parallel architecture, which computes the activation degree of the rules, is chosen since the rule bases contain a low number of rules and there is a low number of the rule antecedents. This means that the implementation cost in terms of hardware resources is affordable. Furthermore, this solution provides the best performance in terms of timing since the proposed activation degrees of the rules are evaluated with the smallest number of clock cycles.

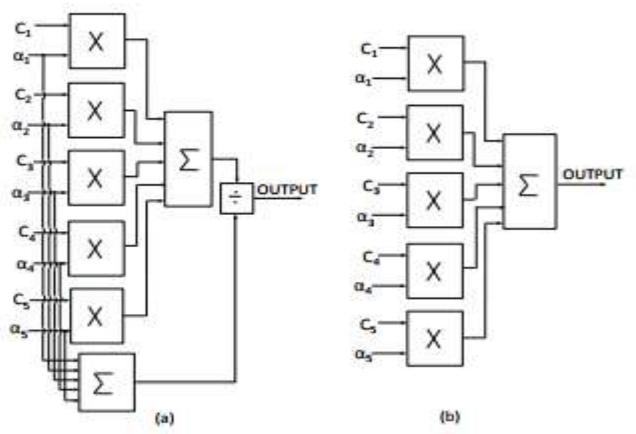


Figure 7. (a) Block diagram to implement the Fuzzy Mean defuzzification method in FS3. (b) The divider is suppressed since the sum of the activation degrees of the rules is equal to one.

2.2.3. Defuzzification stage

Figure 7(a) shows the block diagram of the parallel architecture to implement the Fuzzy Mean method that is employed by the fuzzy systems. It requires as many multipliers as the number of rules in the rulebase, adders and one divider. The divider in equation (5) can be suppressed if the sum of the activation degrees of the rules is equal to one. The shapes of all membership functions are chosen to ensure that no more than two rules are simultaneously activated and the sum of the activation degrees is always normalized. Hence, the expression in (5) is simplified as follows:

$$FM \rightarrow output = \sum_i \alpha_i \cdot c_i \qquad (9)$$

The block diagram to implement the method in (9) for the case of five rules (as is the case in FS3) is depicted in Figure 7(b).

III. DESIGN METHODOLOGY FOR FPGA-BASED FUZZY IP CORE

The hardware implementation of the three fuzzy systems that compose the algorithm for video de-interlacing has been done following the architecture detailed in Section 2.2. A methodology has been proven to implement the algorithm on Xilinx FPGAs. A design flow, which connects the abstract algorithmic level with its physical FPGA implementation, is depicted in Figure 8. At algorithmic level, Matlab and its Image and Video Processing Toolbox have been employed to develop the algorithms. Xfuzzy 3 [34] and its integrated tool called xfsl have been used to tune the parameters of the fuzzy rule bases. System Generator from Xilinx (XSG) has been employed to develop efficient hardware-description-language (HDL) definitions from the schematic diagram described by the designer. XSG consists of a specific blockset for Simulink, called Xilinx Blockset, and the necessary software to translate the mathematical model described in Simulink into a hardware model described in HDL 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 with VHDL. The hardware description developed with XSG has been verified with Simulink simulations of test video sequences stored in the Matlab workspace. Finally, CAD environments to easily develop FPGA designs from HDL descriptions (in particular, ISE from Xilinx) have been employed to develop the implementations that perform experimental validation. Furthermore, XSG allows the verification of the implementation on the FPGA introducing the inputs through the Matlab environment. This verification is also known as hardware-in-the-loop testing.

A picture of the top level description in the XSG design is shown in Figure 9. Since the design in XSG is integrated into Simulink, other elements from several blocksets can be used to verify the correct behavior of the design. For

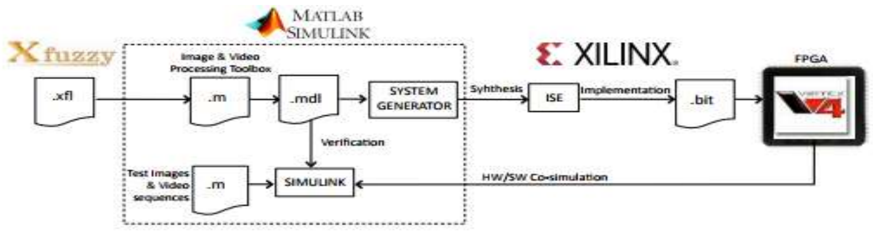


Figure 8. Computer-aided design methodology

instance, the XSG design to implement the fuzzification process for the fuzzy system in charge of implementing motion-adaptive de-interlacing is shown in Figure 10. The performance of this membership function generator block can be verified with a scope element as shown in Figure 9.

