FPGA-based implementation of a fuzzy motion adaptive de-interlacing algorithm

P. Brox, S. Sánchez-Solano and I. Baturone

Instituto de Microelectrónica de Sevilla, IMSE-CNM (CSIC)
Edificio CICA, Av. Reina Mercedes s/n, 41012 Sevilla, SPAIN
Phone: +34955056666, Fax: +34955056686, E-mail: {brox|santiago|lumi}@imse.cnm.es

***Abstract*— This paper surveys the hardware implementation of a de-interlacing algorithm on Field-Programmable Technology for real-time processing. The algorithm presented evaluates the level of motion at each pixel, and determines the interpolation between a spatial and a temporal method according to the presence of motion. To achieve it the algorithm employs an hierarchical structure with three simple fuzzy systems. The first one performs a set of fuzzy rules to apply reasoning in order to detect motion; the second one selects the most convenient direction to implement an edge-dependent line average method; and the third one is used to choose the most adequate temporal method.**

The hardware implementation of this algorithm combines pipeline architecture with a parallel processing of fuzzy rules to accelerate the computation. As result an efficient implementation is developed in terms of computational time and hardware cost.

I. INTRODUCTION

Current FPGA devices include look-up tables, registers, multiplexers, and distributed and block memory, as well as specific circuitry for fast adders, multipliers, and I/O processing. This characteristic, together with a complete and unlimited reprogramming capability, have made FPGAs become key components in implementing high performance DSP systems in recent years, especially in the areas of digital communications, networking, video and imaging. Several tools have been developed in order to facilitate the design of FPGA-based DSP designs. The presented design flow utilizes one of these tools called System Generator, a system level tool developed by Xilinx (XSG)[1].

This paper describes the design and implementation of an algorithm for video de-interlacing. This type of algorithms are currently in demand by a wide number of devices, such as HDTVs, DVDs, projectors, etc., that require a progressive scanning format. Interlacing was introduced by TV industry as the most efficient method to reduce transmitted information. It consists of halving video bandwidth by eliminating lines, according to the order in which frames are sent. Frames with odd numbers only contain the odd lines of the image whereas frames with even numbers only contain the even lines. A complete frame containing odd and even lines can be calculated at the receiver using the interpolation techniques provided by de-interlacing methods [2].

The paper is organized as follows. A brief description of the algorithm is expounded in Section II. The strategy of algorithm implementation is explained in Section III. Implementation results are presented in Section IV. Finally, the main conclusions are outlined in Section V.

II. ALGORITHM STUDY

Among de-interlacing algorithms, motion adaptive algorithms offer a good trade-off between cost and quality [3]. This kind of algorithms combines a spatial method, I_S, and a temporal method, I_T, according to the presence of motion. They are based on the idea that temporal interpolation is very suitable for static areas, while spatial interpolation is more adequate when the level of motion is high.

The algorithm implemented herein uses fuzzy logic to interpolate between I_S and I_T de-interlacing methods. It uses as input system an evaluation of motion, which is calculated as the bi-dimensional convolution of a difference matrix (H) of luminance values from consecutive fields of the sequence:

$$motion = \frac{\Sigma[Cij][Hij]}{\Sigma[Cij]} = \frac{[2\ 4\ 2]\left[H_{11}\ H_{12}\ H_{13}\right]^T}{8} \qquad (1)$$

where $C_{(i,j)}$ are the convolution weights and $H_{(i,j)}$ are the following values (see Fig.1):

$$H_{11} = \frac{|B - B_0|}{2} \qquad H_{12} = \frac{|X_n - X_0|}{2} \qquad H_{13} = \frac{|B - B_0|}{2} \qquad (2)$$

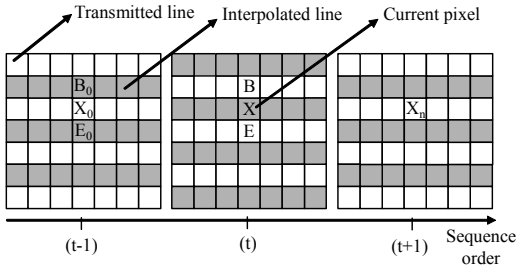


Fig. 1: Pixels involve in the calculation of *motion*.

