A Novel Macroblock-Level Filtering Upsampling Architecture for H.264/AVC Scalable Extension

T. L. da Silva[1], L. A. S. Cruz[2], and L. V. Agostini[3]

[1] Telecommunications Institute – University of Coimbra, Portugal
[2] Department of Electrical and Computer Engineering - University of Coimbra, Portugal
[3] Group of Architectures and Integrated Circuits – Federal University of Pelotas, Brazil
e-mail: thaisa.silva@co.it.pt

ABSTRACT

The scalable extension of the H.264/AVC standard, also called H.264/AVC SVC standard or simply SVC standard uses spatial upsampling in the spatial scalability modes. This work presents a novel upsampling architecture designed for operation at macroblock level and dyadic upsampling ratio with QVGA as the base layer resolution and VGA as the enhancement resolution. The adoption of a macroblock-level solution translates into a more efficient use of hardware resources with savings of approximately 25% in the number of ALUTs and DLRs and using about two hundred times less memory bits, when compared to previously published works. The designed architecture was synthesized targeting four FPGAs: Altera Cyclone III and Stratix IV and Xilinx Spartan 3E and Virtex 4. The best throughput was reached by the Xilinx Virtex 4 device, with a processing rate of 506 VGA frames per second. The worst result was reached by the Xilinx Spartan 3E FPGA, with 249 VGA frames per second. All target FPGAs surpasses the necessary throughput to decode VGA videos in real time. This very high throughput is important especially when low power applications are considered, since with low operation frequencies (9.34MHz) it is possible to reach real time (30 frames per second) for all target FPGAs.

Index Terms: video coding, spatial scalability, H.264/AVC SVC, upsampling, architectural design.

1. INTRODUCTION

The growth in the number of devices that handle digital video and the diversity of features of these devices stimulated the development of an extension to the H.264/AVC standard [1], called H.264/AVC Scalable Video Coding or SVC [2] as a solution to the problem of delivering video to portable energy-limited devices. Transmission of a single stream of high resolution and high quality video is not a solution as it would have unwanted impacts like higher reception bandwidth and larger power consumption without a visible profit given the usually low resolution displays used in these devices. The alternate solution of using multicast has its problems too, since it is necessary to code and transmit or store different resolution versions of the same content.

A H.264/AVC SVC compliant coder is able to encode a video signal generating a representation which contains the video information encoded with different spatial, temporal and quality resolutions or levels with the data of those scalability combinations embedded in a single bitstream. From this bitstream, subset bitstreams, containing distinct resolutions, frame rates, and/or bit rates can be extracted without the need to re-encode or transcode the video signal.

The H.264/AVC SVC standard provides support for spatial scalability, which is based on the concept of encoding separate different resolution layers in a hierarchical fashion, starting with the base layer coder, which is compliant with the H.264/AVC coder [1], and going up in resolution with the addition of one or more enhancement spatial layers and respective coders.

The spatial scalability coding scheme defined in H.264/AVC SVC uses interlayer prediction mechanisms, which allow the exploration of correlations between different layers and the data already encoded/decoded in the base layer can be reused in the encoding/decoding of the enhancement layers [3]. There are three interlayer prediction mechanisms: motion prediction, residual prediction and intra prediction.

The interlayer motion prediction is used to reduce interlayer redundant motion data, such as macroblock partitioning, reference frame index and

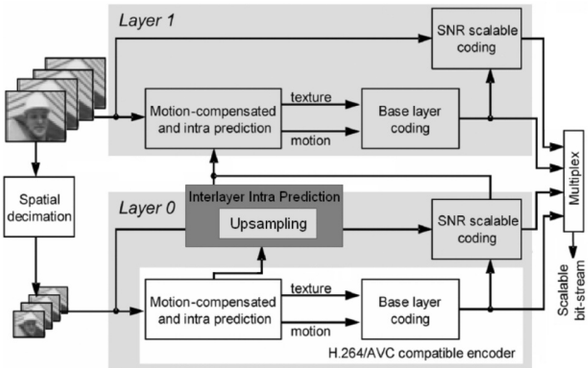

Figure 1. H.264/AVC Scalable Video Coding with two layers.

