PROCEEDINGS OF SPIE

VLSI Circuits and Systems V

Teresa Riesgo
Eduardo de la Torre-Arnanz
Editors

18–20 April 2011
Prague, Czech Republic

SPIE
Microtechnologies

Volume 8067

SPIE

PROCEEDINGS OF SPIE

VLSI Circuits and Systems V

Teresa Riesgo
Eduardo de la Torre-Arnanz
Editors

18–20 April 2011
Prague, Czech Republic

Sponsored and Published by
SPIE

Volume 8067

Proceedings of SPIE, 0277-786X, v. 8067

Please use the following format to cite material from this book:
Author(s), "Title of Paper," in *VLSI Circuits and Systems V*, edited by Teresa Riesgo, Eduardo de la Torre-Arnanz, Proceedings of SPIE Vol. 8067 (SPIE, Bellingham, WA, 2011) Article CID Number.

ISSN 0277-786X
ISBN 9780819486561

Published by
SPIE
P.O. Box 10, Bellingham, Washington 98227-0010 USA
Telephone +1 360 676 3290 (Pacific Time) · Fax +1 360 647 1445
SPIE.org

Paper Numbering: Proceedings of SPIE follow an e-First publication model, with papers published first online and then in print and on CD-ROM. Papers are published as they are submitted and meet publication criteria. A unique, consistent, permanent citation identifier (CID) number is assigned to each article at the time of the first publication. Utilization of CIDs allows articles to be fully citable as soon they are published online, and connects the same identifier to all online, print, and electronic versions of the publication. SPIE uses a six-digit CID article numbering system in which:

- The first four digits correspond to the SPIE volume number.
- The last two digits indicate publication order within the volume using a Base 36 numbering system employing both numerals and letters. These two-number sets start with 00, 01, 02, 03, 04, 05, 06, 07, 08, 09, 0A, 0B … 0Z, followed by 10-1Z, 20-2Z, etc.

The CID number appears on each page of the manuscript. The complete citation is used on the first page, and an abbreviated version on subsequent pages. Numbers in the index correspond to the last two digits of the six-digit CID number.

Contents

Conference Committee

Symposium Chair

Thomas Becker, EADS Innovation Works (Germany)

Symposium Cochairs

José Feliciano López, Universidad de Las Palmas de Gran Canaria (Spain)
Gerhard Krötz, Hochschule Kempten (Germany)

Symposium Local Chair

Cestmír Ondrůšek, Technical University of Brno (Czech Republic)

Conference Chair

Teresa Riesgo, Universidad Politécnica de Madrid (Spain)

Conference Cochairs

Eduardo de la Torre-Arnanz, Universidad Politécnica de Madrid (Spain)
Peter Langendörfer, IHP GmbH (Germany)

Program Committee

Said Fares Al-Sarawi, The University of Adelaide (Australia)
Mladen Berekovic, TUBS (Germany)
Sebastián A. Bota, Universitat de les Illes Balears (Spain)
João Canas Ferreira, Universidade do Porto (Portugal)
Massimo Conti, Università Politecnica delle Marche (Italy)
Angel de Castro, Universidad Autónoma de Madrid (Spain)
Francisco V. Fernández, Instituto de Microelectrónica de Sevilla (Spain)
Eby G. Friedman, University of Rochester (United States)
Eckhard Grass, IHP GmbH (Germany)
Milos Krstic, IHP GmbH (Germany)
Sebastián López Suárez, Universidad de Las Palmas de Gran Canaria (Spain)
Celia López-Ongil, Universidad Carlos III de Madrid (Spain)
Antonio López-Martín, Universidad Pública de Navarra (Spain)
Gustavo Marrero Callicó, Universidad de Las Palmas de Gran Canaria (Spain)
Salvador Mir, TIMA Laboratoire (France)
Pere Lluis Miribel-Català, Universitat de Barcelona (Spain)

Félix Moreno, Universidad Politécnica de Madrid (Spain)
Antonio Núñez Ordóñez, Universidad de Las Palmas de Gran Canaria (Spain)
Ioannis Papaefstathiou, Technical University of Crete (Greece)
Belén Pérez-Verdú, Centro Nacional de Microelectrónica (Spain)
Jorge Portilla, Universidad Politécnica de Madrid (Spain)
Juan José Rodríguez Andina, Universidad de Vigo (Spain)
Ángel B. Rodríguez-Vázquez, Centro Nacional de Microelectrónica (Spain)
Marco Domenico Samtambroglio, Politecnico di Milano (Italy)
César Sanz, Universidad Politécnica de Madrid (Spain)
Gilles Sassatelli, Laboratoire d'Informatique de Robotique et de Microelectronique de Montpellier (France)
José Silva Matos, Universidade do Porto (Portugal)
Dirk Stroobandt, Universiteit Gent (Belgium)
Walter Stechele, Technische Universität München (Germany)

Session Chairs

1 Bio-inspired and Reconfigurable Systems
Eduardo de la Torre-Arnanz, Universidad Politécnica de Madrid (Spain)

2 Wireless Communication Systems
Massimo Conti, Università Politecnica delle Marche (Italy)

3 Off-chip & On-chip Communications
Lambert Spaanenburg, Lund University (Sweden)

4 Multimedia and High Performance Architectures
Lukáš Sekanina, Brno University of Technology (Czech Republic)

5 IC Design
Eduardo Juarez, Universidad Politécnica de Madrid (Spain)

6 Measuring, Detecting and Obscuring Defects and Effects
Torsten Schmidt, Technische Universität Dresden (Germany)
José Feliciano López, Universidad de Las Palmas de Gran Canaria (Spain)

Performance Analysis of the Scalable Video Coding (SVC) extension of H.264/AVC for constrained scenarios

N. Suarez*[a], Gustavo M. Callico[a], Sebastian Lopez[a], Jose Lopez[a], Roberto Sarmiento[a]
[a]Institute for Applied Microelectronics and Department of Integrated Systems,
University of Las Palmas de Gran Canaria, E–35017,
Las Palmas de Gran Canaria, SPAIN

ABSTRACT

Scalable Video Coding (SVC) is the extension of H.264/AVC standard proposed by Joint Video Team (JVT) to provide flexibility and adaptability on video transmission. SVC is an extension of the H.264/AVC codec that exploits the use of layers, what permits to obtain a bit stream where specific parts can be removed to obtain an output video with a lower resolution (temporal or spatial) and/or lower quality/fidelity.

This paper provides a performance analysis of the scalable video coding (SVC) extension of H.264/AVC for constrained scenarios. For this, the open-source decoder called "Open SVC Decoder" was adapted to obtain a version likely to be implemented in reconfigurable architectures. For each scenario a set of different sequences were decoded to analyze the performance of each functional block inside the decoder.

From this analysis we conclude that reconfigurable architectures are a suitable solution for an SVC decoder in a constrained device or for a specific range of scalability levels. Our proposal consists in architecture of a SVC decoder that admits different options depending on device requirements where certain blocks are customizable to improve the performance of decoder in hardware resources usage and execution time.

Keywords: SVC, H.264/AVC, video scalability, constrained devices, reconfigurable architectures

1. INTRODUCTION

Nowadays, video coding standards are used for a widely range of applications and, in particular, video streaming has quickly increased her usage on the Internet. Video transmission over packet networks is exposed to variable transmission conditions. Moreover, to the complexity of video streaming over Internet it is also added the amount of heterogeneous devices with different requirements in spatial and temporal resolution. To support these heterogeneous environments [1-4] with different client capabilities and transmission channel capabilities, as shown in Figure 1, is necessary provide scalable profiles in video coding standards.

A network device, for instance, can extract from this global bit-stream a sub-stream that is suitable for a mobile handset with constrained capabilities, like screen size, hardware processing or battery. This is an example of how the scalable video coding can be used to adapt the bit stream to its application demands, network condition and/or processing capability, and obtain a representation with a lower quality by partial decoding.

The Scalable Video Coding (SVC) is an Annex of the H.264/AVC standard [5-6] which extends the original standard with new tools designed to efficiently support temporal, spatial and quality (SNR) scalability developed by the Joint Video Team (JVT). The JVT is a group of experts from ITU-T Video Coding Experts Group (VCEG) and ISO/IEC Moving Pictures Experts Group (MPEG) created to develop the H.264/AVC standard.

This paper evaluates the performance of SVC extension of H.264/AVC for constrained scenarios and is organized as follows. Section II explains the Scalable Video Coding standard and the fundamental scalability types. Section III describes the Open SVC decoder and conditions for evaluating the SVC decoder. Section IV analyses the results obtained and the conditions to consider in reconfigurable architectures. Finally, the conclusions are summarized in Section V.

*nsuarez@iuma.ulpgc.es; phone +34 928 457 337; fax 1 +34 928 451 083; www.iuma.ulpgc.es

VLSI Circuits and Systems V, edited by Teresa Riesgo, Eduardo de la Torre-Arnanz,
Proc. of SPIE Vol. 8067, 80670J · © 2011 SPIE · CCC code: 0277-786X/11/$18 · doi: 10.1117/12.888269

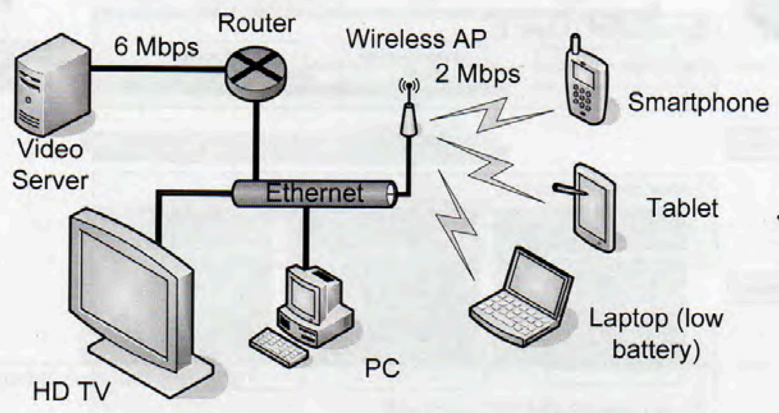

Figure 1. Example of broadcasting cases over heterogeneous network.

2. SCALABLE VIDEO CODING STANDARD

2.1 H.264/SVC standard

In video coding, the term scalability means that parts of a coded video can be extracted in such a way that the resulting sub-stream forms another decodable bit-stream for the decoder, which represents the source content in a reduced reconstruction quality compared to the original bit-stream. This technique is different to *simulcast*, where a video content is first independently coded into different bit-streams with different resolutions or qualities and then all these bit streams are simultaneously transmitted. Simulcast is not efficient since there is much redundancy among the bit streams.

Scalable video coding is not a new research area [6]. Previous coding standards, i.e. H.262, MPEG-2 Video, H.263, and MPEG-4 Visual already include certain degree of scalability. However, before H.264/SVC, none of video coding standards provided full scalability and these scalable profiles of those standards have been never used extensively. The main reason for not using these scalable profiles is that these features involve a loss in coding efficiency and increase the decoder complexity as compared to simulcast.

SVC was approved in October 2007 as Amendment 3 of the Advanced Video Coding (AVC) standard ISO/IEC 14496-10 (also published as ITU-T Rec. H.264) [7]. The Moving Picture Experts Group (MPEG) Joint Video Team (JVT) in collaboration with the International Telecommunication Standardization Sector (ITU-T) developed a Scalable Video Coding (SVC) standard. As a part of the standardization process, the Joint Scalable Video Model (JSVM) [8] reference software has been developed as well.

The main goal of SVC standard [9] is to obtain a unique high-quality bit stream that contains one or more subset bit streams that can be decoded separately without affecting significantly the coding efficiency and reconstruction quality similar to the existing H.264/AVC with the same quantity of data. Compared with scalable profiles in previous video coding standards the coding efficiency of SVC is greatly improved [10].

The SVC bit stream is structured in several levels or layers of information, consisting of a base layer and several enhancement layers. Each enhancement layer has a base layer, from which it reuses parameters such as motion vectors or residuals as prediction. Each SVC base layer either has its own base layer or is layer zero. For compatibility reasons, SVC provides a base layer which is totally H.264/AVC compliant. That way, terminal devices that only implement H.264/AVC would still be able to present the lowest quality available in the SVC stream. Scalability modes enabled by the enhancement layer information include temporal (increase of frame rate), spatial (increase of picture resolution) and fidelity (increase of quantization accuracy) scalability. These three types of scalability can be combined or used separately into a single bit-stream.

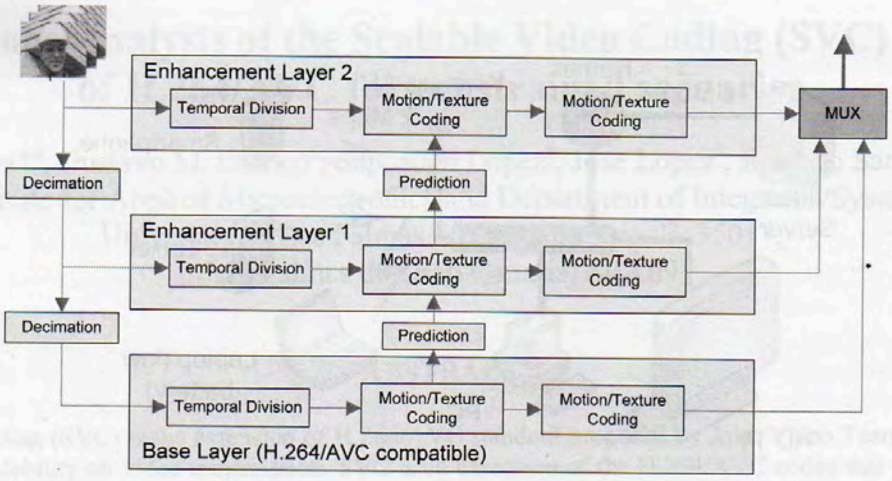

Figure 2. Scalable Video Coder structure

With temporal scalability several frames can be chosen for decoding and, then, obtaining a lower frame rate than original one. It is possible when sequence is divided in Group of Pictures (GOP) organized in a hierarchical structure as shown in Figure 3 where the four frame rates can be obtained in the decoder. For each T_k a bit tream can be obtained eliminating frames in higher layers than k.