TABLE 1. Rulebase for interpolation selection

	if	then
1)	*motion* is S	I_S
2)	*motion* is L	I_T
3)	*motion* is M	$\lambda I_T+\delta I_S$

Different sizes of matrices H and C have been studied to achieve a good trade-off between the resources required and the quality of motion measurement obtained [4]-[5]. The selected matrices, as can be seen in expression (1), only includes neighbors in vertical direction since a wide number of simulation sequences shown a non-decisive influence of horizontal neighbors.

Analyzing the values of matrix H in expression (2), it can be seen that is necessary the use of interpolated values calculated in the previous field (B_0, E_0 in Fig.1). To calculate the first progressive frame the spatial method I_S is applied.

The influence of motion in selecting the contribution of each interpolator is evaluated by considering the rules in Table 1. The fuzzy concepts small (S), large (L) and medium (M), used in the rulebase are modeled according to the membership functions shown in Fig.2. Using the Fuzzy Mean as defuzzification method, the new pixel value is calculated as follows:

$$X= \alpha_1 I_T + \alpha_2 I_S + \alpha_3(\lambda I_T + \delta I_S) \qquad (3)$$

where α_i is the activation degree of rule i. λ and δ are linear coefficients being its sum equal to one ($\lambda + \delta = 1$).

Our proposal introduces two main novelties over conventional motion adaptive methods. The first one is the use of fuzzy instead of crisp values to define different motion levels. This provides a more robust

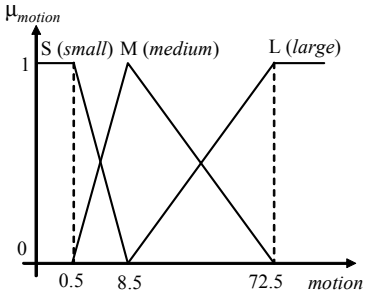
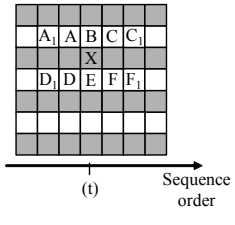


Fig. 2: Membership functions used in the rulebase for interpolation selection.

motion detection since threshold values usually produce wrong decision in areas where the decision is unclear. The second one is the inclusion of a third rule that increases the interpolation capability of the fuzzy system. Rulebases with up to five rules have been analyzed in [6]. Nevertheless, the base with three rules provides the most attractive solution in terms of hardware resources and quality of the interpolated image [6].

Moreover, the proposed algorithm also used two simple fuzzy systems to calculate the I_S and I_T interpolation modes. The proposal for the spatial interpolation performs an edge-adaptive interpolation by analyzing the five predetermined directions (a_1,a,b,c,c_1) shown in Fig.3(a) [7]. The rules of the Table in Fig.3(b) select the most adequate direction to apply the average of luminance values. The fuzzy concepts very large (VL), large (L), small (S) and very small (VS) used in the rulebase are defined by the membership functions shown in Fig.3(c). The final result I_S is given by:

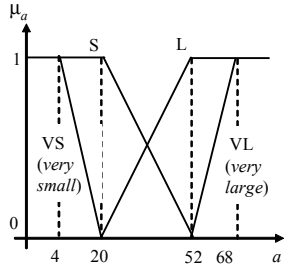
$$I_S= \beta_1\left(\frac{A+F}{2}\right)+\beta_2\left(\frac{C+D}{2}\right)+\beta_3\left(\frac{A+C+D+F}{4}\right)+ \\ +\beta_4\left(\frac{A_1+F_1}{2}\right)+\beta_5\left(\frac{C_1+D_1}{2}\right)+\beta_6\left(\frac{B+E}{2}\right) \qquad (4)$$