motion vectors. The interlayer residual prediction is used to reduce the amplitude of the residues after the inter-frames prediction [4]. In interlayer intra prediction, the signal prediction for a block in the enhancement layer is generated upsampling the reconstructed intra signal from the co-located block in the base layer [5]. This prediction can occur only when the co-located block in the base layer has been encoded with intraframe prediction [3].

This paper focuses on interlayer intra prediction, more precisely in the hardware implementation of the spatial upsampling operations on which it depends. Figure 1 shows a typical coder structure with two spatial layers. The module responsible for the spatial upsampling process is the focus of this work and it is highlighted in Figure 1.

The architecture presented in this work was designed specifically for the dyadic case, in which the horizontal and vertical resolutions are doubled between spatially adjacent layers. Our work supports QVGA resolution (320x240 pixels) at the base layer and VGA resolution (640x480 pixels) at the enhancement layer.

Both layers used the YCbCr color space with 4:2:0 color sub-sampling arrangement [6].

This paper is structured as follows: in Section 2 we outline the operation of the upsampler. In the Section 3 the novel architecture for the upsampler is presented. Section 4 explains the validation methodology. Section 5 reports the synthesis results and comparisons with competing works. Finally, Section 6 presents the conclusions of this work and outlines future developments.

2. H.264/AVC SVC UPSAMPLING METHOD

The upsampler is responsible for increasing the spatial resolution of a block in the base layer to the resolution of the enhancement layer. It is applied when the prediction mode of a block is interlayer and the corresponding block in the base layer has been

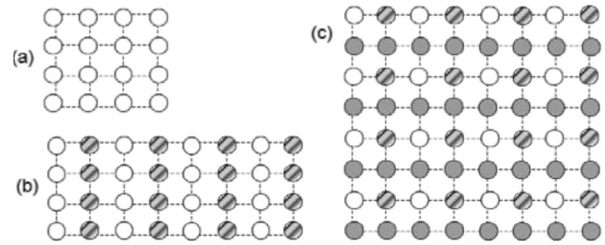

Figure 2. (a) Original Image, (b) Horizontal Filtering and (c) Vertical Filtering.

encoded using intra prediction. The upsampling operation is also used in the interlayer residual prediction [3].

To implement the upsampling operation a 4-tap multiphase filter is applied to the luminance component and a bilinear filter is applied to the chrominance components. These two dimensions of the input block are handled using the filter separability property and then, two instances of a 1-D filter are applied, first in the horizontal direction of the block and afterwards in the vertical direction. The use of different filters for luma and chroma is motivated by complexity issues. The upsampling module includes a set of 16 filters, where the filter to be used is selected according to the upsampling scale factor and the spatial position of the sample to be interpolated. For the dyadic case, the filters with indexes 4 and 12 are used alternately [7].

Figure 2 shows the filtering steps involved in the upsampling process. In Figure 2 (a) the white circles represent the samples of a video frame of the base layer. Figure 2 (b) presents the result of horizontal filtering. The frame resulting from the horizontal upsampling is vertically interpolated as show in Figure 2 (c). The resulting frame has twice the width and twice the height of the original frame, as expected.