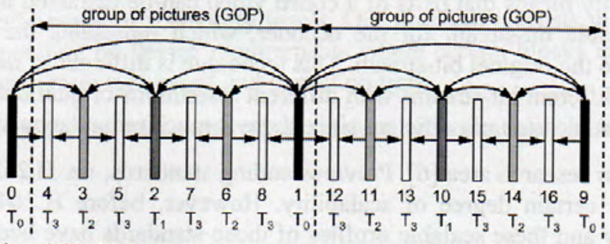

Figure 3. Example of temporal scalable bit stream

With spatial scalability several spatial resolutions can be chosen when decoding as shown in Figure 4. In this case three different spatial resolutions can be chosen extracting the information of each resolution for such frame.

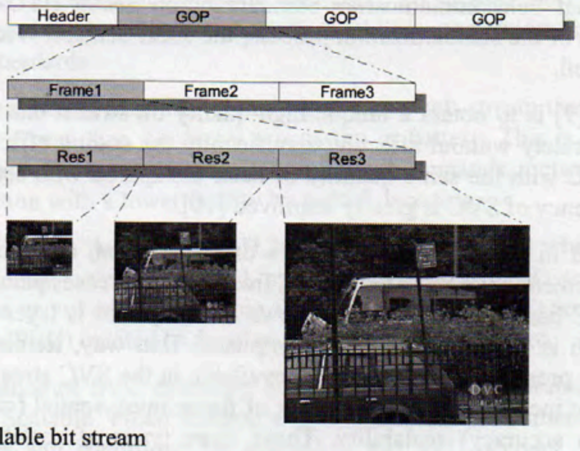

Figure 4. Example of spatial scalable bit stream

With the quality scalability, also called SNR scalability, it is possible to select between several quality levels in the decoding flow. Figure 5 shows an example with three quality layers, each one coded with a different quantification value. The information is contained in a separated field in the bit stream.

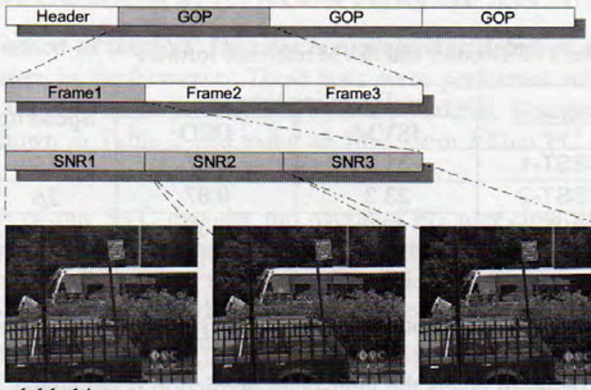

Figure 5. Example of quality scalable bit stream

In the SVC amendment, temporal and quality scalabilities should be supported by decoders without any restrictions. Only the spatial scalability is limited according to the profile defined. There are three profiles in the SVC amendment:

- Scalable Baseline profile: In this profile, the base layer must conform to H.264/AVC Baseline profile. Resolution ratios between successive spatial layers in both horizontal and vertical direction must be 1.5 or 2.
- Scalable High profile: In this profile, the base layer must conform to H.264/AVC High profile. No restrictions have been defined for the enhancement layers.
- Scalable High Intra profile: Only intra pictures in all layers are authorized. As the previous profile, the base layer should be conforming to H.264/AVC High profile.

2.2 Open SVC Decoder

The Open SVC Decoder (OSD) [11] is an open source library created at the IETR/INSA of Rennes that implements a SVC decoder written in C language. It has been developed and tested over different platforms: x86 processors, PDA and DSPs. OSD implements only the Scalable baseline profile. For our case of study, this profile is the most suitable for our proposals due to limited hardware resources and displays [12-13]. In addition to baseline profile, other restrictions has been applied as, no considering CABAC, disabling the use of B-slices in our bit streams and using only the constrained baseline profiles identifiers number 66 and 83. OSD has been modified to separate some blocks of the decoder for performance analysis and redirect the output from display to a YUV file.

The Open SVC Decoder can decode partially the bit stream until a specific layer with a specific temporal scalability as is shown in Figure 6. This characteristic allows controlling in the SVC decoder which layers are decoded, contrary to the JSVM which decodes all layers in a bit stream. Additionally, the OSD has been tested over the conformance sequences provided on the JVT site [14]. OSD is highly optimized, as shown in Table 1, where OSD is up to 50 times faster than JSVM reference software in its version 9.16.

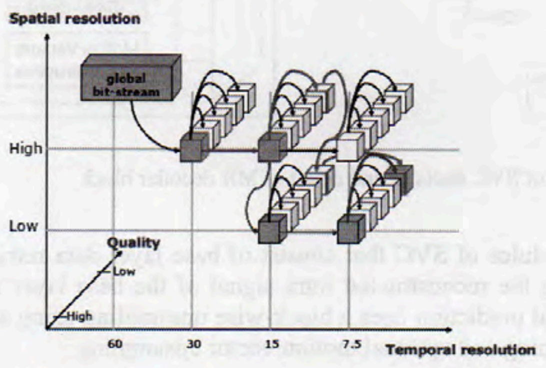

Figure 6. Scaling of global bit stream to obtain sub-bit stream

Table 1. Comparison between Open SVC Decoder and JSVM reference software

Sequence	Decoding time (s)		Speed up
	JSVM	OSD	
SVCBST-1	31.2	0.87	35
SVCBST-2	23.3	0.87	26
SVCBST-14	137	2.69	50
SVCBST-15	50	2.11	23

As we mention before, in Open SVC Decoder decode partially the bit stream until the chosen layer which is decoded and displayed. By dropping the bit stream of unwanted layer, the temporal, spatial, and quality scalabilities are achieved. To remove the redundant information between layers the inter-layer prediction is employed. On the other hand, some layers may need to be partially decoded in order to use these inter-layer data correlation. These partially-required layers are called sublayers.

A simplified flow diagram of the decoding process for a H.264/AVC compliant stream is shown in Figure 7 and processes of more interest for performance evaluation are marked by a dot and with shadow the new blocks in SVC [15]. The decoder reads the H.264 stream from an input buffer and decodes the NAL units in sequence. When the NAL unit contents information to decode and this information belongs to the selected layer or a sublayer this information continues to the Entropy Decoder (CAVLD). If the processed NAL is from the selected layer each macroblock (MB) is completely decoded, however, if the NAL information belongs to a sublayer then is partially decoded. After that, when a frame has been completely decoded, the deblocking filter is applied. Finally, the decoded pictures are stored in the right order in a YUV output file.

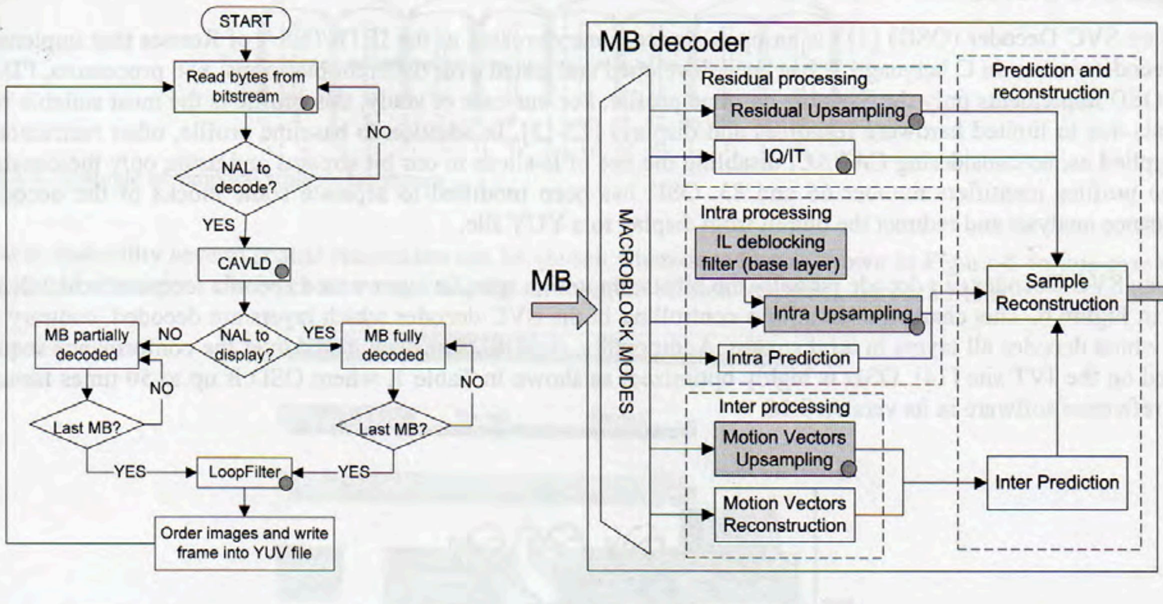

Figure 7. Simplified diagram flow of SVC decoder and detail of MB decoder block

Upsampling processes are new modules of SVC that consist of base layer data retrieval for inter-layer prediction. To improve quality before upsampling the reconstructed intra signal of the base layer are filtered using the H.264/AVC deblocking filter. Inter-layer residual prediction does a block-wise upsampling using a bi-linear filter. Inter-layer motion prediction uses macroblock partitioning and optional motion vector upsampling.

3. PERFORMANCE EVALUATION OF SVC DECODER

In this section the evaluation method of the SVC Decoder is explained in detail. A set of tests has been carried out to verify the decoder and to measure its performance. These tests were performed with different video sequences with various motions and textures. Considered sequences are Mobcal, Parkrun, Shields and Stockholm in 4CIF format. Encoder parameters used are shown in Table 2 and using an Intel Core 2 Duo PC at 2.5 GHz running Windows XP operating system.

For the performance evaluation of our SVC decoder the overall CPU user time dedicated to the decoding process, excluding I/O operations and system calls, of each sequence in each tested configuration was measured. All complexity measures are given in relative terms to guarantee as much as possible to be independent from the underlying hardware platform. As we mentioned, OSD has been divided into main blocks of decoder marked in Figure 7. In Table 3 the main blocks observed are listed and linked with OSD function names.

Table 2: Summary of encoder parameters

SPATIAL SCALABILITY:	QCIF, CIF, 4CIF
TEMPORAL SCALABILITY:	7.5, 15, 30 fps
SNR SCALABILITY:	QP: 36, 26, 16 (CGS)
FRAMES TO ENCODE:	500-600
ENTROPY ENCODER:	CAVLC
OTHER RESTRICTIONS:	DISABLEBSLICES, PROFILE_IDC = 66 | 83

Table 3: Correspondence of each block of SVC decoder with functions in Open SVC Decoder (* = any words)

Functional block	**Function name**
IQ/IT	ict_4x4_residual_C + rescale_4x4_*
INTERLAYER MOTION PREDICTION	sample_interpolation + _SampleInterpolation8x8
DEBLOCKING FILTER	filter_mb_svc
INTERLAYER DEBLOCKING FILTER	filter_mb (only when is called by MB_Spatial)
CAVLD	Residual
BITSTREAM PROCESSING	GetNalBytesInNalUnit
RESIDUAL UPSAMPLING	upsample_*
INTRA UPSAMPLING	upsample_mb_luninance + _upsample_mb_chroma

4. RESULTS AND PROPOSED RECONFIGURABLE ARCHITECTURE

In this section results of performance analysis are shown.

4.1 Temporal scalability

SVC achieves temporal scalability thanks to a hierarchical temporal prediction structure as was explained in Figure 3, in fact, H.264/AVC already support in the non scalable profiles the temporal scalability. For that reason, it is not weird that the results are quiet similar to a H.264/AVC decoder. In Figure 8 deblocking filter concentrate 40% of overall time spent in the algorithm. The computational cost of entropy decoder decreases with each new temporal scalability layer because much more information is predicted from neighborhood frames.

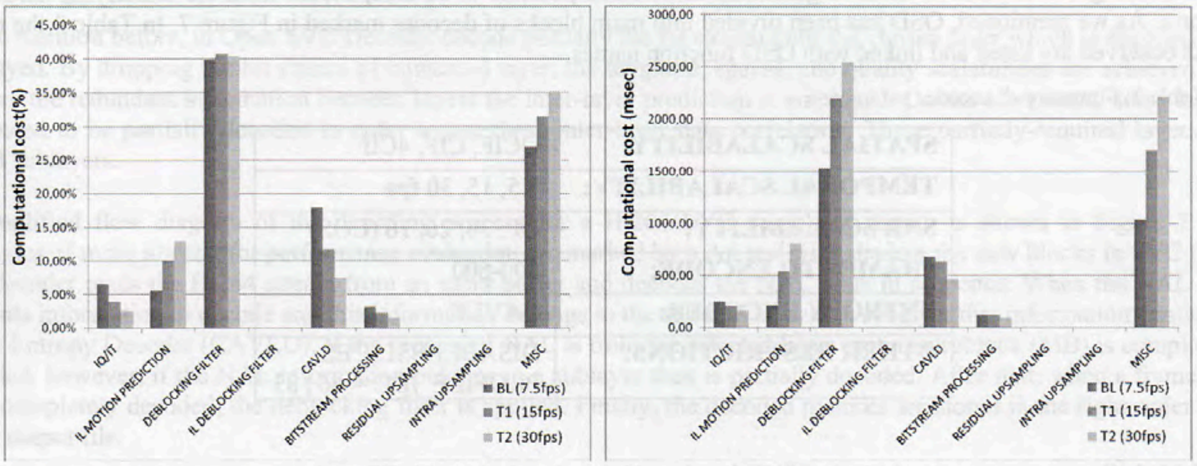

Figure 8. Results for temporal scalability

The influence of deblocking filter block is stable when temporal scalability increase because it depends directly on the number of frames decoded. It should be considered as a good candidate to be implemented in hardware with this scalability type.

4.2 SNR scalability

Basically, quality scalability modifies the values of quantification parameter to obtain different quality levels. Then, the entropy decoder, that is the responsible of decoding bit stream information, has to decode more coefficients with each new quality layer with high quantification differences. On the other hand, the deblocking filter does not increase his weight because the number of filtering operations is the same for all cases. Proportionally, deblocking filter loses importance for the benefit of entropy decoder as it is shown in Figure 9.