(a)

	if	then
1)	*a* is S *and b* is L *and c* is L	I_S=(A+F)/2
2)	*a* is L *and b* is L *and c* is S	I_S=(C+D)/2
3)	*a* is VS *and b* is L *and c* is VS	I_S=(A+F+C+D)/4
4)	a_1 is S *and a* is L *and b* is L *and c is* VL *and* c_1 is VL	I_S=(A_1+F_1)/2
5)	a_1 is VL *and a* is VL *and b* is L *and c is* L *and* c_1 is S	I_S=(C_1+D_1)/2
6)	*otherwise*	I_S=(B+E)/2

(b)



(c)

Fig. 3: (a) Pixels involve in the calculation of the spatial interpolator. (b) Rulebase to select the spatial interpolation according to the presence of edges. (c) Membership functions of the fuzzy concepts used in the rulebase.

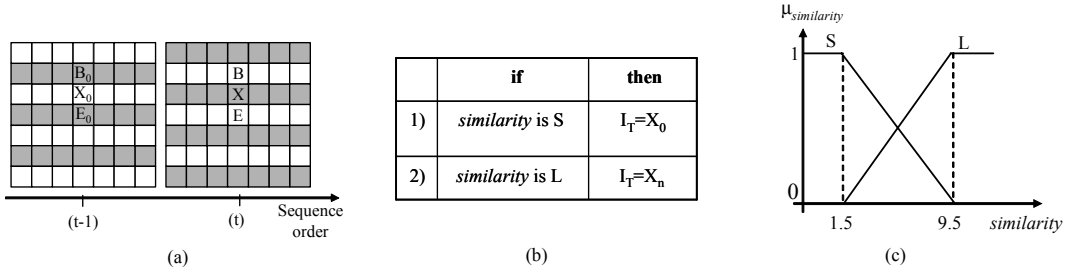


Fig. 4: (a) Pixels involve in the calculation of the temporal interpolator. (b) Rulebase to select the temporal interpolator (c) Membership functions of the fuzzy concepts used in the rulebase.

where β_i is the activation degree of the rules in the Table of Fig.3(b).

To select the best choice for temporal interpolation another fuzzy system is used. It makes a decision depending on the *similarity* between two consecutive fields, given by the following expression:

$$similarity(x, y, t) = \frac{|B - B_0| + |E - E_0|}{2} \quad (5)$$

The pixels used in expression (5) are shown in Fig.4(a). The rulebase takes a decision using a fuzzy transition to distinguish which pixel is the most adequate: the pixel in the previous X_0 or in the next field X_n (see Table of Fig.4(b)). The fuzzy definitions used in the rulebase are shown in Fig.4(c) and the result of I_T is calculated as follows:

$$I_T = \gamma_1 X_0 + \gamma_2 X_n \quad (6)$$

where γ_i is the activation degree of the rules in the Table of Fig.4(b).

III. ALGORITHM IMPLEMENTATION WITH *XSG*

Advances in VLSI technologies have encouraged a rapid growth in capacity and performance of FPGAs. On the other hand, the reconfiguration capability of FPGAs allows adapting its hardware resources for a specific processing system. This ability together with the development of powerful design tool such as XSG, which considerably reduces the overall system development time, have made FPGAs as one of the most attractive solution to develop rapid prototypes of digital signal processing (DSP) applications. The following subsections describe the implementation of the de-interlacing algorithm proposed in this paper.

A. Design specifications

The experimental set-up is configured using a XUP Virtex-II Pro Development board [8].It is an advanced hardware platform that contains a Virtex-II Pro FPGA surrounded by peripheral components that can be used to create a complex system. This board incorporates expansion connectors than can be used to connect a video capture board. This device acts as an interface between a video source such as camcorder, VCR, CCD camera, etc. and the board. The video decoder board is centered on the ADV7183B video decoder chip from Analog Devices, which can detect standard analog baseband television signals (NTSC, PAL and SECAM) and provides an output digital video signal. This conversion is realized according to the ITU-R BT 656 recommendation from the International Telecommunication Union (ITU), and is independent of the standard (NTSC, PAL or SECAM).

