A DSP based SVC IP STB using open SVC decoder

Fernando Pescador, David Samper, Matias Garrido, Eduardo Juarez, Médéric Blestel

A DSP based SVC IP STB using Open SVC Decoder

F. Pescador, D. Samper, M.J. Garrido and E. Juarez
Grupo de Diseño Electrónico y Microelectrónico (GDEM)
Universidad Politécnica de Madrid.
Madrid, Spain.
{pescador, dsamper, matias, ejuarez}@sec.upm.es

M. Blestel
IETR/Image Group Lab
UMR CNRS 6164/INSA
Rennes, France
mblestel@insa-rennes.fr

Abstract— **In this paper, a implementation of a DSP-based IP set-top box (IP-STB) to decode CIF sequences compliant with the new Scalable Video Coding standard (14496-10 Amd 3) using Open SVC Decoder (OSD) is presented. The OSD software, designed for the PC environment, has been integrated into a previously developed IP-STB prototype. About 15 CIF frames per second can be decoded with the IP-STB.**

Set-top box; Scalable Video Coding; Open SVC Decoder; DSP

I. INTRODUCTION

In the last years, an increase in the deployment of all kinds of telecommunication networks (cable, satellite and terrestrial) arose in many parts of the world. These networks support multimedia services and applications such as digital TV, videoconferencing, video surveillance and Internet access among others. In this context, the consumer multimedia terminals play a central role. In these terminals, video management is the most demanding task in terms of computational power. Nevertheless, different terminals can have different computational power or display capabilities.

Scalable Video Coding (SVC) techniques [1] allow multimedia terminals to accommodate the spatial and temporal resolutions and the quality of a decoded video sequence to the available hardware/software resources. SVC techniques have been defined in most video coding standards [2][3][4], but their use has not become widespread because of their jitter problems and poor efficiency [1]. However, the SVC features recently included in H.264 [5] surpasses those used in former standards and facilitate new possibilities. Up to now, the available SVC encoder/decoder implementations are restricted to the PC domain [6][7][8]. One of the SVC decoder implementations is Open SVC Decoder (OSD) [8]. This decoder is written in C language and implements the baseline profile.

On the other hand, the new multimedia Digital Signal Processors (DSPs) [9] [10] allow the implementation of terminals supporting a variety of video coding standards at relatively low cost. In our previous work [11] [12] [13], MPEG-2, MPEG-4 and H.264 decoders have been ported and optimized for different DSPs and a methodology to implement real-time video decoders has been extracted from this previous experience [13].

The methodology is based on porting a C code from PC to the DSP environment, testing the standard conformance of the DSP-based decoder, optimizing the C code to improve the performance and integrating the optimized decoder in a video decoding system like an IP-Set Top Box (IP-STB). This methodology has been applied to port the OSD decoder to a DSP-based system.

In this paper, the implementation of a DSP-based SVC IP-STB using OSD is explained. The SVC and OSD software are outlined in section II. In section III, the previously developed IP-STB is introduced for reference. Section IV summarizes the OSD porting process. The tests and performance results are presented in section V. Finally, section VI concludes the paper.

II. SCALABLE VIDEO CODING AND OPEN SVC DECODER

A. H.264/SVC standard

In October 2007, a new SVC algorithm was standardized as ISO/IEC 14496-10 Amd 3 [5] [14]. As a part of the standardization effort, the Joint Scalable Video Model (JSVM) reference software [6] has been developed as well.

In this standard, the video compression is performed by generating a unique hierarchical bit-stream structured in several levels or layers of information, consisting of a base layer and several enhancement layers. The base layer provides basic quality. The enhancement layers provide improved quality at increased computational cost. Three types of scalabilities: spatial, temporal and quality are specified in the standard.

In a temporally scalable video sequence, several frame rates (temporal layers) of a video sequence can be chosen when decoding. Fig. 1 shows an example of a Group of Pictures (GOP) where the user can select three plausible frame rates. If the device decodes the four frames of the GOP (I1, B1, B2, B3), a full-frame-rate sequence are obtained. If the decoder discards B1 and B3 frames and only decodes I1 and B2, a half-frame-rate sequence is achieved. The third case is a quarter-frame-rate sequence, which is obtained when the decoder discards B1, B2 and B3 frames and only decodes I1.