IV. DEVELOPMENT OF THE EMBEDDED SYSTEM

The previously described fuzzy IP core has been included into an embedded system that implements fuzzy logicbased de-interlacing algorithm meeting real-time requirements.

A Virtex-4 FPGA has been selected as the target device. It is included in the ML402 board of the Xilinx 'Virtex-4 Video Starter Kit' (VSK). In addition to the ML402 board, VSK contains the Video Input Output Daughter Card (VIODC) board, which is in charge of providing video sources and sinks to the ML402 board. Among the video sources, the camera and VGA signals are used, both with the same resolution (640x480 progressive). The VGA input signal is provided by a laptop. Among the available video sinks in the VIODC board, the VGA output is chosen. In order to store video output, a video capture board is used on a host PC. The host PC is also in charge of downloading the configuration files of the FPGAs (*.bit files) by using a parallel IV download cable. It also enables communication over a RS-232 serial port.

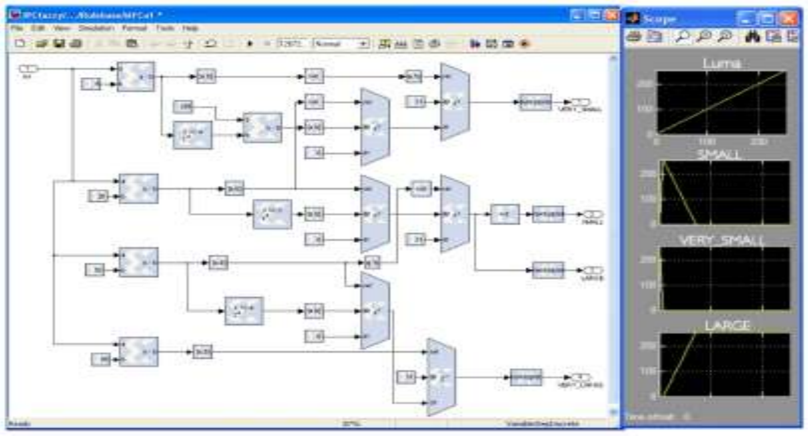


Figure 9. XSG design to implement the fuzzification process for the FS1

The system embedded in the Virtex-4 FPGA uses the Xilinx MicroBlaze [35] as general purpose processor in charge of controlling the data flow. The CAD tools of Xilinx Embedded Development Kit (EDK) have been employed to develop the software and hardware configurations of the embedded system. Three IP cores are connected to MicroBlaze by using a dedicated Fast Simplex Link (FSL) bus system based on FIFO communications. The FSL core called 'VIO' in Figure 11 is used to interface the various video inputs and outputs on the VIODC board. By means of the VIO, MicroBlaze controls all the video sources available on the VIODC board. The FSL core called 'DDR' in Figure 11 allows a parallel access to two external DDR memories, which store video frames. The latest core of the video processing channel is the 'fuzzy de-interlacing' IP core. This core is obtained by exporting from Simulink to EDK the XSG design described in the previous section as an IP core of MicroBlaze. MicroBlaze and all its cores run at a 100 MHz clock frequency to achieve real-time requirements. Figure 11 shows a block diagram of the complete system.

Table 5 shows the implementation results in terms of the FPGA occupational level. The implementation of the de-interlacing system occupies a percentage of approximately 80% of the slices on the Virtex-4 FPGA (this number considers the MicroBlaze and its three FSL cores). The 'fuzzy de-interlacing' core requires a very low number of resources (1211 slices, 7% in percentage). The resources required by the 'fuzzy de-interlacing' core are very low because the algorithm has been optimized to use simple operators (such as addition, minimum, and multiplication) and simple weighting factors that can be reduced to shift operators. In contrast, other de-interlacers proposed in the literature demand heavy arithmetic operations from a hardware point of view. For example, the spatial interpolator MELA in [36] requires a square root, a divider, and an arctan function. Another example is the spatial interpolator FWW in [37], which requires two dividers and a cos function. To illustrate the requirements of these operations, they have been implemented on the same Virtex-4 FPGA with the following results of slices utilization: one square root requires 511 slices, a divider 809 slices, a cos function 272 slices, and the arctan function 172 slices. Therefore, the implementation of any of these spatial interpolators requires more hardware resources than the complete implementation of the 'fuzzy de-interlacing' core.

Figure 12 shows a photo of the complete system while it is working. The embedded system is able to accept a progressive video signal coming from the camera or the VGA input (the VGA input is used in Figure 12), interlace it, and send the de-interlaced signal to the VGA output in real time.