This recommendation describes an interface in which the code words that describe the video signal are transmitted in the form of eight bits at 13.5MHz. This forces the system to compute a new interpolated pixel value at 27MHz.

B. System design with XSG

Current FPGAs incorporate large amounts of block RAMs resources. Particularly, the design is developed on the Virtex-II Pro XC2VP30, which contains 136 block RAMs (BRAMs) and a total memory of 2,448 Kb [9]. Each block RAM built into the FPGA can be used with a configurable depth and width data. Due to the system requirements the design implements BRAMs with an 8-bit word width and a parametric memory depth (it is adaptive with the format of the video sequence).

To develop the fuzzy system that evaluates motion and selects the interpolation methods three field memories are required: a first one for the previous field (*t-1*), a second one for the current field (*t*), and a third one to store the calculated values of the previous field (see Fig.1(a)).

Block memories which implement field memories

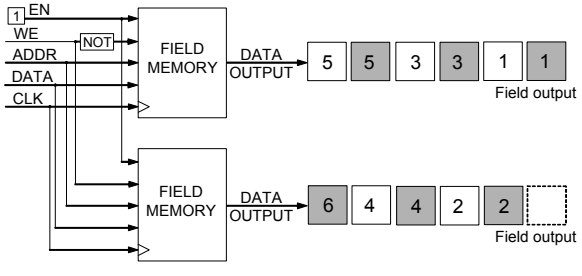


Fig. 5: Block diagram of the field memories. Each field memory provides alternately the previous (white box) or current (grey box) field.

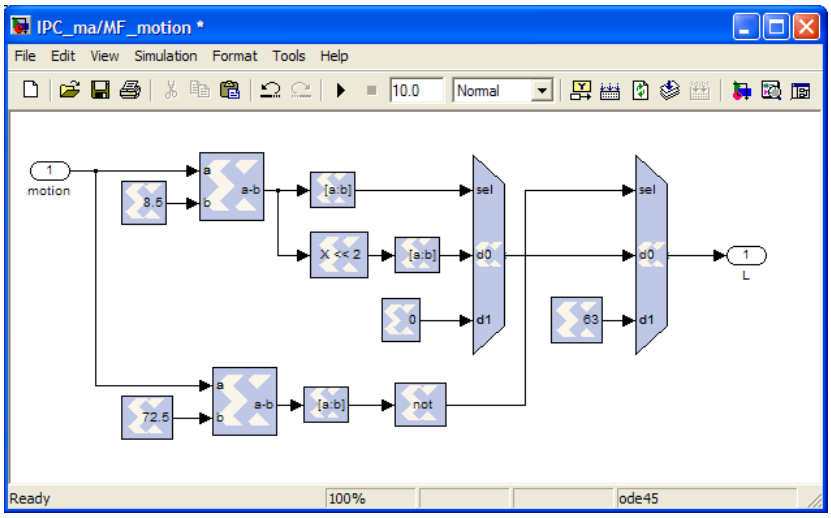


Fig. 6: XSG design to implement the fuzzy concept L.

(*t-1*) and (*t*) are configured into '*read after write mode*'. The implementation is performed enabling the write mode in one of the field memories, and disabling it in the other. Therefore, both modes are complementary and this causes that the current field continuously changes from one output field memory to the other as shown in Fig.5. A control signal is used to identify the current field.

The implementation of the field memory to store the interpolated pixels is realized following two different strategies. BRMAs and the use of a distributed memory, which employs slices of the FPGA. Both alternatives are presented in Section IV. For the two first de-interlaced fields, this field memory stores the values calculated by the I_S interpolator. For the rest of the fields, the previously calculated field is used.