3. UPSAMPLER ARCHITECTURE

The complete designed architecture for the upsampling method is presented in Figure 3. Unlike the solution presented in [8] which was also designed

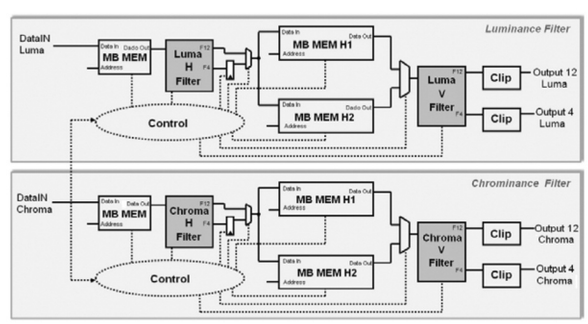

Figure 3. Complete upsampling architecture.

in our group and processed an entire frame at once, the architecture proposed in this paper operates at macroblock-level. This processing mode has the advantage that, to process each macroblock, we only have to store the macroblock to be filtered and two rows and two columns of the neighboring macroblocks, instead of storing the whole frame.

The macroblock-level processing causes important gains in the number of needed memory elements. This modification implied in changes to the design presented in [8], particularly on the structures of the luminance and chrominance modules, with respect to the registers, memories and the control algorithm.

One important characteristic of the H.264/AVC SVC coding algorithm is the independent processing (at some stages) of the luminance and chrominance components which allow the absences of data dependencies in the luminance and chrominance datapaths.

This independence was explored in the architecture designed in this paper, as presented in Figure 3, where two parallel datapaths were designed allowing very high processing rates.

The architecture is composed of two luminance filters: Luma H Filter (horizontal filtering) and Luma V Filter (vertical filtering) and two chrominance filters: Chroma H Filter (horizontal filtering) and Chroma V Filter (vertical filtering).

The macroblock data to be filtered is stored in MB MEM memories in Figure 3, which feeds the luminance and chrominance horizontal filters. The horizontal filtering results are stored in MB MEM H1 and MB MEM H2 memories in booth datapaths. These two last memories are used as ping-pong transposition buffers between the horizontal and vertical filters. The ping-pong buffers are important to allow high throughput, since when one memory is used to write the H filters results the second memory is used to read the inputs to the V filters. In this case there is not wasted time in the filters caused by the memory access restrictions and the filters are able to operate in their maximum processing rates.

Figure 3 also shows the clipping operators (Clip) and the control. The first is used to adjust the outputs to the same dynamic range of the inputs and the second is responsible to the correct activation of the hardware modules. The control is represented by the dotted lines. The luminance and chrominance control signals were independently designed and a few synchronism signals were used to indicate the processing start and end points for each kind of sample. The memory address registers and their respective multiplexers were omitted in Figure 3 for simplification, as well as their control signals.

Figure 4 and Figure 5 present the luminance filter and chrominance filter core architectures, respectively.

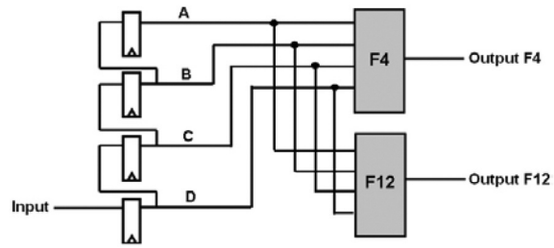

Figure 4. Luminance filter core architecture.

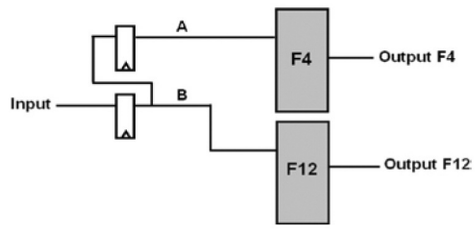

Figure 5. Chrominance filter core architecture

These architectures were designed to implement the luminance and chrominance filters with indices 4 and 12, and then the input data are shared between filters F4 and F12.

In Figure 4, the filter uses a four-tap shift register, where each position stores a sample used as input to both filters (F4 and F12). When the four registers are filled with valid data, the calculations of the filters are started. The chrominance filter core architecture (Figure 5) is similar and adopts the same strategy to leverage the data input between filters 4 and 12.