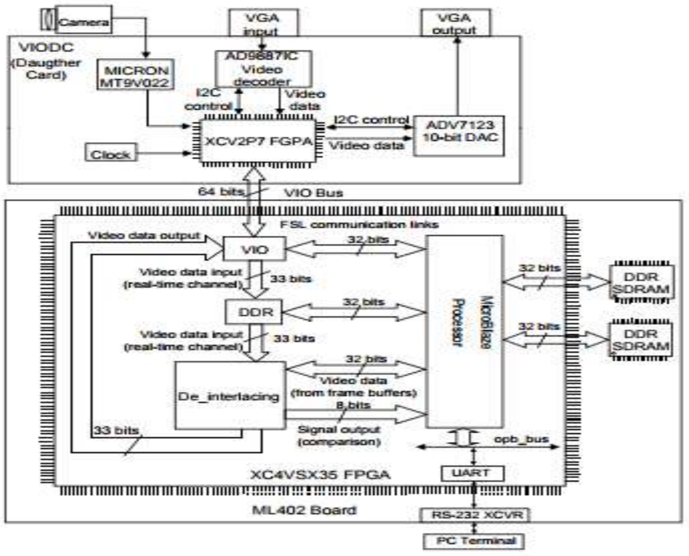
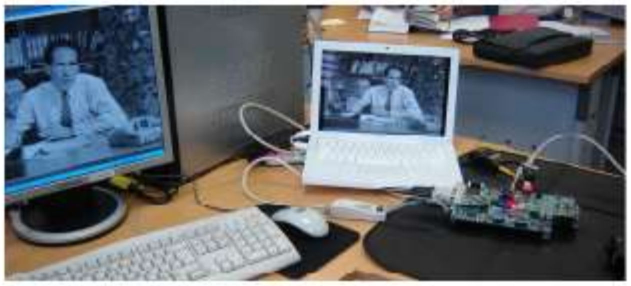


Figure 10. Block diagram of the complete system.



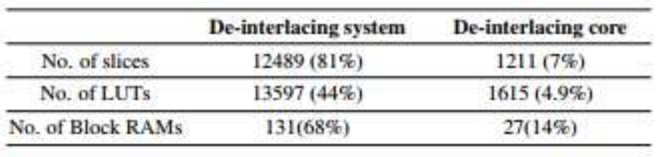
V. Conclusions

This paper presents an embedded system implemented on a FPGA that provides VGA 30 fps (frames per second) real-time video de-interlacing system. The video de-interlacing algorithm employs a hierarchical structure composed of three fuzzy logic-based systems. The algorithm is implemented as hardware IP core following a parallel rule processing architecture that increases the inference speed. Concerning the results for area occupation, an effective implementation is obtained since the proposed algorithm is optimized to use simple operators. A design methodology is provided to implement it on Xilinx FPGAs.

ACKNOWLEDGEMENTS

This work was partially supported by MOBY-DIC project FP7-INFSO-ICT-248858 (www.mobydic-project.eu) from European Community, TEC2011-24319 project from the Spanish Government, and P08-TIC-03674 project from the Andalusian Regional Government (with support from the PO FEDER). P. Brox is supported under the post-doctoral

Table 5. Implementation results of the embedded system on the Virtex-4 FPGA



	De-interlacing system	De-interlacing core
No. of slices	12489 (81%)	1211 (7%)
No. of LUTs	13597 (44%)	1615 (4.9%)
No. of Block RAMs	131(68%)	27(14%)

program called 'Juan de la Cierva' from the Spanish Ministry of Science and Innovation.

REFERENCES

[1] G. de Haan, E. B. Bellers, De-interlacing: an overview. Proc. of the IEEE: 1839-1857, 1998.

[2] http : //www.tridentmicro.com

[3] http : //www.st.com

[4] http : //www.siimage.com

[5] http : //www.zoran.com

[6] http : //www.micronas.com

[7] http : //www.nxp.com

[8] http : //www.mstarsemi.com

[9] http : //www.marvell.com

[10] Z. He, S. Natarajan, De-interlacing and YUV 4:2:2 to 4:2:0 conversion on TMS320DM6446 using the resizer. Application report from Texas Instruments, pp.1-17, Feb. 2008.

[11] C. Peng, Z. He, Y. Cager, An efficient motion-adaption de-interlacing technique on VLIW DSP architecture.IEEE Conference on Advanced Video and Signal Based Surveillance (AVSS), pp.644-649, Sep. 2005.

[12] E. B. Bellers, J. G. W. Janssen, R. J. Schutten, A software approach to high-quality video format conversion. Int. Conference on Consumer Electronics (ICCE), pp.441-442, Jan. 2005.