XSG tool is integrated into the Simulink environment. It 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. Fig.6 shows the XSG design to implement L membership function (see Fig.2). The membership functions used in the rest of rulebases are implemented in a similar way. Note from the rules of the Table in Fig.3(b) that the antecedents are connected with *and* connectives. The minimum operator is selected for the implementation of these connectives.

Modern FPGA devices also incorporate embedded multiplier blocks [9]. The inputs of these embedded multiplier blocks can be up to 18 bits wide, and the output up to 36 bits. They are optimized for high-speed operations and have a lower power consumption compared to a multiplier implemented in slices. Besides, the use of the embedded multipliers leaves free slices in the FPGA that can be employed to implement other resources. Our design uses twelve of these multipliers to perform the expressions in (3), (4) and (6).

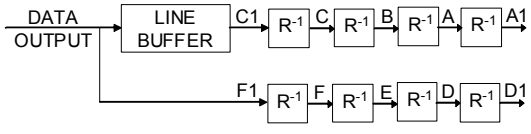


Fig. 7: Block diagram to obtain the ten pixel values shown in Fig.3(a).

The inputs of the block that implements the fuzzy system to calculate I_S, are taken from the output of the current field memory (*t*). Ten luminance values are necessary to compute the five inputs (a_1,*a*,*b*,*c*,c_1) of the system (see Fig.3(a)). A line buffer and eight registers are used to achieve the required luminance values as shown in Fig.7. XSG provides two ways to implement a line buffer, using delay blocks or specific Virtex-II line buffers [1]. The delay block is a shift register of configurable length. Data presented at the input will appear at the output after a user specified number of sample periods. The Virtex-II line buffer block delays a sequential stream of pixels by the specified buffer depth. It is optimized for the Virtex-II family since it uses the '*read before write*' option on the underlying Single Port RAM block. Both options have been used in the design implementation as shown in Section IV.

IV. IMPLEMENTATION RESULTS

The performance of the proposed algorithm has been analyzed by de-interlacing standard video sequences. The video sequences considered have widely been used as benchmarks in video processing applications. The interlaced video data have been obtained from these progressive sequences by eliminating lines. The peak signal-to-noise ratio, which is called PSNR, has been employed as figure of merit to compare the quality between the obtained interpolated frames and the original ones. It is defined as follows:

TABLE 2. PSNR values in (dBs) for different de-interlacing methods

Sequence Format	Missa CIF	Paris CIF	Trevor CIF	Salesman CIF	News QCIF	Mother QCIF	Carphone QCIF
Line Doubling	36.44	23.61	31.05	29.75	25.18	31.81	28.25
Line Average	40.47	26.67	35.04	33.53	29.25	35.94	32.61
ELA 3+3	39.49	25.53	34.11	32.11	26.63	35.39	32.65
ELA 5+5	38.56	24.64	33.31	30.17	25.92	34.2	31.51
Field Insertion	38.36	29.86	34.36	36.17	33.13	36.14	30.34
VT 2fields	40.25	30.73	36.61	36.54	35.46	39.61	34.08
VT 3 fields	40.52	31.37	37.16	36.95	35.67	40.89	34.54
Technique in [10]	40.01	33.12	35.38	37.62	34.73	39.49	32.27
Technique in [11]	40.18	35.28	36.69	38.29	37.51	41.87	34.78
Proposal	40.81	35.87	37.63	38.35	38.78	42.11	35.09

$$PSNR = 20\log\left(\frac{255}{\sqrt{MSE}}\right) \quad (7)$$

where *MSE* is the mean squared error between the original and the reconstructed image.