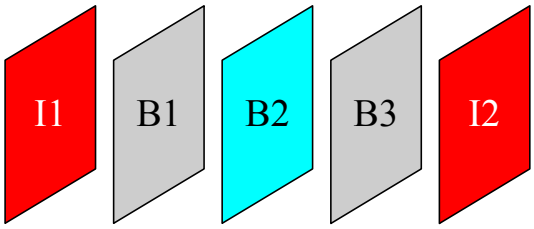

Fig. 1. Example of a GOP in a temporally scalable bit-stream.

This work was supported by the Spanish Ministry of Science and Technology under grants TEC2006-13599-C02-01 and TEC2009-14672-C02-01.

In a spatially scalable video sequence, several spatial resolutions (spatial layers) of the video frames can be chosen when decoding. Fig. 2 depicts an example of a spatial scalable bit-stream containing three possible resolutions. As can be seen, the information relating to each resolution of a frame is contained in the field reserved for such frame in the bit-stream.

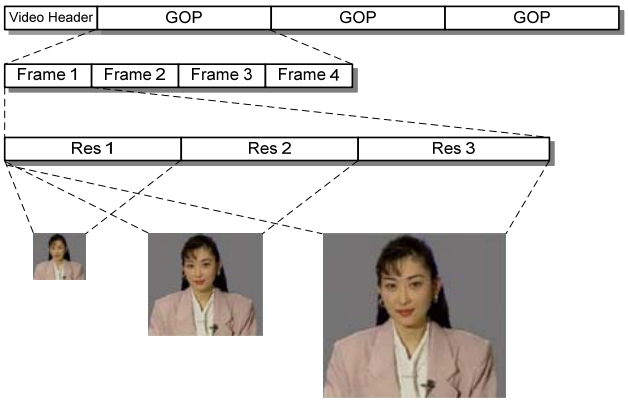

Fig. 2. Example of a spatially scalable bit-stream.

In a quality-scalable video sequence (or SNR sequence), it is possible to select several quality levels (quality layers) when decoding. Fig. 3 shows an example of a quality scalable bit-stream with three types of quality. The information relating to the three qualities of a frame is contained in the space reserved for this frame in the bit-stream.

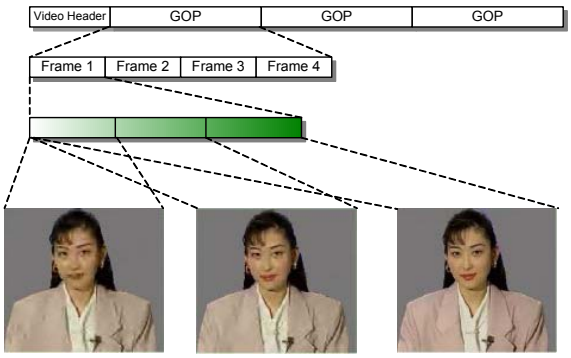

Fig. 3. Example of a quality-scalable (SNR) bit-stream.

Finally, the three types of scalability specified in H.264/SVC can be combined into a single bit-stream.

B. Open SVC Decoder

IETR has developed the Open SVC Decoder [15], a C language Scalable baseline profile decoder supporting all tools to deal with spatial, temporal and quality scalabilities. It is based on a fully compliant H.264 baseline decoder with most of the tools of the main profile. In the Scalable baseline profile the base layer has to be conformant with AVC baseline profile. In this profile, contrary to quality and temporal scalability which are supported without any restriction, the spatial scalable coding is restricted to 1.5 and 2 resolutions ratios between two successive spatial layers.

This work was supported by the Spanish Ministry of Science and Technology under grants TEC2006-13599-C02-01 and TEC2009-14672-C02-01.

The Open SVC Decoder has been developed in the framework of Scalim@ges [16] project. This project aimed to promote SVC standard in order to reduce the number of formats manipulated in production, distribution, and use of video compatible with existing solutions. Currently, others French and international projects like SVC4QoE [17] and ScalNet [18] are using Open SVC Decoder. These projects try to focus on the impacts and needed applications on network in order to manage efficiently the SVC technology and to improve the end user quality of experience.