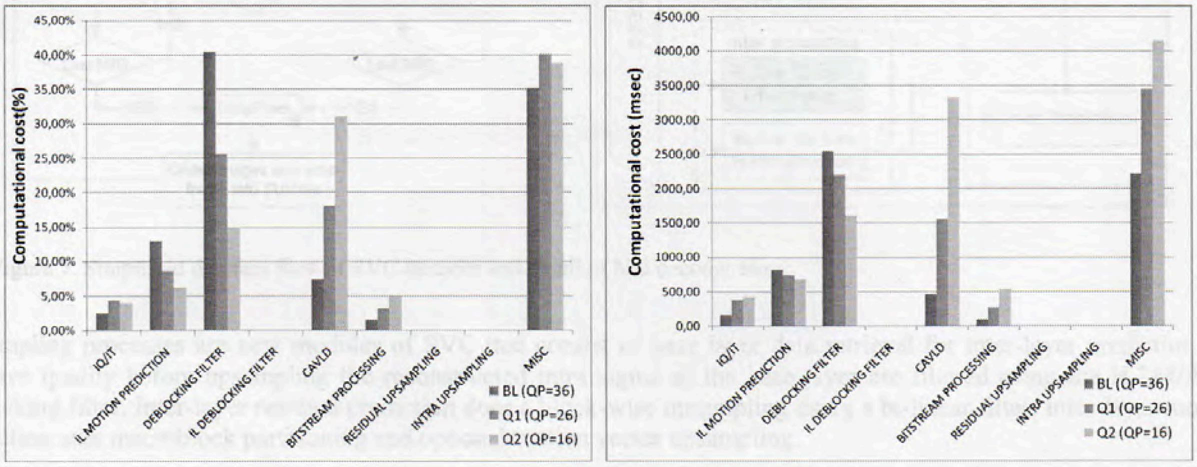

Figure 9. Results for SNR scalability

4.3 Spatial scalability

With this kind of scalability the computational number of macroblocks is multiplied by four in our test conditions with each new layer. As a consequence of that computational cost increases geometrically in modules that are used in each and for all the macroblocks as is shown in Figure 10. That is the case of deblocking filter that is applied to all the macroblocks in a frame that increase his size by four. Additionally, only for spatial scalability, the inter-layer prediction modules (inter-layer deblocking filter, residual upsampling and intra upsampling) are used. All these operations together represent important blocks that can be eliminated completely whether spatial scalability is not used.

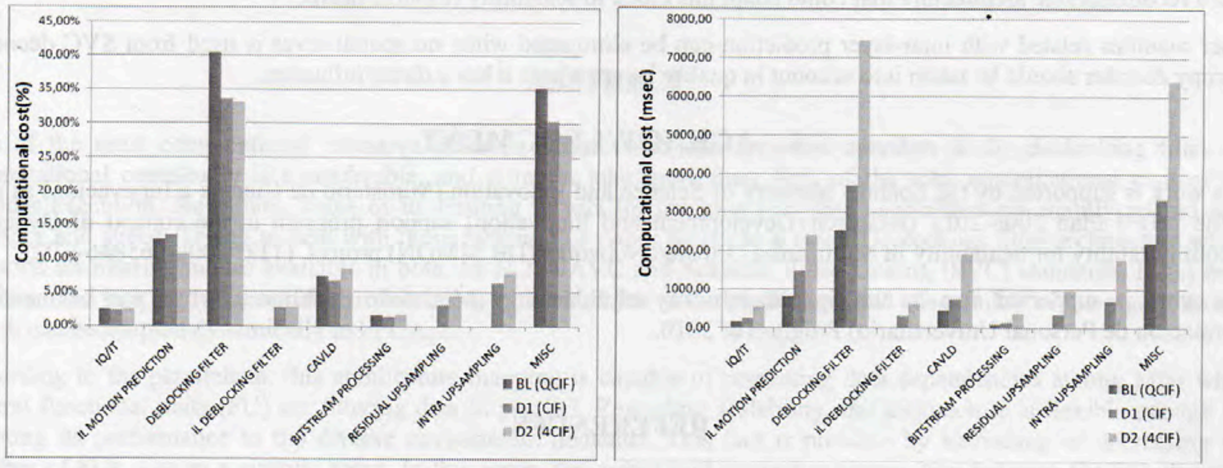

Figure 10. Results for spatial scalability

4.4 Suggested SVC decoder architecture

Our proposal is determined by the results obtained for each scalability type. As they are independent from each other the proposed architecture should be specified by the scalability characteristics of sequence to decode. Then, a reconfigurable architecture of course grain is the most suitable architecture to high area efficiency in constrained devices. In Table 4 proposed characteristics for SVC decoder architecture are shown.

Table 4. Proposed architecture characteristics in function of scalability types

	Scalability		
Functional blocks	Temporal	Quality	Spatial
CAVLD	Linear increase	Exponential increase. HW implementation suggested	Exponential increase. HW implementation highly suggested
DEBLOCKING FILTER	High influence (stable). HW implementation suggested	High influence (stable). HW implementation suggested	Exponential increase. HW implementation highly suggested
RESIDUAL and INTRA UPSAMPLING	Not used. Remove from decoding process	Not used. Remove from decoding process	Low influence (< 15%) in overall performance
INTER-LAYER DEBLOCKING FILTER	Not used	Not used	Low influence (< 5%). HW reuse of deblocking filter is possible
BITSTREAM PROCESSING and IQ/IT	Low influence (< 5%) in overall performance of SVC decoder for all cases with enhancement layers.		
INTER-LAYER MOTION PREDICTION	Low influence (< 15%) in overall performance of SVC decoder for all cases with enhancement layers.		

5. CONCLUSIONS

SVC is a good approach for video broadcasting services over heterogeneous networks. The profiling analysis demonstrates that the scalability type has an important influence overall and in particular blocks of algorithm performance.

Deblocking filter has an important influence on overall performance with around a 40% of algorithm time in the temporal scalability. For that reason this is a good candidate to be implemented in hardware and should be implemented over a reconfigurable architecture that could adapt this block to scalability resource needs.

Other modules related with inter-layer prediction can be eliminated when no spatial layer is used from SVC decoder. Entropy decoder should be taken into account in quality layers where it has a direct influence.

6. ACKNOWLEDGMENT

This work is supported by the Spanish Ministry of Science and Innovation (Ministerio de Ciencia e Innovación) as part of the I+D+I Plan 2008-2011 (Research+Development and Innovation) support program in the context of Dynamic Reconfigurability for Scalability In Multimedia Oriented Networks (DR.SIMON) project. (TEC 2008-065846-C02)

This work is supported also by the Spanish Ministry of Education (Ministerio de Educación) as part of the FPU (Formación de Personal Universitario) Program of 2010.

REFERENCES

[1] Bennett, B., Dee, C., "Scalable Video Coding across heterogeneous networks," Military Communications Conference, 2008. MILCOM 2008. IEEE , vol., no., pp.1-6, (2008).

[2] Schierl, T., Stockhammer, T., Wiegand, T., "Mobile Video Transmission Using Scalable Video Coding," Circuits and Systems for Video Technology, IEEE Transactions on , vol.17, no.9, pp.1204-1217, (2007).

[3] Sangjin H., Changseob P., Keunsoo P., Munchurl K., "Implementation on a Real-Time SVC Encoder for Mobile Broadcasting," Consumer Communications and Networking Conference, 2008. CCNC 2008. 5th IEEE , vol., no., pp.1236-1237, (2008).

[4] Ziliani, F., "The importance of 'scalability' in video surveillance architectures," Imaging for Crime Detection and Prevention, 2005. ICDP 2005. The IEE International Symposium on , vol., no., pp. 29- 32, (2005).

[5] Wiegand, T., Sullivan, G.J., Bjontegaard, G., Luthra, A., "Overview of the H.264/AVC video coding standard," Circuits and Systems for Video Technology, IEEE Transactions on , vol.13, no.7, pp.560-576, (2003).

[6] Mrak, M., Sprljan, N., Izquierdo, E., "An overview of basic techniques behind scalable video coding," Electronics in Marine, 2004. Proceedings Elmar 2004. vol., no., pp. 597- 602, (2004).

[7] ITU-T and ISO/IEC JTC 1, ITU-T Recommendation H.264 - ISO/IEC 14496-10(AVC), "Advanced Video Coding for Generic Audio Visual services, Amendment 3: Scalable Video Coding," (2007).

[8] Joint Scalable Video Model JSVM-9.16, Available in CVS repository at Rheinisch-Westfälische Technische Hochschule (RWTH) Aachen.

[9] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the Scalable Video Coding extension of the H.264/AVC standard," IEEE Trans. Circuits Syst. Video Technol., vol. 17, no. 9, pp. 1103–1120, Sept. (2007).

[10] Alfonso, D., Gherardi, M., Vitali, A., Rovati, F., "Performance analysis of the scalable video coding standard," Packet Video 2007 , vol., no., pp.243-252, 12-13 (2007).

[11] M. Blestel and M. Raulet, "Open SVC decoder: a flexible SVC library," Proceedings of the international conference on Multimedia (MM '10). ACM, New York, NY, USA, 1463-1466, (2010).

[12] Horowitz, M., Joch A., Kossentini F., and Hallapuro A., "H.264/AVC baseline profile decoder complexity analysis. Circuits and Systems for Video Technology, IEEE Transactions on, 13(7):704–716, (2003).

[13] Maiti, S.N., Gupta, A., Piccinelli, E.M., and Saha, K., "Real-time SVC Decoder in Embedded System," In Proceedings of SIGMAP. (2009).

[14] Joint Video Team, "Conformance testing," http://wftp3.itu.int/av-arch/jvt-site/bitstream_exchange/SVC/, (2008).

[15] Pelcat, M., Blestel, M. , Raulet, M., "From AVC decoder to SVC: minor impact on a dataflow graph description," in Proc. IEEE PCS, Lisboa, Portugal, (2007).

Scalable 2D architecture for H.264 SVC deblocking filter with reconfiguration capabilities for on-demand adaptation

T. Cervero[a], A. Otero[b], E. de la Torre[b], S. López[a], G.M. Callicó[a], T. Riesgo[b], R. Sarmiento[a]
[a]IUMA, Univ. of Las Palmas de Gran Canaria, Campus de Tafira s/n, 35019, Canary Islands, Spain;
[b]CEI, Polytechnic Univ.of Madrid, José Gutiérrez Abascal 2, 28006, Madrid, Spain.

ABSTRACT

One of the most computational intensive tasks in recent video encoders and decoders is the deblocking filter. Its computational complexity is considerable, and it might take more than 30% of the total computational cost of the decoder execution. Nowadays, some of its limiting factors for reaching real-time capabilities are mainly related with memory and speed. Trying to deal with these factors, this paper proposes a novel Deblocking filter architecture which supports all filtering modes available in both the H.264/AVC and Scalable Video Coding (SVC) standards. It has been implemented in a hardware scalable architecture, which benefits of the parallelism and adaptability of the algorithm and which can be adapted dynamically in FPGAs.

Regarding to the parallelism, this architecture mapping is capable of respecting data dependencies among MBs while several functional units (FU) are filtering data in parallel. Regarding scalability, the architecture is flexible enough for adapting its performance to the diverse environment demands. This fact is possible by increasing or decreasing the number of FUs, like in a systolic array. In this sense, this paper will present a composition between the FU proposed against the state-of-the art work.

Keywords: H.264/AVC, SVC, Deblocking Filter, MB-level parallelism, FPGA

1. INTRODUCTION

Advances in video coding technology together with fast developments in network infrastructures, storage capacity, and computing power are enabling an increasing number of multimedia applications, such as video streaming contents on-demand, etc. Video standards try to achieve social expectations according to technology advances. In this sense, Scalable Video Coding (SVC) standard [1] is an extension of previous H.264/AVC standard [2], both normalized by the Joint Video Team. Its goal is to broadcast video information to multiple users, with multiple displays and connected through multiple networks delivering to all of them the same video bitstream. Improvements in performance of both standards, H.264/AVC and SVC, are at cost of a higher computational cost and complexity [3]. These constrained characteristics become almost unfeasible to fulfil with real-time restrictions by using pure software solutions. Current trend is to use embedded hardware/software platforms, and afterwards, parallelize the execution of the application as much as possible. These platforms often run the most complex tasks in hardware, and the rest of the application in software.

Different profiling exercises, such as [4], [5], demonstrate how the Deblocking-Filter (DF) algorithm represents a critical task in H.264/AVC encoding/decoding loop, which might spend more than 30% of the total execution time. This results are more critical in SVC since the DF can be used not only at the end of the reconstruction loop, but also as part of the inter-prediction stage for recovering spatial information within the scalable video bitstream. In base of these data, the DF is a suitable candidate for being hardware-accelerated, as part of a higher system. In fact, there exist several DF hardware implementation approaches in the literature. They are mainly focused on reducing cycles on a macroblock filtering process, exploiting different techniques [6-8]. However, their common drawbacks are that they have to process MBs sequentially, filtering them one by one, and that almost everyone of them are, static solutions. This means they are not able to adapt their performance according to the environment variations. Trying to overcome these issues, this paper presents a novel 2D coarse-grain scalable DF architecture based on a flexible systolic array structure. In addition, this structure exploits a new MB-level parallelism strategy which reduces cycles with respect to multiprocessor architectures [9-10].

The rest of the paper has been organized as follows. Section 2 presents a deep explanation of the DF algorithm, including its functionality into the H.264/AVC and SVC standards. Section 3 explains the concept of systolic array and how it has been used in the literature for implementing scalable architectures. These sections allow introducing the DF architectural proposal, which is explained in Section 4. Once the hardware structure has been exposed, Section 5 presents synthesis results and a comparison with other approaches belonged to the state-of-the-art. Finally, Section 6 summarizes main conclusions.

2. DEBLOCKING FILTER BLOCK INTO THE H.264/AVC AND SVC VIDEO STANDARDS

All video sequences are composed by a set of consecutive pictures or frames. Afterwards, each frame is divided into smaller units called slices. More in detail, one slice is formed by macroblocks (16x16 pixels of luminance and two 8x8 pixels of chrominance). These ones, in turn, are organized in groups of 4x4 pixels called blocks, numbered from zero to 23 (16 corresponding to luminance and 4 for each chrominance). Finally, each set of 4 consecutive pixels form a line of pixels (LOP); hence each block is composed by 4 LOPs.

As part of the last stage of encoding and decoding H.264/AVC and SVC loops, the DF filters each decoded macroblock (MB) of a picture. Its goal is to improve the appearance of a decoded picture, by reducing blocking distortion [11] produced by previous tasks. However, not every edge of a picture is filtered, as those ones belonging to the border of the picture are not filtered.