The proposed algorithm has also been compared with other de-interlacing algorithms with less or similar computational cost: four spatial method such as line doubling, line average, and conventional ELA using 3+3 and 5+5 taps; the simplest temporal de-interlacing algorithm called field insertion, and two vertico-temporal filtering with two and three fields [2]; and, finally, other fuzzy motion adaptive algorithms reported in [10] and [11]. Table 2 shows the average PSNR obtained when de-interlacing fifty fields of seven video sequences. As it can be seen, the proposed algorithm achieves the better results. All the algorithms presented in Table 2 have been coded in Matlab, and these results correspond to its execution using double-precision.

This section also contains implementation results in terms of device utilization, that is, hardware resources from FPGA used in the implementation, and also the maximum frequency achieved by the design to compute a new pixel value.

The XSG blocks which compose the design have been defined using parametric values, that is, their dimensions are non-fixed and are configured with variables from the input workspace. This provides a design with a high reconfigurability degree so as to work with different video sequence formats. For instance, Table 3 shows a summary of FPGA utilization using QCIF (176x144) and CIF (352x288) formats. The designs use Virtex-II line buffers blocks to implement line buffers and BRAMs to implement the three field memories. As it can be seen in Table 3, the processing of a higher format mainly implies a high increase of the number of BRAMs, whereas the rest of resources rise moderately. Obviously, the number of embedded multipliers used to implement the expressions (3), (4) and (6) are the same.

Table 4 shows the implementation results when the Virtex-II line buffer blocks are substituted for delay blocks. This reduces the number of BRAMs since each line buffer requires one BRAM at expense of a slight increase in the number of slices: 1.23% (QCIF) and 2.58% (CIF).

Finally, the field memory to store the interpolated pixel values is implemented using distributed memory instead of BRAMS. The results showed that this option is not efficient since it implies a large increase of the number of slices into the FPGA. The design for the QCIF format almost requires 90% of slices whereas there is not enough slices into the XC2VP30 FPGA to implement the algorithm for the CIF format. Finally, implementation results for the fuzzy systems to calculate the spatial and temporal interpolator are shown in Table 5.

The algorithm implementation can be evaluated from results obtained in the Simulink environment or by modeling the VHDL description generated by XSG

TABLE 3. Implementation results in terms of hardware resources for the complete proposed algorithm. The design uses Virtex-II line buffers and BRAMs to implement field memories

Format Sequence	Number of slices	Number of slices Flips Flops	Number of 4-input look-up tables (LUTs)	Number of BRAMs	Number of embedded multipliers
QCIF	1357 (9.91%)	1505 (5.49%)	1306 (4.76%)	35 (25.73%)	12 (8.82%)
CIF	1490 (10.87%)	1550 (5.65%)	1515 (5.53%)	92 (67.64%)	12 (8.82%)

TABLE 4. Implementation results in terms of hardware resources for the complete proposed algorithm. The design uses delays blocks to implement line buffers and BRAMs to implement field memories

Format Sequence	Number of slices	Number of slices Flips Flops	Number of 4-input look-up tables (LUTs)	Number of BRAMs	Number of embedded multipliers
QCIF	1526 (11.14%)	1855 (6.77%)	1294 (4.72%)	32 (23.52%)	12 (8.82%)
CIF	1843 (13.45%)	2271 (8.27%)	1506 (5.49%)	89 (65.44%)	12 (8.82%)

TABLE 5. Implementation results in terms of hardware resources for the fuzzy systems to calculate the spatial and temporal interpolator

Interpolator	Format Sequence	Number of slices	Number of slices Flips Flops	Number of 4-input LUTs	Number of embedded multipliers
Spatial	QCIF	579 (4.22%)	659 (2.41%)	639 (2.33%)	6 (4.41%
Spatial	CIF	721 (5.27%)	1073 (3.92%)	1016 (3.71%)	6 (4.41%)
Temporal	QCIF	72 (0.52%)	64 (0.23%)	58 (0.21%)	2 (1.47%)
Temporal	CIF	72 (0.52%)	64 (0.23%)	58 (0.21%)	2 (1.47%)

with the ModelSim tool from Mentor Graphics. The average PSNR value for 'Salesman' sequence is 37.71 dBs, whereas for the 'Mother' sequence is 40.35 dBs. As it can be seen from the results in Table 2, errors are higher for the hardware implementation because of the algorithm described in Matlab works with double-precision numbers (64-bits).