The Open SVC Decoder, contrary to the JSVM which decodes all layers in a bit-stream, can decode a specific layer with a specific temporal scalability. This particularity provides an adaptability of the decoder over different platforms by selecting the right layer in order to have a real-time decoding. The changing of layer can be also done during the decoding process when a missing enhancement occurs due to transmission errors. Fig. 4 shows the data flow graph of the decoder when the top layer of a four layers stream is not decoded. Variable Length Coding and Texture Decoding are processes for the first three layers but not for the fourth.

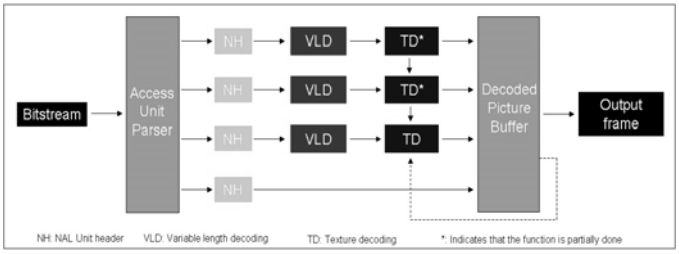

Fig. 4. Open SVC Decoder data flow graph.

In Fig. 5, a simplified flow diagram of the decoding process for a H.264 compliant stream is shown. The decoder reads the H.264 stream from an input buffer and decodes the NAL units in sequence. After decoding the NAL header, the NAL unit content is identified as a slice header or another syntax element (e.g. an SPS or a PPS). When the NAL unit contains an interesting slice for the selected layer, the decoder extracts all the syntactical elements from the stream and stores them in an intermediate buffer. If the processed NAL must be displayed, each macroblock (MB) is completely decoded, however, if the NAL must not be displayed the MB is partially decoded.

In the next step, if a frame has been completely decoded, the deblocking filter is applied. Finally, the decoded pictures are stored in images buffers and presented in the right order using the PC Simple Direct Media Layer (SDL) library [19].

The Open SVC Decoder has been compared to the JSVM 9.16 to benchmark and to test the conformance of the decoder. Table I shows the results of the speeding up between both decoders on several conformance sequences. The performance of the OSD is up to 50 times faster than the JVSM decoder [6], making this decoder a good starting point in the development of a new SVC decoder.

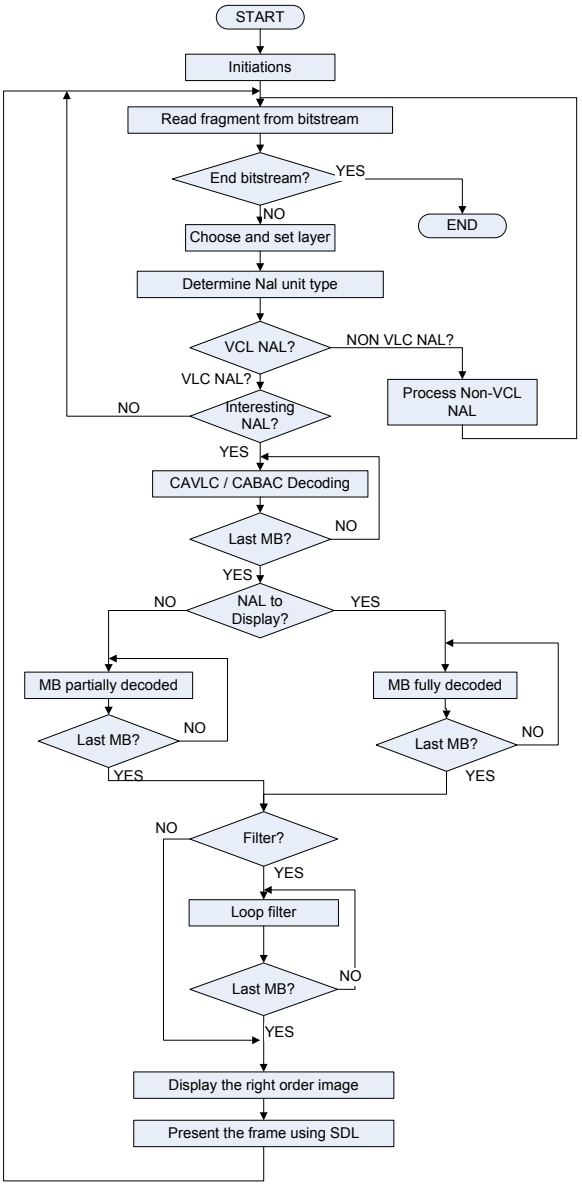

Fig. 5. Simplified Open SVC Decoder flow chart.