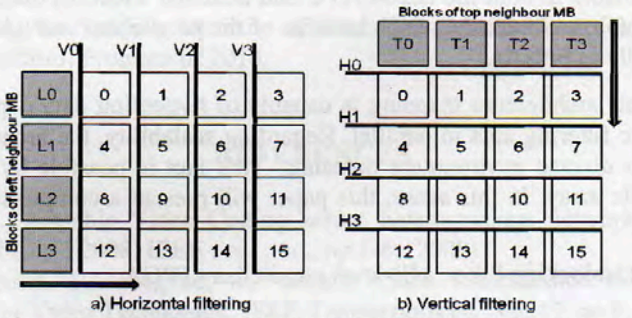

Figure 1. MB edges for a) horizontal and b) vertical filtering

In order to compliant with H.264/AVC and SVC standards, DF filters all the 4x4 blocks edges in a MB in a specific order. First at all, vertical block's edges have to be processed from left to right, and then horizontal ones from top to bottom. These edges are shown in Figure 1. As this figure depicts, vertical edges (V0, V1, V2, and V3) are filtered horizontally, whereas horizontal edges (H0, H1, H2, and H3) are filtered vertically. Standards are flexible about the filtering pattern to follow while vertical and horizontal orders are respected. These dependencies, among left and top neighbours, make difficult to exploit data parallelism into the DF execution, since they also introduce a higher level of complexity. In fact, many state-of-the-art approaches process pictures sequentially, MB by MB in a raster scan order, such as [12-14].

2.1 Deblocking Filter functionality

The DF is highly adaptive because it adjusts its filtering operations according to input data characteristics. The adjustment is provided and executed by two elements. One of them is boundary strength (BS) unit and the other one is a filter unit. The former calculates the strength of the filtering, which means how many pixels of each LOP will be modified by the filter unit. The latter processes two LOPs after evaluating some expressions. If they are true, filter processes data according to BS, else it passes data through without modified them.

2.1.1 Boundary Strength (BS)

This unit works before the filter itself, since it is responsible of determine filtering strength's values. It always returns a value between zero and four; where zero means data have not to be filtered, and four means the strongest filtering. Actually, BS value impacts on the amount of pixels per LOP that filter should process. In order to determine filtering strength, BS unit needs information related to the current the 4x4 block that is going to be filter (q), and its neighbour (p). Depending on their characteristics, BS takes one value or another, such as Table 1 details.

Table 1. BS values according to the current and neighbour MBs considered

Filtering conditions	*Boundary strength values*
If q and/or p are Intra coded, and boundary is a MB boundary.	bs = 4 (Strongest filtering)
If q or p is Intra coded but boundary is not a MB boundary.	bs = 3
Neither q nor p is Intra coded and they contain coded coefficients.	bs = 2
Neither q nor p is *Intra* coded and: Their motion vectors differ by one sample or more. They have different number of reference pictures. They use different reference pictures.	bs = 1
Otherwise.	bs = 0 (No filtering)

2.1.2 Filter unit

Its goal is to process MBs for smoothing their blocks artefacts. In spite of a MB is the input data unit into the DF, it contains many information for being filtered at once. Hence, it is divided into smaller data units, which make easier filtering process. Thus, every filter unit processes two LOPs each time, according to BS value, when it is necessary. One of these LOPs belongs to the block which it is being currently filtered (q), and the other one is its neighbouring data (p). However, a BS different to zero does not ensure that filter processes its inputs. There is also to consider the relationship among q and p pixels. If they do not fulfil with the expressions (1), LOPs go through filter unit unmodified.

$$\begin{gathered} |q_0-p_0| < \alpha \\ |p_1-p_0| < \beta \\ |q_1-q_0| < \beta \end{gathered} \qquad (1)$$

Alpha (α) and beta (β) are thresholds defined in the H.264/AVC and SVC standards by empiric experiments. They increase with the average quantiser parameter QP of the two blocks p and q. Once all the clauses of (1) have been checked, filtering operations are executed. As it was mentioned before, depending on the BS value different amount of pixels are processed. This issue is showed in Table 2, in which on the left column is represented the different BS values, and on the right one pixels transformations are calculated.

Table 2. Filtering operations according to the BS value

Boundary Strength values	*Filtering operations*
$BS = 0$	$q_0' = q_0;\ q_1' = q_1;\ q_2' = q_2;\ q_3' = q_3$ $p_0' = p_0;\ p_1' = p_1;\ p_2' = p_2;\ p_3' = p_3$
$1 \leq BS \leq 3$	$q_0' = q_0 - \Delta_0;\quad \Delta_0 = \min(MAX(-c_0, \Delta_{0i}), c_0)$ $p_0' = p_0 + \Delta_0;\quad \Delta_{0i} = [4+4\cdot(q_0 - p_0)+(p_1 - q_1)] >> 3$
	If $\lvert q_2 - q_0\rvert < \beta$ then: $q_1' = q_1 + \Delta_{q1};$ $\Delta_{q1} = \min(MAX(-c_0, \Delta_{q1i}), c_0)$ $\Delta_{q1i} = (q_2 - 2\cdot q_1 + ((p_0+q_0+1) >> 1)) >> 1$
	If $\lvert p_2 - p_0\rvert < \beta$ then: $p_1' = p_1 + \Delta_{p1};$ $\Delta_{p1} = \min(MAX(-c_0, \Delta_{p1i}), c_0)$ $\Delta_{p1i} = (p_2 - 2\cdot p_1 + ((p_0 + q_0 + 1) >> 1)) >> 1$
$BS = 4$	If $((\lvert q_2 - q_0\rvert < \beta)\ \&\ (\lvert p_0 - q_0\rvert < (\alpha >> 2) + 2)\ \&\ luminance)$ then: $q_0' = (q_2 + 2\cdot q_1 + 2\cdot p_0 + p_1 + 4) >> 3$ $q_1' = (q_2 + q_1 + q_0 + p_0 + 2) >> 2$ $q_2' = (2\cdot q_3 + 3\cdot q_2 + q_1 + q_0 + p_0 + 4) >> 3$
	If $((\lvert p_2 - p_0\rvert < \beta)\ \&\ (\lvert p_0 - q_0\rvert < (\alpha >> 2) + 2)\ \&\ luminance)$ then: $p_0' = (p_2 + 2\cdot p_1 + 2\cdot q_0 + q_1 + 4) >> 3$

Boundary Strength values	***Filtering operations***
BS = 4	$p_1' = (p_2 + p_1 + p_0 + q_0 + 2) >> 2$ $p_2' = (2\cdot p_3 + 3\cdot p_2 + p_1 + p_0 + q_0 + 4) >> 3$
	If $((\lvert q_2 - q_0\rvert > \beta) \ \& \ (\lvert p_0 - q_0\rvert > (\alpha >> 2) + 2) \ \&$ *luminance*) then: $q_0' = (2\cdot q_1 + q_0 + p_1 + 2) >> 2$
	If $((\lvert p_2 - p_0\rvert > \beta) \ \& \ (\lvert p_0 - q_0\rvert > (\alpha >> 2) + 2)$or *chrominance*) then: $p_0' = (2\cdot p_1 + p_0 + q_1 + 2) >> 2$

2.2 Deblocking Filter into the H.264/AVC and SVC standards

H.264/AVC is one of the newest video coding standards developed by ITU in 2003. Its original aim was to provide similar functionality to earlier standards, such as MPEG-4, but with significantly better compression performance and improved support for reliable transmissions. Besides, the standard was designed to facilitate implementation on as wide range of processor platforms as possible. Into the H.264/AVC encoding and decoding loop, the DF is the last stage on the reconstruction picture.

Regarding to SVC, it was developed like an extension of the H.264/AVC standard. With backward compatibility purposes, it reuses most of its components. This fact is highlighted on Figure 2 that represents a schematic of the decoding loop process. H.264/AVC video bitstreams go through the bitstream processing stage, and then follow the upper way of the image. This process in included into the decoding loop process of the SVC, with the unique difference that a new stage is added, named inter-layer prediction stage in figure 2.

SVC standard is scalable because it delivers one common bitstream to different users with different displays and bandwidth characteristics. This is possible thanks to its hierarchical bitstream structure, which is divided into two differentiated layers. A base layer (BL), which is H.264/AVC completely compliant, and enhancement layers (EL). BL bitstream corresponds to a minimum quality, frame rate, and resolution information; while each EL introduces extra information to the BL. However, EL information always depends on the BL decoded data. On the other hand, as part of the EL information there are three possible scalabilities: spatial, temporal and quality. The first one allows choosing diverse sequence resolutions in terms of image dimensions, i.e. QCIF, CIF, 4CIF, etc. The second one is used to choose convenient frame rates, i.e. 7.5fps, 15fps or 30fps. The last one allows choosing suitable data rates according to the environment conditions, display device resources, etc. Within these scalabilities, temporal and quality data increase the amount of information into the decoding loop, but always reusing the H.264/AVC functions. However, SVC introduces new tasks into the decoding loop, identified by inter layer prediction in Figure 2. These are related with the spatial scalability, since it is the one which modifies the final image properties. Every EL uses the decoded information from its previous layer, till the BL one. But every decoded spatial layer reconstructs a picture, and consequently this process makes use of the DF services. Each time the DF is used within the inter-layer prediction loop it is called inter-layer deblocking filter (ILDF). In this way it is possible to differentiate the ELs decoding from the normal use of the DF as final stage into the decoding loop.

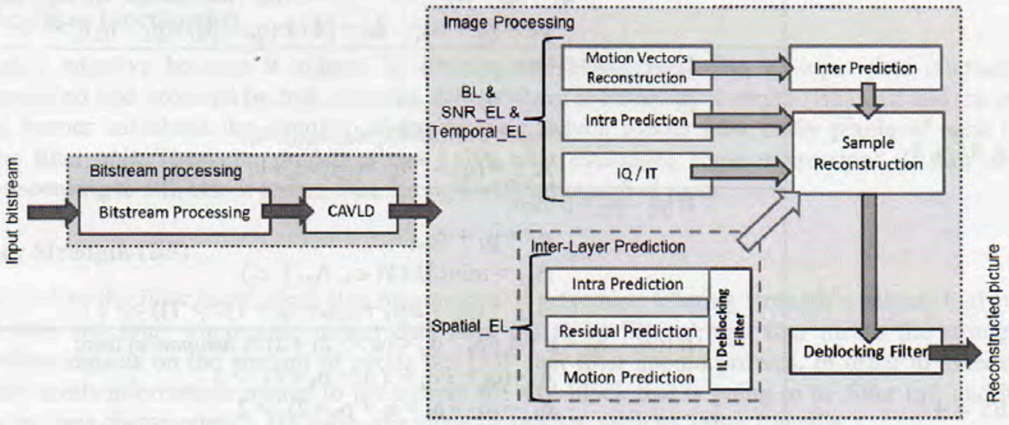

Figure 2. High-level SVC decoder schematic

3. SYSTOLIC ARRAYS

Scalable architectures allow the modification of its size to adapt the dimensions of the operation they carry out, as well as to adjust some design features, like the trade-off between resource consumption and offered performance. Adaptation of the operation size increases the architecture reusability among applications, while balancing area and performance, allow envisaging more flexible systems. Regarding the former one, an example may be a scalable DCT which could be adapted to different block sizes depending on the specific requirements of each coding standard [17]. Area-performance trade-offs are explored, for instance, in [18].

Tightly associated with scalability, modularity and regularity are capital architectural features, because of their impact on the cost of the scalability process. Considering these requirements, Systolic Arrays (SA) and Distributed Arithmetic (DA) are the most extended approaches to design and implement scalable architectures. Distributed arithmetic is a way of carrying out basic operations based on its bit-wise decomposition, well suited to FPGA implementations due to its LUT based nature [19]. In [18], for instance, this technique is applied to calculate the scalar product of a variable vector and a constant, in a number of clock cycles proportional to the number of modules included in the architecture, to carry out parallel partial additions on partial results. On the other hand, Systolic Arrays are regular and inherently pipelined arrays of processing elements working in parallel [20]. Several examples of scalable architectures based on SA can be found in the state of the art, like [21], providing a scalable 2D wavelet transform, as well as the scalable DCT in [17]. Regarding its granularity, Systolic Arrays can perform complete computing intensive tasks while Distributed Arithmetic architectures applicability is reduced to carry out single arithmetic operations. Considering the computational requirements of DF, a Systolic implementation has been exploited in this work. In addition, SAs offer some advantages like its intensive processing capabilities as well as the regularity of its communications. This fact, together with its pipelining nature, is exploited in the present work, since most data is shared between neighbouring elements of the architecture, reducing the number of accesses to the external memory.

Regarding the way of exploiting architectural scalability, it can be static or dynamic, that is, the size of the architecture can be adapted at design time or at run-time. Considering design-time scalability, main approach is to include generic parameters in the HDL architectural description, making easier the later design customization. A general parameterization framework is offered in [22]. Differently, run-time scalability can be achieved by means of other techniques, like clock gating the unused elements of the architecture [23]. This approach has been successfully applied with the purpose of reducing power consumption. However, it does not allow the release of the unused area in the reconfigurable device when shrinking the scalable architecture, to make it really available for being used by other processing cores which would have to be executed simultaneously. The alternative that offers this benefit is the use of Dynamic and Partial Reconfiguration (DPR) of modern FPGAs, a technique that allows changing the configuration of the desired area of the device while the rest remains working. This technique increases resource sharing, and permits a flexible distribution of resources in order to fulfil run-time variable conditions. In this case, modularity feature of these architectures reduces extremely the overhead of scaling the architecture, both considering reconfiguration time and configuration information storage necessities, since the same module can be replicated and relocated in different positions of the device. An example of scalability by means of DPR is the scalable FIR filter provided in [24] that offers the capability of adapting the number of taps to adjust the filter order, in order to set a compromise between filtering quality and required resources. Also the scalable DCT implementation in [17] allows varying the number of DCT coefficients calculated by the core to adapt the number of coefficients that will be subsequently quantified, and therefore, video coding quality. Regarding the video applications area, and related with the same algorithm as in this proposal, a DPR scalable DF architecture is proposed in [25]. However, this work explores block level parallelism, that is, the variation of number of processing units working in parallel affects the execution time of a single MB, without dealing with Macroblock dependences. This approach, while still being interesting for flexibility purposes, is limited by the maximum number of 4×4 blocks that can be processed simultaneously.