The luminance filters (F4 and F12, in Figure 4) were designed based on equations (1) and (2) and the chrominance filters (F4 and F12, in Figure 5) were designed according to equations (3) and (4). These four equations were generated from the original ones available at the standard specification [2]. With simple algebraic manipulations was possible to design multiplier free filter cores, since all multiplications were broken in shift-adds. These equations produce exactly the same results that the original ones defined in the standard [2].

$$S4 = (-2.A - A + 16.B + 8.B + 4.B + 8.C - D) >> 5 \quad (1)$$

$$S12 = (-A + 8.B + 16.C + 8.C + 4.C - 2.D - D) >> 5 \quad (2)$$

$$S4 = (16.A + 8.A + 8.B) >> 5 \quad (3)$$

$$S12 = (8.A + 16.B + 8.B) >> 5 \quad (4)$$

Figure 6 and Figure 7 present the architectures that implement the luminance and chrominance filters with index 4, i.e., the equations 1 and 3, respectively.

Figure 8 and Figure 9 show the architectures designed for the luminance and chrominance filters with index 12, related to equations 2 and 4, respectively.

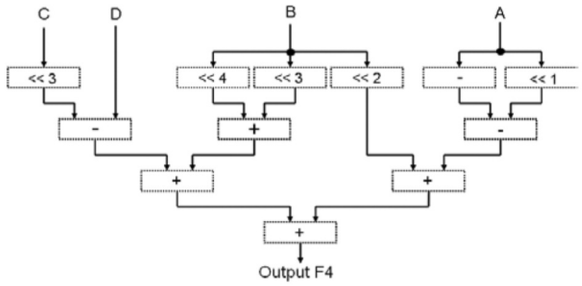

Figure 6. Internal architecture of the luminance filter 4.

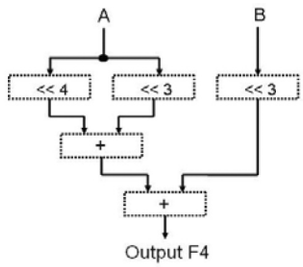

Figure 7. Internal architecture of the chrominance filter 4.

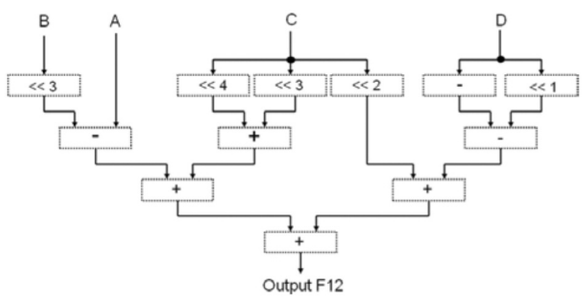

Figure 8. Internal architecture of the luminance filter 12.

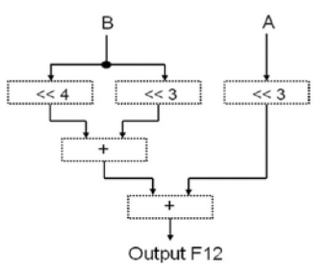

Figure 9. Internal architecture of the chrominance filter 12.

4. VALIDATION METHODOLOGY

The first step of the architecture validation was the insertion of some functions in the H.264/AVC SVC reference software [9], aiming to extract the input and output values from each module that compose the upsampling architecture. Afterwards, to verify the modules implementation correctness, some simple simulations using the Xilinx ISE [10] and Altera Quartus II [11] tools were performed.

Then, after the integration of all architectural modules, more complete simulations were done through Mentor Graphics ModelSim tool [12].

Test benches were written to run these simulations. These test benches use the data captured from the reference software as input stimuli for the architectures under validation. The test benches also store the outputs of the designed architectures in text files.

Thus, through the comparison between the outputs obtained with the test benches and outputs extracted from the reference software, was possible to verify the correction of the results generated by the architectures. After some minor modifications, the designed architectures were considered validated.