TABLE I JSVM AND OPEN SVC DECODER COMPARISON.

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

III. IP-STB ARCHITECTURE

The IP-STB [11] has been implemented in a commercial prototyping board [20] based on the TMS320DM6437 DSP [21].

A. TMS320DM6437 Architecture

This DSP is based on a third-generation high-performance VLIW architecture. A simplified block diagram is shown in Fig. 6. A fixed-point core with internal memory (L1D, L1P and L2) and an internal DMA (IDMA) is used. A switched central resource interconnects the core with a set of standard peripherals and a video processing subsystem.

The peripheral set includes, among others: an Ethernet MAC (EMAC), an I2C Bus interface, two multichannel buffered serial ports (McBSPs), a multichannel audio serial port (McASP), two 64-bit general-purpose timers, two UARTs, a PCI interface, two external memory interfaces (EMIF) and an EDMA controller (EDMA3) to handle the data transfers.

The video processing subsystem includes a Video Processing Front-End (VPFE) for video capture, and a Video Processing Back-End (VPBE) for video display.

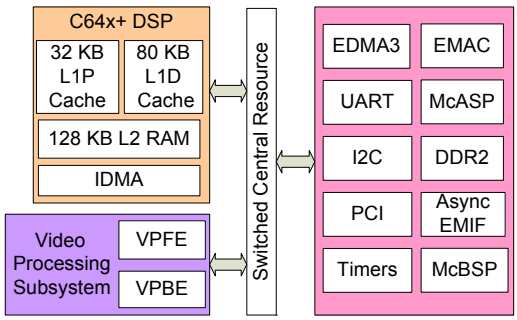

Fig. 6. Internal architecture of the new DSP.

B. Hardware Architecture

A commercial prototyping board (Fig. 7) based on the TMS320DM6437 DSP has been used to implement the IP-STB. The board has a DSP working at 600 MHz, 128 MB of SDRAM external memory, 80 MB of Flash external memory and several interfaces.

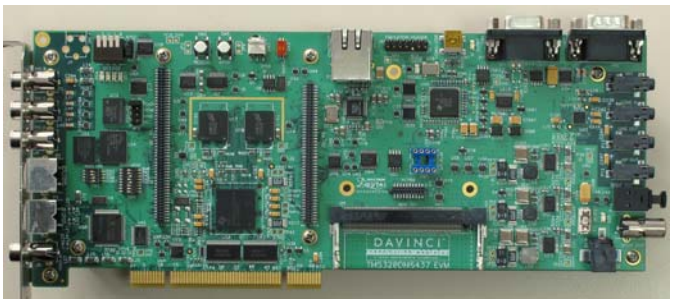

Fig. 7. The DSP-based development board.

C. Software Architecture

The IP-STB has been implemented over the board referenced in the previous subsection using a multi-task architecture (see Fig. 8). The *transport* task reads an MPEG-2 Transport Stream (MP2TS) containing the program through an Ethernet port (EMAC) and extracts the audio and video streams. The *video dec* task decodes the video stream and the

video play task sends the decoded pictures to a video port. The audio stream is decoded and played in a similar way. The *application* task is a very simple user interface. A real-time kernel schedules the tasks execution and allows inter-tasks communication.

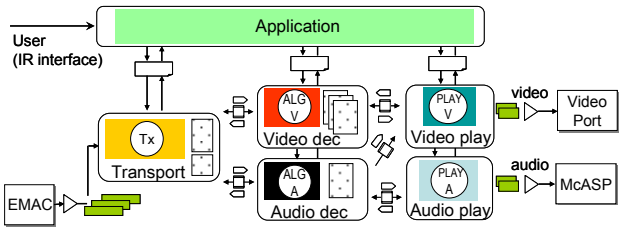

Fig. 8. IP-STB software architecture.

IV. Code Porting and Optimization

The OSD has been integrated into the DSP based IP-STB explained in section III. The integration process has been carried out in two steps. First, the decoder has been ported to the DSP. Second, the decoder has been integrated into the IP-STB software architecture. Next subsections describe the changes in the decoder during the porting process.