In this work, a DF architecture based on a SA structure is proposed. Scalability, in this first version, refers to static design-time adaptation, but scalability by means of DPR will be addressed in the future work.

4. ARCHITECTURAL PROPOSAL

4.1 Architectural characterization

The proposed hardware DF architecture, depicted on Figure 3, is based on a coarse grain scalable systolic array of functional units (FU). On the one hand, it is scalable because it allows modifying its number of FUs according to on application demand. On the other hand, it is coarse grained due to its basic data units are MBs. Both features make possible to explore a new MB-level parallelism technique, which allows processing several MBs simultaneously. Moreover, in order to exploit all advantages of a scalable systolic architecture, this proposal takes advance of reconfiguration capabilities of some hardware devices, such as modern Xilinx FPGAs. As a typical reconfigurable design, this architecture is separated into two stages: the static one and the reconfigurable one. In figure 3 these stages are perfectly identified. Static stage is composed by control units, and they are always present independently of the reconfigurable array configuration implemented. However, reconfigurable array is formed by replicating four kinds of elements (IM, Router, FU and OM). Independently of the reconfigurable array dimensions, any element modifies its behaviour. All these set of elements execute a DF task, exploiting the MB-level parallelism inherent to this algorithm.

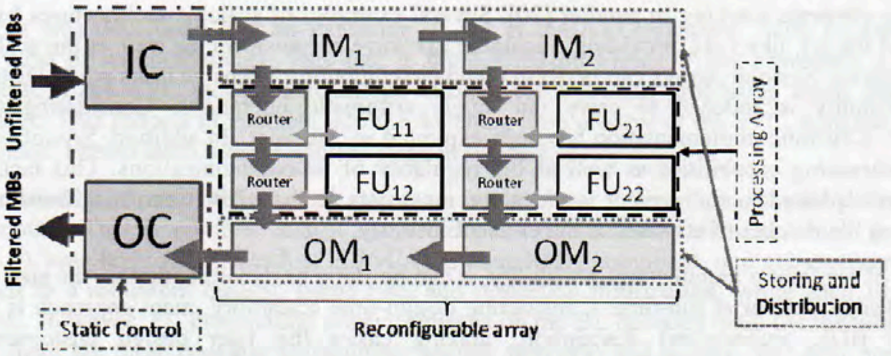

Figure 3. Coarse-grained architectural proposal

4.2 Elements description

As a whole, this architecture might group all its elements according to their main functionality into three categories: control elements, storage and distribution elements, and computational elements. Control elements always belong to the static part of the architecture, while the rest of elements are part of the reconfigurable array.

4.2.1 Control elements

Belonging to this category are Input Control (IC) and Output Control (OC) blocks. Both together manage all tasks related to data transfers, synchronization signals, and reconfigurable array dimensions. They are also the interface with external system and they read/write in reconstructed memory. IC transfers data to the reconfigurable array, while OC transmits filtered data out of the architecture.

4.2.2 Storage and distribution elements

All these elements belong to the reconfigurable array, and their function is to distribute data through the array. Input Memory (IM) is the top element of every column of the array, while Output Memory (OM) is always the bottom one. They are composed mainly by FIFOs and their specific goal is to transfer and store unfiltered and filtered MBs along the array, respectively. IM receives unfiltered MBs from IC and send them to its right neighbour until all of them have filled their FIFOs. Then, each one sends MBs to its own column of computational elements. On the other hand, OM receives filtered MBs from its own column of computational elements, and transmits them to their left OM neighbour, until reach OC.

There is also another element into this category named router. It is allocated within the processing array, and it always precedes a computational element. About its functionality, it receives MBs from IC, where the first one is sent to its computational element, and the rest of MBs are going through unmodified. Once processing stage has finished, filtered MBs are transmitted to OC from computational elements and then through routers.

4.2.3 Computational elements

Functional units (FU) are characterized for being computational elements. They are the processing core of the reconfigurable array, since on them MBs are filtered. Each FU is a Deblocking filter on its own capable of processing frames of MBs. But also it might be part of more complex 1D or 2D structures of $m \times n$ FUs, where m is the width of the array, while n is the height of it.

Within the reconfigurable array, all elements are synchronized for avoiding data losses or collisions. This means that not only all FUs are working simultaneously but also everyone is into the same filtering state. In order to parallelize the execution as much as possible, each FU has to run other tasks during filtering operations, such as transferring semi filtered MBs between direct top and bottom neighbours FUs. This is due to unfiltered MBs come in through the router element to the FU, but left and top neighbours MBs are also necessary for filtering the current MB properly. These data transfers go through direct links between FUs with their direct top and bottom neighbouring FUs. In this way intermediate memory accesses are avoided, and routers are not overloaded with data.

5. RESULTS

This section presents and analyzes an interesting set of results of the DF architectural proposal, explained in this paper. This architecture has been designed in RTL and synthesized in a Virtex-5 FPGA. Due to the huge set of reconfigurable array configurations, this paper is focused only on a subset of them, which seem to be enough for supporting requirements of modern video applications.

5.1 Array's configurations

Despite of many synthesis results have been obtained for multiple $m \times n$ arrays configurations, there are not significant differences among many of them. Hence, this section presents those configurations which are more interesting in terms for its analysis. Table 3 describes the differences, in terms of resources, among the selected configurations: 1×1, 1×2, 2×1, 2×2, 1×4, 4×1 and 4×4.

Table 3. Elements distribution according to the array configuration

Configurations	1×1	1×2	2×1	1×4	2×2	4×1	4×4
Nº ICs	1	1	1	1	1	1	1
Nº OCs	1	1	1	1	1	1	1
Nº IMs	1	1	2	1	2	4	4
Nº OMs	1	1	2	1	2	4	4
Nº Routers	1	2	2	4	4	4	16
Nº FUs	1	2	2	4	4	4	16

Some of these configurations have the same number of FUs, however their distribution along the reconfigurable array is not the same. As a consequence their number of distribution elements also varies. Control elements (IC and OC) are the same for all configurations, since they are the static stage. In addition, number of OMs and IMs coincides per configuration, and they correspond with the reconfigurable array width (n).

Furthermore, another remarkable consideration about this architecture, which impacts on final performance, is its interconnection structure. Independently of which configuration has been implemented, all elements follow the same pattern. Control elements are connected to distribution elements, while these ones communicate with their neighbours and routers. Finally, routers are directly connected with FUs, which are also intercommunicated with their top and bottom neighbours.

5.2 Synthesis results

One of the most important synthesis results is to know how many hardware resources are needed for implementing the simplest configuration: 1×1 array. This information is shown in Table 4. Hardware resources have been expressed in terms of number of slices, and number of BRAMs; whereas 1×1 array has been separated into its elements and their influence over the total results.

Table 4. 1×1 configuration synthesis results

HW resorurces	*IC*	*OC*	*%(IC+OC)*	*IM*	*OM*	*%(IM+OM)*	*Router*	*%Router*	*FU*	*%FU*	*TOTAL*
Slices reg	357	116	18%	172	124	11%	90	3%	1814	69%	2673
Slices LUTs	444	108	17%	134	226	11%	112	3%	2274	69%	3298
Block RAMs	1	0	7%	2	2	27%	0	0	10	67%	15

Control (IC, OC) and distribution elements (IM, OM and Router) together are less than 30% over the total synthesis results, while the higher hardware charge is on the FU, which needs 69% of resources. The higher $m \times n$ configuration implemented, lower control contribution over the total resources; since they are static elements which do not vary its size neither its control logic. In regard to distribution and computational elements, their percentages on hardware resources vary according to the $m \times n$ configuration. Number of IMs and OMs varies in the same amount than n, thus higher values of n will increase the contribution of IM and OM in hardware. In case of routers, they always increase their number according to $m \times n$ FUs. In any case, FUs hardware resources will be always higher than the rest of elements, thus their percentage will be always around 60% or more.

5.3 State-of-the-art comparison

Unfortunately synthesis results are not an appropriate comparison parameter, since they depends on the hardware device. This trouble makes difficult to establish a fair comparison between the architectural proposal presented in this paper, and the rest of the state-of-the-art approaches. As a solution, this section presents a speed up comparison with many hardware architectures, but it also describes the general characteristics of every approach.

Table 5 compares reconfigurable arrays configurations selected previously, with eight state-of-the-art hardware approaches. First column is a reference to each approach. Then, columns two and three specify in which hardware device has been implemented or simulated each one. Following three columns describe some behavioural and synthesis results. Finally, last two columns consider clock cycles results for filtering a full frame of one CIF and one UHDTV frames.

In order to present a fair comparison, throughput values only consider execution time related to the filtering process. In this sense for, *1×1, 1×2, 1×4, 2×1, 2×2, 4×1* and *4×4* arrays we only have taken in consideration the processing array execution time, without including loading and downloading time from IC to IM, neither from OM to OC. As a consequence, all the arrays with the same number of FUs, and also the same number of clock cycles per MB, will get same results.

Table 5 State-of-the-art comparison

Architecture	*Device*	*Family*	*clock cycles per MB*	*Max. Freq. (Mhz)*	*clock period*	*CIF Clock cycles*	*UHDTV Clock cycles*
1×1 (1FU)			208			82368	26956800
1×2/2x1 (2FU)			208			41392	13478608
4×1/2x2 (4FU)	FPGA	Xilinx V5	208	124	8,06452E-09	23088	6789328
1×4 (4FU)			416			46176	13578656
4×4 (16FU)			416			**18720**	**3399968**
[6]	FPGA	Xilinx V-II	608	42,8	2,33645E-08	240768	78796800
[26]	FPGA	Xilinx V-IIP	256	135	7,40741E-09	101376	33177600
[7]	FPGA	Altera Statrix-II	152	126	7,93651E-09	60192	19699200
[27]	Standard Cell	0.18um	204	200	5,00E-09	80784	26438400
[8]	Standard Cell	0.18um	228	100	1,00E-08	90288	29548800
[28]	Standard Cell	0.18um	192	200	5,00E-09	76032	24883200
[28]	FPGA	Xilinx V-II	192	76	1,31579E-08	76032	24883200
[29]	Platform	ARM ESL7.1	306	200	5,00E-09	121176	39657600
[30]	FPGA	Xilinx V4	204	200	5,00E-09	80784	26438400

According to table 5, neither 2×2 nor 4×1 arrays share their numbers of clock cycles per MB. The same happens with 4×4 array. This is due to the architectural proposal has a limitation on its height. Clock cycles per MB have to be enough for transmitting data along all the elements on the same column; consequently the array height is lower or equal to two then 208 clock cycles are enough for fulfil with transfers constraints. Otherwise, clock cycles have to be increased, like it happens in 4×1 and 4×4 arrays.

Regarding to the performance, lower clock cycles are got by almost every proposed $m\times n$ configuration, which are 1×2, 2×1, 1×4, 2×2 and 4×4. In spite of 1×1 configuration is the worst within the selected subset of configurations, its results are similar to the state-of-the-art. These results demonstrate the benefits of exploiting parallelism.

6. CONCLUSION

This paper proposes a novel hardware DF architecture for H.264/AVC and SVC video standards. Its contributions, with respect to the state-of-the-art, are its high degree of parallelism, its scalability and its adaptability to the environment conditions on-demand.

Regarding the parallelism, this algorithm and its architectural mapping are capable of respecting data dependencies among MBs while several functional units (FU) are filtering data in parallel. Regarding scalability, the architecture is flexible enough for adapting its performance to the diverse environment demands. This fact is possible by increasing or decreasing the number of FUs, like in a systolic array. Thus, it means that the size of the architecture, mapped on HW, may grow in 1D or 2D, depending on the area restrictions in the FPGA, by using dynamic reconfiguration.

A comparison between different configurations of this architectural proposal, and the state-of-the-art, demonstrates how exploiting parallelism improves the execution time, reducing number of clock cycles for filtering a picture.

ACKNOWLEDGEMENTS

This work is supported by the Spanish Ministry of Science and Innovation and European Union (FEDER funds) as part of the I+D+I Plan 2008-2011) support program in the context of Dynamic Reconfigurability for Scalability In Multimedia Oriented Networks (DR. SIMON) project, under contract TEC 2008-065846-C02.

REFERENCES

[1] T. Wiegand, G. Sullivan, J. Reichel, H. Schwarz and M. Wien, "Joint Draft 9 of SVC Amendment", ISO/IEC JTC1/SC29/WG11 and ITU-T SG16 Q.6, Doc. JVT-V201, (2007).

[2] ITU-T Rec. H.264 and ISO/IEC 14496-10 (MPEG-4 AVC), ITU-T and ISO/IEC JTC; "H.264/AVC Standard Advanced Video Coding for Generic Audiovisual Services", Version 1 – Version 7; (2007).

[3] M. Wien, H. Schwarz and T. Oelbaum, "Performance Analysis of SVC", ISO/IEC JTC1/SC29/WG11 and ITU-T SG16 Q.6, Doc. JVT-U141, (2006).

[4] M. Horowitz, A. Joch, F. Kossentini and A. Hallapuro; "H.264/AVC baseline profile decoder complexity analysis". IEEE Trans. On Circuit Systems for Video Tech., 13(7), 704-716. (2003).

[5] I. Werda, T. Dammak, T. Grandpierre, M. Ayed and N. Masmoudi. "Real-time H.264/AVC baseline decoder implementation on TMS320C6416". Journal of Real-Time Image Processing Springer Berlin, 1-18, (2010).

[6] T. Warsaw and M. Lukowiak, "Architecture design of an H.264/AVC decoder for real-time FPGA implementation", Int. Conference on ASAP, 253-256, (2006).

[7] F. Tobajas, G.M. Callicó, P. A. Pérez, V. de Armas and R. Sarmiento, "An efficient double-filter hardware architecture for H.264/AVC Deblocking filtering", IEEE Trans. on Consumer Electronics, 54(1), (2008).

[8] M. Torabi, A. Vafaee and N. Movahhedinia, "A fast architecture for Deblocking filter in H.264/AVC using clock cycles saving process", IMPACT, (2009).

[9] M. Alvarez, A. Ramírez, A. Azevedo, C. Meenderinck, B. Juurlink and M. Valero. "Scalability of Macroblock-level Parallelism for H.264 Decoding", ICPADS, (2009).