5. SYNTHESIS RESULTS AND COMPARISONS

The upsampling architecture was described in VHDL and synthesized for Altera [11] and Xilinx [10] FPGAs devices. The synthesis targeted high performance and more expensive FPGAs, like Altera Stratix IV and Xilinx Virtex 4, and slower and cheaper FPGAs, like Altera Cyclone III and Xilinx Spartan 3E.

Table I presents the synthesis results of the architecture designed for Altera Cyclone III and Stratix IV FPGAs. Table I shows the synthesis results in terms of combinational ALUTs and dedicated registers used by each module and by the complete upsampling architecture. Furthermore, the timing analysis results indicate that the complete upsampling architecture synthesized for Altera Cyclone III FPGA is able to work at a frequency of 108.51 MHz, which is enough to process up to 348 VGA fps and when synthesized for Altera Stratix IV FPGA this architecture reaches a frequency of 145.54 MHz, processing up to 465 VGA fps. The throughput of both devices outperforms the real-time requirements. Table II presents the synthesis results for Xilinx Spartan 3E and Virtex 4 FPGAs. Through the obtained results it was possible to verify that the complete upsampling architecture is able to process 348 VGA frames per second when targeted to Spartan 3E FPGA and 465 VGA frames per second when targeted to Virtex 4 FPGA.

Considering the number of cycles which are necessary to process one frame by the designed architecture, it is possible to calculate the minimum operation frequency that is necessary to reach a processing rate of 30 frames per second. Since the number of cycles is an architectural feature, this metric does not depends on the target FPGA, then, the minimum operation frequency to reach 30 FPS is common for all FPGAs. This operation frequency is of only 9.34 MHz and this means that with this low operation frequency the architecture is able to reach real time for VGA frames.

Table III presents a processing rate comparison among the four target FPGAs. This table also presents throughput estimates for higher resolutions considering the four target FPGAs. These resolutions for the

Table I. Altera Synthesis Results.

	Cyclone III			Stratix IV		
	Luma	Chroma	Complete	Luma	Chroma	Complete
Frequency (MHz)	113.47	173.67	108.51	146.67	290.02	145.54
ALUTs	1,216	810	1,989	930	606	1,554
Dedicated Logic Registers	429	342	767	429	341	766
Memory Bits	27,200	4,800	32,000	27,200	4,800	32,000
FPS @ VGA	364	1,121	348	469	1,852	465

Devices: Cyclone III EP3C16F484C6 / Stratix IV EP45SGX530HH35C3

Table II. Xilinx Synthesis Results.

	Spartan 3E			Virtex 4		
	Luma	Chroma	Complete	Luma	Chroma	Complete
Frequency (MHz)	77.53	131.16	77.53	157.54	226.80	157.54
LUTs	1,129	834	1,894	1,149	882	1,939
Registers	411	345	748	411	345	748
BRAMs	3	3	6	3	3	6
FPS @ VGA	249	846	249	506	1,464	506

Devices: Spartan 3E XC3S1600E / Virtex 4 XC4VLX15

Table II. Processing rates comparison.

	Spartan 3E	Cyclone III	Stratix IV	Virtex 4
FPS @ VGA	249	348	465	506
FPS @ 4VGA	*62	*87	*117	*127
FPS @ 1080p	*37	*52	*69	*75
FPS @ 16VGA	*15	*22	*29	*31

*estimate

Table IV. Comparison between upsampling architectures.

	Macroblock-Level Filtering	Frame-Level Filtering [8]
Frequency (MHz)	145.54	127.42
ALUTs	1,554	2,024
DLRs	766	1,032
Memory Bits	32,000	6,451,200
FPS @ VGA	465	384

Device: Stratix IV EP45SGX530HH35C3

enhancement layer are 4VGA (1280x960 pixels), 16VGA (2560x1920 pixels) and 1080p (1920x1080). It is important to highlight that these estimates consider the dyadic case.