A. Porting of the Open SVC decoder to the DSP

The OSD has been ported to the DSP as follows:

- The decoder has been encapsulated into a DSP-BIOS [22] task. The size of the stack associated to this task has been adjusted to 1 MB and has been allocated in external memory.
- Code and data have been allocated in external memory. To limit the amount of memory of the DSP implementation, the maximum size of the decoded pictures has been reduced from HD (1920×1080) to SD (720×576).
- Internal memory has been configured as follows: L1D is divided in 32 KB for cache memory and 48 KB for general purpose data; L1P is configured as a 32 KB cache program memory and L2 is splitted between level-2 cache memory and general purpose memory. Currently, neither the code nor data are allocated in internal memory but these memories have available space for future optimizations.
- The decoder output interface has been modified. In the original code, the decoded pictures are displayed on screen using the SDL library as is shown in the Fig. 4. In the DSP code, the decoded pictures are written in a YUV file instead of being presented.
- Functions used to access the stream files have been adapted to the functions available in the DSP real-time support libraries.
- The way to select the layer to be decoded has been modified. In the original code, the layer was selected using the command line arguments while in the DSP version these parameters are introduced through a configuration file that is parsed at the begging of the decoding process.

B. OSD integration in the IP-STB

The decoder has been embedded into the IP-STB *video dec* task (see Fig. 8). To do so, several changes have been done in the IP-STB tasks:

- The *transport* task has been updated to support the NAL types used in SVC.
- The decoder initialization has been adapted so as the decoder can be embedded in a multi-task architecture.
- The decoder reads the video stream from a memory buffer, shared with the *transport* task, instead of a file. Moreover, the output pictures are written to a buffer, shared with the *Video play* task, instead of a file.

V. Tests and Performance Results

A set of tests has been carried out to verify the decoder and to measure its performance. Three types of tests have been carried out: tests of the decoder using the simulator, tests of the decoder using the prototyping board and tests with the IP-STB in an actual environment.

A. Tests of the decoder using the simulator

The decoder performance has been measured in simulation with CCS [23] using the sequences described in Table I. These sequences are the same that were used in [15] to test the performance of the OSD in a PC implementation.

The testbench of the ported OSD to the DSP is shown in Fig. 9. The decoding task reads a portion of the SVC stream from a file and internally stores it in a stream buffer. The decoding process reads its input data from this buffer and stores the decoded data in a picture buffer. The process is repeated until the SVC stream ends. A reconstructed YUV file is written each time a full picture is decoded.

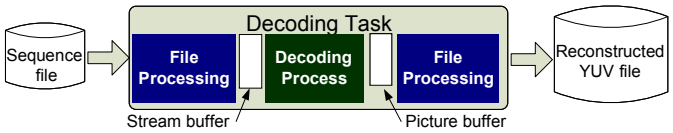

Fig. 9. DSP-based Open SVC decoder testbench.

Table II contains the profiling results for the decoder in simulation. The columns *layer* 0 and *layer* 1 show the average number of clock cycles needed to decode a picture from a layer of the SVC stream.

TABLE II OpenSVC decoder benchmark.

sequence	*Layer 0*	*Layer 1*
SVCBST-1	6895906 (360x240)	49736106 (720x480)
SVCBST-2	6899799 (360x240)	49756945 (720x480)
SVCBST-14	23649626 (640x360)	Not decoded (HD)
SVCBST-15	4457216 (320x180)	27099036 (640x360)

B. Tests of the decoder using the prototyping board

In these real-time tests several complex sequences have been decoded to measure the decoder performance extracting the different layers included in each stream. A commercial SVC encoder [24] has been used to generate the streams.

The tests have been carried out with the testbench presented in Fig. 9 but using a TMS320DM6437 based prototyping board [20] instead of the DSP simulator.

The decoder performance has been measured in real-time using a DSP internal timer. The timer is captured both at the beginning and at the end of a Network Abstraction Layer (NAL) decoding process. The difference between the two data is the time spent by the decoder for this NAL unit.