[10] W. Kim, K. Cho and K. Chung, "Stage-based Frame-Partitioned Parallelization of H.264/AVC Decoding". IEEE Transactions on Consumer Electronics, 56(2), 1088-1096, (2010).

[11] I. E. G. Richardson, "H.264 and MPEG-4 Video Compression; Video Coding for Next-generation Multimedia", John Wiley & Sons Ltd., 159-224, (2003).
[12] Y.W. Huang, T.W. Chen, B.Y. Hsieh, T.C.Wang, T.H. Chang and L.G. Chen. "Architecture design for deblocking filter in H.264/JVT/AVC". Proc. IEEE Int. Conference on Multimedia and Expo, (2003).
[13] C.-C. Cheng, T.-S. Chang and K.-H. Lee. "An in-place architecture for deblocking filter in H.264/AVC". IEEE Trans. on Circuits and Systems II, 53(7), 530-534, (2006).
[14] T.-H. Tsai and Y.-N. Pan; "High efficient H.264/AVC deblocking filter architecture for real-time QFHD". IEEE Trans. on Consumer Electronics, 55(4), 2248-2256, (2009).
[15] M. N. Bojnordi, M. R. Hashemi and O. Fatemi. "A fast two dimensional deblocking filter for H.253/AVC video coding". CCECE, 2017-2020, (2006).
[16] MPEG, http://ip.hhi.de/imagecom_G1/savce/MPEG-Verification-Test/MPEG-Verification-Test.htm
[17] J. Huang and J. Lee, "A Self-Reconfigurable Platform for Scalable DCT Computation Using Compressed Partial Bitstreams and BlockRAM Prefetching," IEEE Trans. on Circuits and Systems for Video Technology, 19(11), 1623-1632, (2009)
[18] K. Danne, "Distributed arithmetic FPGA design with online scalable size and performance", Proc. of the 17th Symposium on Integrated Circuits and Systems Design, 135-140, (2004).
[19] Xilinx, http://www.xilinx.com/appnotes/theory1.pdf
[20] T. Ishimura, and A. Kanasugi, "A Design and Simulation for Dynamically Reconfigurable Systolic Array", Proc. of the Third International Conference on Convergence and Hybrid Information Technology, 2, 172-175, (2008).
[21] J. Chen and M.A. Bayoumi, "A scalable systolic array architecture for the 2D discrete wavelet transform", Proc. of the 38th Midwest Symposium, (1995).
[22] Z. Junchao, C. Weiliang and W. Shaojun, "Parameterized IP core design", Proc. of the 4th Int. Conference on ASIC, 744-747, (2001).
[23] J. Huang, J. Lee and Y. Ge, "An array-based scalable architecture for DCT computations in video coding", Proc. of the Int. Conference on Neural Networks and Signal Processing, 451-455, (2008).
[24] C.-S. Choi and H. Lee, "An Reconfigurable FIR Filter Design on a Partial Reconfiguration Platform," Int. Conference on Communications and Electronics, 352-355, (2006).
[25] R. Khraisha and J. Lee, "A scalable H.264/AVC deblocking filter architecture using dynamic partial reconfiguration," IEEE Int. Conference on Acoustics Speech and Signal Processing, 1566-1569, (2010).
[26] V.S. Rosa, A. A. Susin and S. Bampi, "An HDTV deblocking filter in FPGA with RGB video output", IEEE Int. Conference on VLSI, (2007).
[27] K. Xu and C.-S. Choy, "A five-stage pipeline, 204 cycles/MB, single-port SRAM-based Deblocking filter for H.264/AVC", IEEE Trans. on Circuits and Systems for Video Technology, 18(3), (2008).
[28] M. Nadeem, S. Wong, G. Kuzmanov and A. Shabbir, "Low-power, high-throughput Deblocking filter for H.264/AVC", Inter. Symposium on SoC, (2010).
[29] W. Jia, L. Liu, S. Yin, M. Zhu and Z. Wang, "A fast complete Deblocking filter on a coarse-grained reconfigurable processor supporting H.264 High Profile Decoding", Proc. PrimeAsia, 221- , (2010).
[30] J.-W. Hwang and J.-D. Cho, "A reconfigurable pipelined Deblocking filter for H.264/AVC", Int. Conference on Consumer Electronics, Digest of Tech. Papers, 403- , (2010).

Closing the Gap between Software and Hardware Super-Resolution Image Reconstruction: Provision of High-Quality Output

Tomasz Szydzik*[a], Gustavo M. Callico[a], Antonio Nunez[a]
[a]Institute for Applied Microelectronics and Department of Electronic Engineering and Control, University of Las Palmas de Gran Canaria, E–35017, Las Palmas de Gran Canaria, SPAIN

ABSTRACT

The ability of additional detail extraction offered by the super-resolution image reconstruction (SRIR) algorithms greatly improves the results of the process of spatial images augmentation, leading, where possible, to significant objective image quality enhancement expressed in the increase of peak-signal-to-noise ratio (PSNR). Nevertheless, the ability of providing hardware implementations of fusion SRIR algorithms capable of producing satisfactory output quality with real-time performance is still a challenge. In order to make the hardware implementation feasible a number of trade-offs that compromise the outcome quality are needed.

In this work we tackle the problem of high resource requirements by using a non-iterative algorithm that facilitates hardware implementation. The algorithm execution flow is presented and described. The algorithm output quality is measured and compared with competitive solutions including interpolation and iterative SRIR implementations. The tested iterative algorithms use frame-level motion estimation (ME), whereas the proposed algorithm relies on, performance-wise better, block matching ME. The comparison shows that the proposed non-iterative algorithm offers superior output quality for all tested sequences, while promising efficient hardware implementation able to match -at least- the software implementations in terms of outcome quality.

Keywords: Super-Resolution, image processing, quality improvement, quantitative quality, resolution enhancement

1. INTRODUCTION

We are living in a multimedia reality dominated by high quality visual content. The omnipresence of *high resolution* (HR) content, and the fact that it is encountered on a daily basis, results in elevated users' expectations of the content's quality. Thus, in order to meet consumer demands the legacy *low resolution* (LR) content quality has to be enhanced before being displayed. Content pre-processing/up-scaling methods include interpolation and *super-resolution image reconstruction* (SRIR) algorithms. SRIR is a discipline of mathematics that investigates methods for the estimation of spatially augmented representation of an image. The algorithms that SRIR encapsulates are diverse, but a feature that all SRIR algorithms have in common is the capability of additional detail extraction, where possible, from the low resolution content and their incorporation in the content spatially augmented representation. When carried out, this process can lead to significant image quality enhancement, superior to the one offered by interpolation[1]. However, computational requirements posed by SRIR algorithms are very high.

An interesting FPGA implementation of a super-resolution image reconstruction based on *iterative back projection* (IBP) is presented in [2]. In this approach, additional details are reconstructed based on exploitation of sub-pixel shifts caused by warping[3]. The processing is done at pixel level by means of weighted mean optical flow, denominated weight-based picture elements (hereafter *pels*) merging. Weights estimation is based on the inter-frames motion estimations for pel patch matching. The authors claim, that, with a loaded pipeline, their approach is capable of producing one super-resolved pel per cycle. The described architecture, implementing 10 iteration stages, was mapped onto a FGPA device reaching a frequency of 58 MHz. In this configuration the system was capable of super-resolving 61 CIF formatted LR images to 1280x720 pels per second, noting a three-fold speed up of execution time in comparison with the simulations using script high level abstraction language[4]. Nevertheless, a satisfying quality level requires 20 iteration stages, being out of reach for the target device. The number of implemented iterations is said to be limited by the available resources.

*tszydzik@iuma.ulpgc.es; phone +34 928 457 333; fax 1 +34 928 451 083; www.iuma.ulpgc.es

Another interesting implementation[5] of IBP algorithm targeting a development board hosting a FPGA device is also limited by the available resources allowing implementation of a maximal of 10 iterations. The implementation reaches an operating frequency of 80 MHz, which authors claim is sufficient to allow real-time execution outputting 25 VGA 2x super resolved frames. Nevertheless, the output quality is compromised due to the low number of iterations. The drawbacks of this implementation are high memory requirements and the fact that, in order to output a super resolved image, multiple passes through the hardware are required[2].

The aforementioned SRIR implementation bottlenecks are associated with resource requirements and/or memory access schemes. Contrary to the previous works, in our approach the to-be-implemented algorithm is non-iterative, and is expected to deliver satisfactory output quality when mapped on the target FPGA device.

This work is organized as follows. Section 2 presents a general classification of the SRIR algorithms. The non-uniform grid projection algorithm and its software implementation execution flow are presented in detail in, respectively, Sections 3 and 4. Competitive software implementations, namely iterative back projection[6], robust super-resolution[7] (RSR), and the interpolation algorithms[8], used as benchmarks to evaluate the super-resolution (SR) output quality are the topic of Section 5. In Section 6 the experimental set-up and test procedure is described and the measured objective quality values from test runs are presented and compared. Finally, conclusions are drawn in Section 7.

2. SUPER-RESOLUTION IMAGE RECONSTRUCTION ALGORITHMS

Typically, SRIR is carried out in two steps: extraction of meta-data and reconstruction of missing data. The former implies a series of computations on the input data in order to estimate values of metrics characterizing it. These metrics are used in the latter step (called the *super-resolution kernel*, SRK), which ends producing a high resolution (HR) image (called the *super-resolved image*, SRI).

The literature gives a number of possible SRIR algorithm classifications[9]. In this work we consider SRIR algorithms as belonging to one of the two following classes: fusion/restoration class, or synthesis class.

The fusion/restoration algorithms exploit the fact that temporarily close images of the same scene, due to warping and aliasing, can contain a slightly different representation of the same reality, hence providing additional information. In these algorithms a super-resolved image is a superposition of information extracted from a set of low resolution contents. Fusion algorithms can be further classified as iterative or non-iterative. Iterative algorithms estimate an initial image approximation that is refined with each iteration. The approximation with a sufficiently small error, or produced after a given number of iterations, is considered the SRI. The number of iterations determines the balance between quality and execution time for iterative algorithms. Non-iterative fusion algorithms are capable of producing a SRI in only one pass of the execution loop. For these algorithms the parameters that define the candidates set are the ones that specify the balance between quality and performance.

The synthesis algorithms create a HR version that, according to some previously acquired knowledge, is most likely to represent the real HR content. In order to carry out that task, the synthesis algorithms use machine learning and pattern matching techniques. The input LR content is divided into regions based on the extracted meta-data. Extracted metrics of regions of the input content are matched against the ones contained in the knowledge base. Each element contained in the knowledge base is formed by an identification pattern (or value) and the algorithm most suitable for synthesizing HR features for the pattern. The process of pattern matching ends with the identification of the patterns that represent the closest match for each of the regions. Super-resolved representation of each region is then approximated using the algorithm associated with the identification pattern found for the region. The quality of the outcome depends on the resulting match accuracy, and the quality and size of the knowledge base.

3. NON-UNIFORM GRID PROJECTION ALGORITHM

The *non-uniform grid projection algorithm* (NUGPA) is a fusion class algorithm that exploits warping and aliasing. The data used to enhance the LR input is extracted from a set of neighboring frames. This set of frames is denominated the *sliding frames window* (SFW), due to its update policy based on pre-load elements shifts, as presented in Figure 1. In other words after one frame processing is concluded, the least recently loaded frame is discarded, the remaining frames are shifted, and finally the next sequence frame is loaded. The implementation of SFW is used in order to provide dynamic super-resolution. Dynamic super-resolution requires that the number of super-resolved frames per unit time matches the number of input frames.

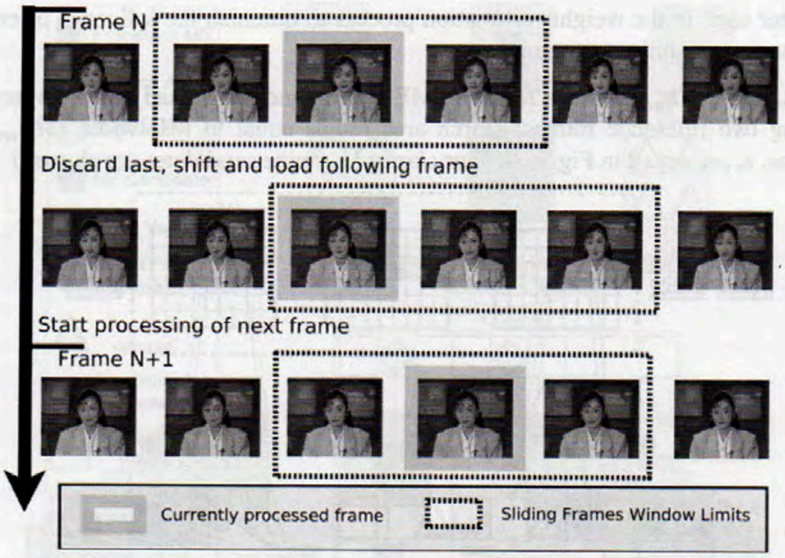
Frame N
Discard last, shift and load following frame
Start processing of next frame
Frame N+1
Currently processed frame
Sliding Frames Window Limits

Figure 1. Example of replacement scheme for a Sliding Frame Window comprising one previous and one following frame

The NUGPA carries out the super-resolution task in three steps: meta-data extraction, HR grid construction and interpolation. First, the warping function and its metrics are estimated. The warping function usually is not known a priori, hence, regions which could contain supplementary information have to be found at run time. What is known is that these regions are believed to differ only slightly from regions that they could enrich. In order to find regions that are most probable of containing additional information the NUGPA uses *motion estimation* (ME). During ME a set of neighboring pels (hereafter a *macro-block* (MB)) from the processed frame is matched against pels from other frames from the sliding window (hereafter *reference frames*). The matching criteria used is the *sum of absolute differences* (SAD) between the patches (hereafter SAD_{inter}). Considering a series of 'n' luminance values, representing a MB, as a lexicographically ordered vector a= $[a_1, a_2, \cdots, a_n]^T$, a SAD_{inter} value is computed as in (1), where $\mathrm{a_{curr_i}}$ represent the processed MB luminance values, and $\mathrm{a_{cand_i}}$ represent luminance values of a candidate MB from a reference frame.

$$\mathrm{SAD_{inter}} = \sum_{i=1}^{n} \left|\mathrm{a_{cand_i}} - \mathrm{a_{curr_i}}\right| \tag{1}$$

Only a limited set of pels confined within certain spatial proximity (called the search area radius, SAR; expressed in number of pels) participate in the ME. This set is called the *search area* (SA). The output of the ME process is a so called *motion vector* (MV). This vector identifies the MB for which SAD values are the lowest. The algorithm for each processed MB produces one MV per each reference frame.