The first line presents the real results, for VGA resolution. The other lines present the estimative for higher resolutions. The lowest frame rates were reached by the Xilinx Spartan 3E FPGA, followed by the Altera Cyclone III and Stratix IV FPGAs and the highest frame rate is reached by the Xilinx Virtex 4 FPGA.

The frame rate for all target FPGAs are enough to reach real time when VGA, 4VGA and 1080p resolutions are considered. But when 16VGA is considered, only the Virtex 4 FPGA reaches a frame rate higher than 30fps.

The architecture designed in this paper was synthesized for the same Altera Stratix IV used in [8], aiming to establish some comparisons between the frame-based solution presented in [8] and the macroblock-based solution implemented in this work.

Table IV compares the hardware resources use and performance indicators of the macroblock-level filtering architecture designed in this paper and those presented in [8], based on frame-level filtering. From this table it is possible to verify that the current architecture requires 25% less ALUTs and DLRs (Dedicated Logic Registers) and uses about two hundred times less memory bits than the frame-level filtering solution. This large difference occurs because the architecture presented in [8] needs to store an entire frame to perform the upsampling filtering while the architecture presented in this paper performs the filtering at a macroblock level, storing only a macroblock and two rows and columns around it to perform each filtering.

Concerning throughput, the architecture designed in this work is able to process 465 VGA frames per second while the previous solution does not go beyond 384 VGA frames per second.

The comparison with related works is limited as, to the best of our knowledge, there are not any other papers published in the relevant literature about architectural design targeting the Upsampling module of the Scalable Extension of the H.264/AVC. However we found some works [13][14][15] about the Motion Compensation Interpolation of the H.264/AVC standard that, despite slight differences, can be compared to the present work.

In the H.264/AVC standard, quarter-pixel interpolation of the Motion Compensator is achieved using a 6-tap horizontal and vertical FIR filter for luminance components and a bi-linear filter for chrominance components. However, the architecture presented in [13] proposes the use of a 4-tap diagonal FIR filter for interpolation luminance instead of the 2-D 6-taps filters generating losses in quality. A three-stage recursive algorithm is used in the interpolation of chrominance samples, reducing the number of multiplications.

In [14] a motion compensation architecture for multiple standards is presented and it works at 148.5 MHz and can achieve the real-time multiple-standard decoding for HDTV 1080i (1920x1088 4:2:0 60field/s) video.

The luminance interpolator architecture proposed in [15] works at an operation frequency of 87 MHz, but it reduces the hardware costs when compared to [14]. The main problem in all these cases is that those works are focused in standard cells technology when our work is targeted to FPGAs, making difficult a direct comparison of the use of hardware resources and throughput since the target technology is different.

6. CONCLUSIONS AND FUTURE WORK

This work presented an Interlayer Upsampling architecture compliant with the H.264/AVC Scalable Video Coding extension, designed to support a dyadic upsampling ratio from QVGA (base layer resolution) to VGA (enhancement layer resolution) which operates at a macroblock-level.

The architecture was described in VHDL, synthesized and validated. The synthesis targeted Altera Cyclone III and Stratix IV FPGAs and Xilinx Spartan 3E and Virtex 4 FPGAs.

From the results obtained it was possible to verify that all FPGA devices surpasses a lot the minimum requirements to decode the VGA (640x480 pixels) enhancement layer resolution in real time. The best throughput was reached by the Xilinx Virtex 4, with a processing rate of 506 VGA frames per second. The worst result was reached by the Xilinx Spartan 3E FPGA, with 249 VGA frames per second.

This high throughput is function of the efficient technology used in the FPGAs and it is also function of the low number of cycles that the designed architecture needs to process each VGA frame. With only 9.34 MHz the architecture is able to process VGA frames at 30 frames per second. This means that this architecture is a good solution for low power applications, since even with a low operation frequency, it is able to reach real time when processing VGA videos.

When compared to a previously designed work, the architecture presented in this work obtained the best results in terms of frame rate, frequency and use of hardware resources. This good result is function of the novel macroblock level processing adopted in this work.