The design which implements the algorithm combines pipeline architecture with a parallel processing of fuzzy rules to accelerate the computation. As result, a new pixel values is interpolated each 9.61 ns (104.04 MHz). Therefore the design overcomes the timing constraints for real-time processing.

V. CONCLUSIONS

This paper presents the hardware implementation of a de-interlacing algorithm on a Virtex-II Pro FPGA. The algorithm uses three simple fuzzy systems to interpolate the non-transmitted lines of video signals. One fuzzy system is used to decide the contribution of spatial (I_S) and temporal (I_T) interpolators according to the presence of motion. Other fuzzy system is used to calculate the I_S interpolator, which is adaptive with the existence of edges in the image. Finally, the third fuzzy system calculates the most adequate I_T interpolator. The three fuzzy systems utilize simple fuzzy rulebases, which are implemented in parallel to accelerate the computation. The strategy of implementation employs pipeline architecture and provides a new interpolated pixel in a clock period. As a result, an efficient implementation of the algorithm in terms of processing time and hardware cost is achieved.

REFERENCES

[1] Xilinx Inc., "Xilinx System Generator for DSP (v9.1.01) User's Guide", March 2007. Web address to download: http://www.xilinx.com/support/sw_manuals/sysgen_ug.pdf.

[2] G. de Haan and E. B. Bellers, "De-interlacing: an overview," in *Proc. of the IEEE*, Sep. 1998, pp. 1839–57.

[3] A. M. Bock, "Motion adaptive standards conversion between formats similar field rates," *Signal Processing: Image Communication*, vol. 6, no. 3, pp. 275–80, Jan. 1994..

[4] P. Brox, I. Baturone, S. Sánchez-Solano, J. Gutiérrez-Ríos and F.Fernández-Hernández, "A fuzzy edge-dependent motion adaptive algorithm for de-interlacing,"*Fuzzy Sets and Systems. Special Issue: Image Processing*, vol. 158, no. 3, pp. 337–347, Feb. 2007.

[5] P. Brox, I. Baturone and S. Sánchez-Solano, "A fuzzy motion adaptive algorithm for interlaced-to-progressive conversion," in *Proc. of Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU)*, Jul, 2006.

[6] P. Brox, I. Baturone and S. Sánchez-Solano, "Fuzzy motion adaptive algorithm for video de-interlacing," in *Proc. of International Conference on Knowledge-Based and Intelligent Information and Engineering Systems (KES)*, Oct, 2006.

[7] P. Brox, I. Baturone and S. Sánchez-Solano, "A fuzzy edge-dependent interpolation algorithm," in *Soft Computing in Image Processing: Recent Advances. Heidelberg, Germany. Springer*, 2007.

[8] Xilinx Inc., "Xilinx University Program Virtex-II Pro Development System UG069 (v1.0)", March 2005. http://www.xilinx.com/univ/xup2vp.html

[9] Xilinx Inc., "Virtex-II Pro and Virtex-II Pro X FPGA User Guide UG012 (v4.1)", March 2007. Web address to download: http://www.xilinx.com/bvdocs/userguides/ug012.pdf

[10] D. Van de Ville, W. Philips and I. Lemahieu, "Fuzzy-based motion detection and its application to de-interlacing," in *Fuzzy techniques in image processing. Book Series of Studies in Fuzziness and Soft Computing*, 2000.

[11] J. Gutiérrez-Ríos, F. Fernández-Hernández, J. C. Crespo and G. Treviño, "Motion adaptive fuzzy video de-interlacing method based on convolution techniques," in *Proc. of Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU)*, Jul, 2004.