Motion estimation can be carried out for an arbitrary MB size. The most common schemes work with, either, a square-shaped block of pels, or a single pel. The former is referred to as *block matching* (BM)[10] ME, whereas the latter as *optical flow*[8]. The size of MB significantly influences motion estimation accuracy, as well as the computational and memory requirements. Smaller MB size relaxes memory requirements, at the cost of higher computational complexity. Block matching is considered an effective scheme for textured regions, but exhibits performance degradation when carried out for regions with high homogeneity[8]. Nevertheless, due to lower performance requirements BM is more common in designs leading to hardware implementations. In order to prevent quality degradation, homogenous regions should be identified and handled differently. For this purpose, SAD between pels belonging to a MB and average pel value for this MB ($\bar{a}_{curr}$, computed as in (2)) is estimated (hereafter SAD_{intra}), in accord with (3).

$$\bar{a}_{curr} = \frac{\sum_{j=1}^{n} a_{curr_j}}{n} \tag{2}$$

$$SAD_{intra} = \sum_{i=1}^{n} \left|a_{curr_i} - \bar{a}_{curr}\right| \tag{3}$$

The SAD_{intra} value is later used in the weights estimation process to diminish the influence of error-prone homogenous regions matching on the super-resolution outcome.

Motion vectors, SAD_{inter} and SAD_{intra} values form the ME output, and are passed on to the next step. An example of BM for SFW comprising two reference frames, search area radius equal to MB width (MB_{width}), comprising $(2 \times MB_{width} + 1)^2$ candidates, is presented in Figure 2. (For clarity only nine candidates are shown.)

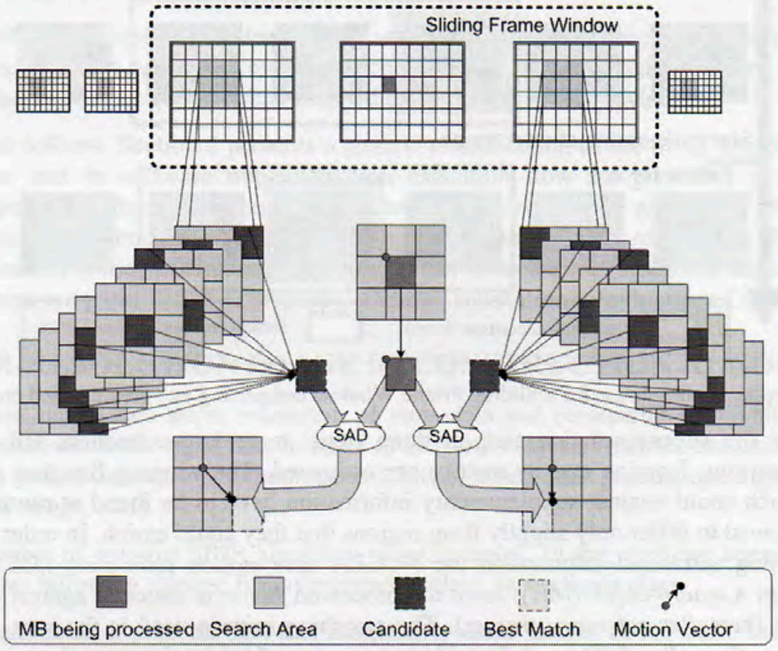

Figure 2. Execution flow of the block matching motion estimation

Next, the HR grid is created. The dimensions of this grid are determined by the ME precision (hereafter $precision_{me}$). First, the HR grid gets filled with LR pels of the frame being super-resolved. Having $f_{LR}(x, y)$ representing pixels of the LR frame, with spatial coordinates (x, y), the new HR spatial coordinates $(\hat{x}, \hat{y})$ are computed by multiplying the LR coordinates by the motion estimation precision, in accord with (4).

$$(\hat{x}, \hat{y}) = (x * precision_{me}, y * precision_{me}) \tag{4}$$

Having placed all LR pixels on the HR grid, pixels from a set of preceding and succeeding LR frames contained in the SFW are considered. At this point, most of the HR grid coordinates do not contain valid data. Those coordinates are referred to as *holes*, and are to be filled with data extracted from the SFW. *HolesFilling* is managed based on ME metrics. The MVs and SADs determine which data will contribute to which *hole* value estimation, and with what weight. It is allowed for more than one value to take part in the process of forming a new super-resolved value. In the case of more than one pel value contributing to the process, each contributing pel value is multiplied by its weight and added together to form an intermediate super-resolved value. For each coordinate, a sum of weights of the contributions is maintained and updated. Having considered all reference frames, the final *holes* values are formed by dividing the intermediate values by the sum of weights associated with them. The data extracted from reference frames can be used to modify only holes values. This guarantees that the beforehand placed LR values remain unchanged during the *HolesFilling* process. Data extraction and HR grid filling execution flow is presented in Figure 3.

A common scenario is that not all holes of the HR grid are filled during the *HolesFilling* process. Unfilled positions values are estimated by means of interpolation. Finally, the post-interpolation HR grid is adjusted to the expected outcome dimensions indicated by the scale factor (hereafter *scaleSR*). The scale determines the relation between the expected super-resolved outcome and the LR input dimension. For *scaleSR* values smaller than the $precision_{me}$ value, HR grid decimation is carried out. When the aforementioned parameters values are equal, no further processing is needed and the post-interpolation HR grid becomes the outcome of the NUGPA.

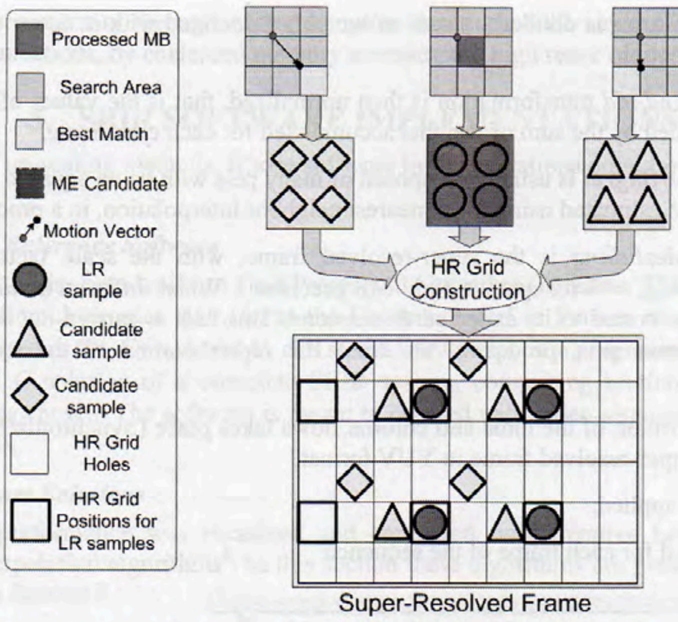

Figure 3. Overview of the Non-Uniform Grid Projection algorithm super-resolution kernel data flow

The presented processing is carried out only for luma (Y) component pels. Chroma pels carry a significantly lower energy concentration than luma pels, and have a significantly lower impact on the subjective quality of the outcome. Thus, chroma pels are not super-resolved, but up-scaled by means of bilinear interpolation.

4. SRIUMA SOFTWARE EXECUTION FLOW

In this section the execution flow of the SRIRiuma software is explained in detail. This section focuses on the presentation of the super resolution kernel (SRK). ME description is not presented as considered out of the scope of this paper.

The super-resolution kernel execution flow is presented in Figure 4. The luma and chroma components are to undergo different processing. Thus, when presented with YUV formatted LR input the software starts processing by separating luma from chroma pels. On the chroma pels a bilinear interpolation is applied, meaning that no additional information is restored. The luma pels undergo full NUGPA super resolution processing, as presented in Figure 4. The steps carried out by the super-resolution kernel are then as follows:

1. The pels of the frame to be processed are loaded from memory.
2. Chroma U and V components pels are separated from the luma pels and undergo bilinear interpolation and are stored in buffers.
3. Luma pels with value equal to "0" (called zeroes) are substituted with value "1", in a process called *Zeroes2ones* transformation.
4. Two HR representations of the transformed LR frame are created, namely the *holesGrid* and the *SRGrid*. Each of the HR representation dimensions is (for quarter pixel ME) 4 times greater than the LR one, resulting in overall size growth of 16. The *SRGrid* is constructed directly from the output of the zeroes2ones transformation. For *holesGrid*, pels belonging to the borders (outer rows and columns) of the zeroes2ones outcome are duplicated before the spatial augmentation takes place. Border duplication is needed to support more precise BM of objects that are moving out of the frame.
5. Steps 1–4 are repeated until all frames from the current SFW have their HR representations.
6. Candidate pels specified by the ME metrics are extracted from all *holesGrid* of frames belonging to the SFW. Extracted data are merged with the *SRgrid* of the frame being processed.

7. Each element of the *SRgrid* is divided by sum of weights associated with it. Steps 6–7 make up the so called *ShiftAndAdd* transformation.
8. Outcome of the *ShiftAndAdd* transformation is then normalized, that is the values of coordinates of the SRgrid after merging are divided by the sum of weights accumulated for each coordinate.
9. After normalization the *SRgrid* is usually composed of many pels with value equal to zero. Those pels are called holes. Holes values get estimated using mean nearest neighbor interpolation, in a process called *HolesFilling*.
10. The *SRgrid* after *HolesFilling* is the super-resolved frame, with the scale factor equal to ME precision ($scaleSR = precision_{me} = 4$ for quarter pixel ME precision). When different $scaleSR$ value is specified, the *SRgrid* has to be transformed to its expected dimensions. This task is carried out by the *Scale* function. This function discards certain pels, producing the final HR representation of the super-resolved of the luma component.
11. Finally, the synchronization of the luma and chroma flows takes place (SynchronizeSR). This step is needed in order to provide the super-resolved frame in YUV format.
12. SFW update policy is applied.
13. Steps 1–12 are repeated for each frame of the sequence.

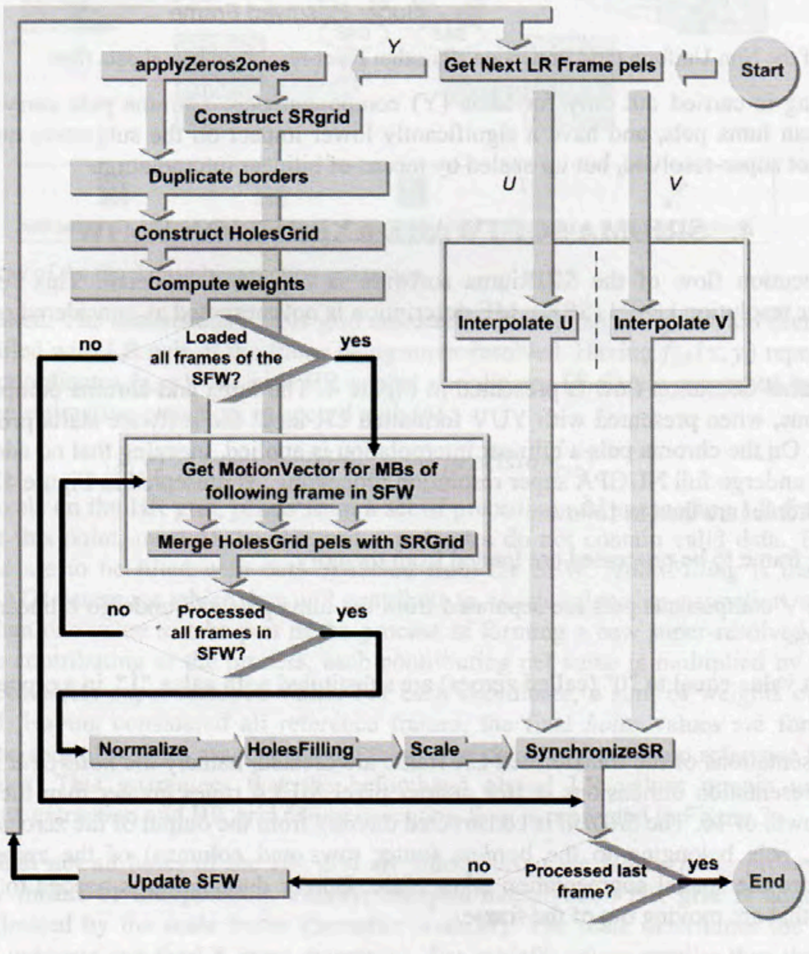

Figure 4. Execution flow of the frame level version of the super-resolution kernel

The presented processing is carried out at frame-level. Transitions from one step to another take place only after step's transformation has been applied on all frame pels. Frame-level processing requires that HR representations of all frames from SFW be available from memory throughout the complete duration of the super resolution process. This scheme minimizes memory communications, by coalesced memory accesses and high reuse of loaded data.

5. SRIR SOFTWARE IMPLEMENTATIONS

This section presents the up-scaling methods, whose software implementations output quality was to be evaluated and compared.

5.1 Super Resolution Reference Software

Software implementation of the Non-Uniform Grid Projection Algorithm (hereafter *SRIRiuma*) is being developed by the Institute for Applied Microelectronics (IUMA) at the University of Las Palmas de Gran Canaria (ULPGC). The software has been implemented in C#, C++, ANSI C, and several device specific programming environments[11].

For this work the ANSI C solution of a complete SRIR system, comprising motion estimation (ME) and super-resolution kernel (SRK), was chosen. The software is meant to be used with video sequences, and is capable of carrying out dynamic super-resolution.

5.2 Competitive Software Solutions

The SRIRiuma software performance was measured and compared with iterative back projection[6], robust super-resolution[7], and several interpolation algorithms[8]. In this section these algorithms are presented. Results of performance comparison are presented in Section 6.