In addition to the synthesis for the VGA resolution at the enhancement layer, estimates were made for higher resolutions, such as 4VGA (1280x960 pixels), 1080p (1920x1080 pixels) and 16VGA (2560x1920 pixels) resolutions at the enhancement layer, considering the use of two spatial layers and the dyadic case. These estimates shown that, for the best case (Virtex 4 FPGA), the upsampling architecture designed in this work would be able to process 127 4VGA frames, 75 1080p, frames and 31 16VGA frames per second, satisfying for all evaluated resolutions, the necessary throughput to process videos in real time.

Future steps will include the prototyping of the upsampling architecture with integration of a Deblocking Filter module designed in our group, to generate a complete Interlayer Intra Prediction module following the H.264/AVC SVC specifications. The development of an upsampling processing structure supporting more than two spatial layers is also planned.

ACKNOWLEDGEMENTS

The authors would like to acknowledge to the FCT Portuguese agency and CAPES and CNPq Brazilian agencies for the financial support of this work.

REFERENCES

[1] Joint Video Team of ITU-T, and ISO/IEC JTC 1: Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264 or ISO/IEC 14496-10 AVC), 2003.

[2] International Telecommunication Union. ITU-T Recommendation H.264 (11/07): advanced video coding for generic audiovisual services. [S.l.], 2007.

[3] H. Schwarz, D. Marpe and T. Wiegand, "Overview of the Scalable Video Coding Extension of the H.264/AVC Standard", *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 17, No. 9, September, 2007, 1103-1120.

[4] M. Wien, H. Schwarz, and T. Oelbaum, "Performance Analysis of SVC", *IEEE Transactions on Circuits and Systems for Video Technology*, Vol.17, No.9, September, 2007, 1194–1203.

[5] H-S. Huang, W-H. Peng and T. Chiang, "Advances in the Scalable Amendment of H.264/AVC," *IEEE Communications Magazine*, Vol. 45, No.1, January, 2007, 68-76.

[6] M. Ghanbari, *Standard Codecs: Image Compression to Advanced Video Coding*. United Kingdom: The Institution of Electrical Engineers, 2003.

[7] C. Segall and G. Sullivan, "Spatial Scalability within the H.264/AVC Scalable Video Coding," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 17, No. 9, September, 2007, 1121-1135.

[8] T. L. da Silva, F. Rediess, L. V. Agostini, A. A. Susin and S. Bampi, "Efficient Hardware Design for the Upsampling in the H.264/SVC Scalable Video Coding Extension", in *Proceedings of the 17th IFIP International Conference on Very Large Scale Integration*, October, 2009, pp. 1-6.

[9] JSVM (Joint Scalable Video Model). "SVC Reference Software", Available in: http://ip.hhi.de/imagecom_G1/savce/downloads/SVC-Reference-Software.htm, 2009.

[10] Xilinx Inc. "Xilinx: The Programmable Logic Company", Available in: http://www.xilinx.com, 2009.

[11] Altera Corporation, "Altera: The Programmable Solutions Company", Available in: http://www.altera.com/, 2009.

[12] MENTOR GRAPHICS. ModelSim, Available in: http://www.model.com, 2009.

[13] W.-N. Lie, H.-C. Yeh, T. C.-I. Lin and C.-F. Chen, "Hardware efficient computing architecture for motion compensation interpolation in h.264 video coding", in *Proceedings of IEEE International Symposium Symposium on Circuits and Systems*, May, 2005, 2136-2139.

[14] J. Zheng, W. Gao, D. Wu and D. Xie, "A novel VLSI architecture of motion compensation for multiple standards," *IEEE Transactions on Consumer Electronics*, Vol. 54, No. 2, May, 2008, 687 – 694.

[15] J. Chen, C. Lin, J. Guo and J. Wang, "Low Complexity Architecture Design of H.264 Predictive Pixel Compensator for HDTV Applications", in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, May, 2006, 932 - 935.