Iterative Back Projection[6]: The IBP solves the restoration problem by gradual refinement of a HR approximation of the original LR image. In each iteration a LR representation ($\widehat{LR}$) of the HR approximation is derived for each frame of the SFW. The idea behind the IBP is to refine the HR approximation based on the accuracy of the $\widehat{LR}$ representation, in reference to the original LR frame. The $\widehat{LR}$ representations are matched against the original LR, giving a quantitative error (residual) of the HR representation. The measured residuals determine the values which are used to create a new refined HR approximation. If a change in HR approximation results in lower residual energy, the change is considered to have increased the accuracy of the HR approximation, and refinement in that direction is encouraged.

The IBP algorithm implementation[6] super-resolution execution flow starts with frame level motion estimation, producing inter-frame shifts and rotations. For each frame in the SFW, an HR approximation ($\widehat{HR_{SFW\iota}}$) is built by applying inverse ME transformation (rotations and shifts) on the approximation from the preceding iteration step. The first iteration approximation is created by means of nearest-neighbor interpolation. Then artificial blur is simulated using the (given *a priori*) *point spread function* (PSF). The representation is created by decimation of HR approximation. The representation is then subtracted from the original LR frame forming a residual ($\widehat{R_{SFW\iota}}$). The residual is interpolated, and undergoes the inverse PSF transformation, shifts and rotation.

Having estimated residuals for all SFW frames, a sum of residuals $\overline{R}_\iota$ is computed. The current iteration approximation ($\widehat{HR}_\iota$) is then computed, by subtraction of the weighted (multiplied by the convolution factor (λ) defining the gradient step size) sum of residuals from the preceding iteration approximation, as in (5).

$$\widehat{HR}_\iota = \widehat{HR_{\iota-1}} - \lambda\overline{R}_\iota \tag{5}$$

The presented IBP algorithm also differentiates the processing for luma and chroma pels. The former are super resolved, whereas on the latter interpolation is applied. In contrast with the SRIRiuma, that uses bilinear interpolation, the IBP software implementation uses bi-cubic interpolation for the chroma pels up-scaling.

Robust Super-Resolution[7]: The *RSR* is a variant of the IBP described above. The differences between these two variants are to be found in the process in which the residual is estimated. The RSR uses multiplied median instead of the sum used in the IBP. This helps to eliminate outliers at the cost of additional computations. The rest of the algorithm remains unchanged.

Interpolation[8]: Interpolation is a process in which a value of a HR pel is estimated based on its existing LR in-frame neighborhood. Interpolation can be either non-adaptive or adaptive. The former treats data as a series of raw values, and do not make run-time decisions on the algorithm to be used. The latter analyzes input, classifying pels as belonging to one of the possible types of regions (edge, non-edge etcetera). Based on the outcome of this pre-processing, different algorithms can be applied on pels belonging to different types of regions.

Bi-cubic, bilinear and nearest-neighbor interpolation are non-adaptive interpolation algorithms. The nearest-neighbor algorithm simply replicates a value of the nearest LR pel and does not take into account values of other neighboring points. This algorithm requiring the least processing power but also delivers the poorest quality of the outcome. The bilinear algorithm is commonly used in image processing, as it offers an attractive trade-off between output quality, and execution time. In this algorithm a neighborhood of 4 LR pels (forming a 2x2 2-dimensional grid) contributes to the process of HR pel value estimation. First a mean of the values of each row are computed. The HR value is estimated as a mean of the means values computed for the two rows. In bi-cubic algorithm a neighborhood of 16 LR pels (forming a 4x4 2-dimensional grid) is used for one HR value computation. The HR value is estimated based on normalized weighted sum of the pels from the neighborhood, where spatially closer values get assigned higher weights. One should bear in mind that no type of interpolation is capable of detail restoration.

6. EVALUATION OF SRIR SOFTWARE OUPUT QUALITY

The input format has been chosen to be the YUV 4:2:0p 8bit QCIF (144x176 pels) sequence. For the implementations used in tests the super-resolution scale factor value was set to 2, resulting in a CIF (288x352 pels) output. The simulations were carried out for the Mobile, Foreman, and the Paris sequences, in accord with the flow presented in Figure 5:

1. First the YUV 4:2:0p 8bit QCIF sequences were obtained from a CIF sequence. In order to do so, the CIF sequence was loaded, and decimated, forming a QCIF sequence that was then stored.
2. The QCIF sequences were used as input for tested algorithms, resulting in CIF YUV 4:2:0p 8bit output.
3. The output (HR_{sr}) luma pels (Y component) were compared with those of the original CIF sequence (HR_{ref}) resulting in a computation of Mean Squared Error (MSE), as shown in (6), where *m* and *n* represent the frame horizontal and vertical dimensions, respectively. The MSE value was then used in (7) to produce the Peak Signal-to-Noise Ratio (PSNR) for an 8 bit per pel representation.

$$MSE = \frac{1}{m * n} \sum_{x=1}^{n} \sum_{y=1}^{m} \left(\mathrm{HR}_{ref}(x,y) - \mathrm{HR}_{sr}(x,y) \right)^2 \tag{6}$$

$$PSNR = 10 * \log_{10} \frac{(255)^2}{MSE} \tag{7}$$

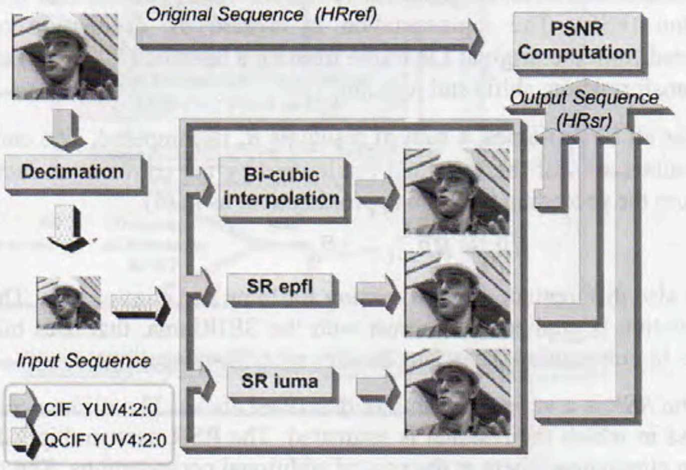

Figure 5. Experiments data flow

Execution of IBP and RSR algorithms was possible by using the "*SUPERRESOLUTION — Graphical User Interface for Super-Resolution Imaging*" application (hereafter *SOFTepfl*), developed by the Laboratory of Audiovisual Communications (LCAV) of the Ecole Polytechnique Federale de Lausanne (EPFL). The application is distributed as a set of functions[4] under the *GNU General Public License*. This type of license allows software modifications for test needs. In order to obtain experimental results the SOFTepfl was changed to support the chosen input/output format, and dynamic super-resolution. Modifications required addition of YUV formatted frames load/store, and Sliding Frames Window management code. The SOFTepfl internally transforms the loaded TIFF data (RGB 24bits format) into YCbCr format and applies the ME and SR kernel on the luma (Y) component. The chroma (Cb, and Cr) pels are not processed by the super-resolution kernel, and are just interpolated. To recap, no changes in the code or execution flow, of motion estimation and SR kernels were made.

Bi-cubic, bilinear, and nearest-neighbor interpolation[8] were carried out by means of high abstraction level function *imresize*[4], invoked with scale set as '2' and method specified as 'bicubic', 'bilinear' and 'nearest', respectively.

Quality Measurement Results: The SOFTepfl offers a range of ME techniques, all performing inter-frame motion estimation. In order to choose the best ME, preliminarily tests over 30 initial frames of the test sequences were carried out. Algorithms referenced in the application as "Vandewalle"[12], "Marcel"[13] "Lucchese"[14], and "Keren"[15] were tested. Results of the preliminary tests noted for the Mobile sequence are presented in Table 1. These results reflect a tendency found to be true for all tested sequences, and suggested the use of "Vandewalle" in main tests. The "Vandewalle" algorithm was chosen over "Keren" as it offers similar performance but suffers less from SFW increase.

Table 1. SOFTepfl motion estimation average PSNR for 30 initial frames of the Mobile sequence

Used	*Output PSNR [dB]*		
Algorithm	SFW=5	SFW=9	SFW=17
Vandewalle	15.4992	**13.0352**	**8.1364**
Marcel	13.2591	8.6469	6.6840
Lucchese	13.9763	12.7489	7.9305
Keren	**15.5840**	12.6789	6.7709

Having chosen the ME algorithm, software solutions performance was tested. The results for *SRiuma* (macro-block (MB) width=4, SFW=5, search area radius=16, and scale=2), SOFTepfl (SFW=5 and interpolation scale=2), and interpolation are presented in Table 2.

The experiments showed that the SRIRiuma offers higher PSNR values for all tested sequences. The main reasons for such a difference are the use of the BM ME, and the non-iterative nature of the implemented algorithm. These characteristics lead to higher PSNR. The impact of the ME algorithm manifests itself especially in results for sequences with abundance of local movement like Foreman and Mobile. Those movements cannot be traced accurately using frame-level ME. For those sequences the PSNR obtained using frame-level estimation is significantly lower.

Table 2. Performance of tested up-scaling software implementations, expressed in psnr computed for luma component for 90 initial frames of the test sequences

Used Algorithm	*Test sequence average PSNR [dB]*		
	Foreman	Mobile	Paris
SRIRiuma	**29.42**	**22.38**	**22.79**
IBP	20.76	15.68	18.14
RSR	23.98	15.98	17.99
SRIR[2] iterative weighted with 20 iterations limit	27.80	20.74	22.22
Bilinear interpolation	28.07	20.46	21.23
Nearest-neighbor interpolation	26.68	18.78	19.89
Bi-cubic interpolation	28.09	20.29	20.92

7. CONCLUSIONS

In this work we presented an alternative that allows to mitigate the impact of the high resources requirements of the SRIR and to reach satisfactory output quality by SRIR hardware implementations. Our approach is based on using a non-iterative algorithm that facilitates hardware implementation and that is capable of providing state-of-the-art output quality. First a general classification of SRIR algorithms was presented. Then details on the non-iterative non-uniform grid projection algorithm and its software implementation execution flow were given. Following, competitive implementations used as benchmarks and their test set-up were shortly described.

Performed tests showed that the chosen algorithm using block matching ME is capable of pròviding higher output quality than competitive iterative SRIR (IBP, RSR; using frame-level ME) and interpolation (bi-cubic, bilinear and nearest-neighbor) algorithms for all sequences used in the tests (Foreman, Mobile, and Paris). Combined with its non-iterative nature, the algorithm promises efficient hardware implementation without loss in output quality, effectively reducing the current gap in output quality between hardware and software implementations.

8. ACKNOWLEDGMENTS

This work has been supported by the Spanish Ministry of Science and Technology under Project PccMute grant TEC2009-14672-C02-02. The authors of this work would like to thank Dr. Christos-Savvas Bouganis from Imperial College London for his help in making this work more complete by providing PSNR values for comparison.

REFERENCES

[1] Callico, G. M., Llopis, R. P., Lopez, S., Lopez, J. F., Nunez, A., Sethuraman, R. and Sarmiento, R., "Low-cost super-resolution algorithms implementation over a hw/sw video compression platform, user's guide and reference manual," EURASIP Journal on Applied Signal Processing, vol. 2006, pp. 1–29 (2006).

[2] Bowen, O. and Bouganis, C.-S., "Real-time image super resolution using an FPGA," in Field Programmable Logic and Applications, International Conference, pp. 89–94 (2008).

[3] Barreto, D., Callico, G. M., Lopez, S., García, L. and Nunez, A., "Real-time super-resolution over raw video sequences," in VLSI Curcuits and Systems II, vol. 5837, SPIE, Bellingham, WA: SPIE, pp. 628-637 (2005).

[4] The MathWorks, Inc., "MATLAB Language Reference," 1st ed., The MathWorks, Inc., 24 Prime Park Way, Natick, MA, (1996).

[5] Angelopoulou, M. E., Bouganis, C.-S., Cheung, P. K. and Constantinides, G. A., "FPGA-based real-time super-resolution on an adaptive image sensor," in Reconfigurable Computing: Architectures, Tools and Applications, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, pp. 125–136 (2008).

[6] Irani, M. and Peleg, S., "Improving resolution by image registration," CVGIP: Graph. Models Image Process, vol. 53, no. 3, pp. 231–239 (1991).

[7] Zomet, A., Rav-Acha, A. and Peleg, S., "Robust super-resolution," vol. 1, pp. I–645–I–650 (2001).

[8] Gonzalez, R. C. and Woods, R. E., [Digital Image Processing], 3rd Edition, Prentice Hall, Inc., pp. 65–68 (2008).

[9] Park, S. C., Park, M. K. and Kang, M.G., "Super-resolution image reconstruction: a technical overview," IEEE Signal Processing Magazine, vol. 20, no. 3, pp. 21-36 (2003).

[10] Callico, G. M., Lopez, S., Sosa, O., Lopez, J. and Sarmiento, R., "Analysis of fast block matching motion estimation algorithms for video super-resolution systems," Consumer Electronics, IEEE Transactions, vol. 54, no. 3, pp. 1430–1438 (2008).

[11] Lopez, S., Callico, G., Tobajas, F., Lopez, J. and Sarmiento, R., "A novel real-time DSP-based video super-resolution system," Consumer Electronics, IEEE Transactions, vol. 55, no. 4, pp. 2264–2270 (2009).

[12] Vandewalle, P., Susstrunk, S. and Vetterll, M., "A frequency domain approach to registration of aliased images with application to super-resolution," Eurasip Journal on Applied Signal Processing, pp. 1–14 (2006).

[13] Marcel, B., Briot, M. and Murrieta, R. "Calcul de translation et rotation par la transformation de fourier," Traitement du Signal, 14(2):135–149 (2006).

[14] Lucchese, L. and Cortelazzo, G. M. "A noise-robust frequency domain technique for estimating planar roto-translations," Signal Processing, IEEE Transactions, 48(6):1769–1786 (2000).

[15] Keren, D., Peleg, S. and Brada, R., "Image sequence enhancement using sub-pixel displacements," Computer Vision and Pattern Recognition, Proceedings CVPR '88, Computer Society Conference, pp. 742–746 (1988).

Author Index

Numbers in the index correspond to the last two digits of the six-digit citation identifier (CID) article numbering system used in Proceedings of SPIE. The first four digits reflect the volume number. Base 36 numbering is employed for the last two digits and indicates the order of articles within the volume. Numbers start with 00, 01, 02, 03, 04, 05, 06, 07, 08, 09, 0A, 0B … 0Z, followed by 10-1Z, 20-2Z, etc.