Nine sequences have been generated to test the decoder in real-time. The spatial resolutions used are CIF, ½ CIF (272x224) and QCIF; the temporal resolutions layers have 20 fps, 10 fps and 5 fps; finally, the quality layers have been generated using different bit rates, with the layers corresponding to a 100%, 50% and 25% of the bit rate associated to each sequence.

The encoded sequences were generated using the well-known video sequence "mobile" with YUV 4:2:0 format and 200 frames. Some encoding parameters are the same for all of the generated sequences:

- GOP size of 16 progressive frames.
- Constant output bit rate.
- Entropy encoding using CABAC.
- Deblocking filter activated.
- One B frame for each P-frame (except in the sequences with a quarter of temporal resolution, which need 3 B-frames for each I- o P-frames)
- Intra- and inter-prediction with all possible partitions for the macroblocks.

The sequences F#1, F#2 and F#3, with three layers each (L0, L1 and L2), has been generated to test the quality and spatial scalabilities simultaneously. The temporal resolution is the same for the three sequences (20 fps). The main features of these sequences are presented in Table III.

TABLE III STREAMS GENERATED FOR 20 FPS OF TEMPORAL RESOLUTION.

	Base Layer (L0)		Enh. Layer 1 (L1)		Enh. Layer 2 (L2)	
	Size	Bit-rate (Kbps)	Size	Bit-rate (Kbps)	Size	Bit-rate (Kbps)
F#1	QCIF	64	QCIF	128	QCIF	256
F#2	½ CIF	84	½ CIF	175	½ CIF	350
F#3	CIF	128	CIF	256	CIF	512

The sequences F#4, F#5 and F#6 with three layers each (L0, L1 and L2) has been generated to test the temporal and spatial scalabilities at the same time. The quality of the layers is the maximum for the three sequences. The features of the generated sequences are presented in Table IV.

TABLE IV STREAMS GENERATED BY FIXING THE QUALITY OF THE SEQUENCES.

	Base Layer (L0)		Enh. Layer 1 (L1)		Enh. Layer 2 (L2)	
	Size	fps	Size	fps	Size	fps
F#4	QCIF	5	QCIF	10	QCIF	20
F#5	½ CIF	5	½ CIF	10	½ CIF	20
F#6	CIF	5	CIF	10	CIF	20

Finally, three sequences have been generated (F#7, F#8 and F#9) with three layers each (L0, L1 and L2) to test the temporal and quality scalabilities simultaneously. The characteristics of the encoded sequences are presented in Table V.

TABLE V STREAMS GENERATED WITH SPATIAL RESOLUTION FIXED TO ½ CIF.

	Base Layer (L0)		Enh. Layer 1 (L1)		Enh. Layer 2 (L2)	
	Bitrate (Kbps)	fps	Bitrate (Kbps)	fps	Bitrate (Kbps)	fps
F#7	64	5	128	10	256	20
F#8	84	5	175	10	350	20
F#9	128	5	256	10	512	20

The computational load needed to decode each layer of the example sequences has been measured. In Table VI the percentage of CPU load spent by the decoder is given.

TABLE VI CPU COMPUTATIONAL LOAD FOR TEST SEQUENCES.

	Base Layer (L0)	Enh. Layer 1 (L1)	Enh. Layer 2 (L2)
F#1	11 %	24 %	35 %
F#2	24 %	53 %	75 %
F#3[1]	39 %	88 %	130 %
F#4	9 %	16 %	29 %
F#5	16 %	32 %	60 %
F#6	26 %	51 %	98 %
F#7	13 %	27 %	52 %
F#8	14 %	28 %	54 %
F#9	16 %	32 %	60 %

C. Tests with the IP-STB in an actual environment

Functional and performance tests have been carried out with the testbench shown in Fig. 10. Actual movies are played with a SD DVD, whose analogue output is connected to a PC based video capture board [25]. The PC runs a commercial SVC encoder [24], generating an MP2TS with the SVC stream and an AC3 audio stream. The MP2TS is transmitted over an Ethernet network using a multicast IP address.

The IP-STB receives the MP2TS stream, extracts audio and video elementary streams, decode both and display the results in a standard TV set. The IP-STB real-time tests have been carried out using actual DVD contents.

[1] The percentage of CPU associated with Enhancement Layer 2 of sequence F#3 is greater than 100% so no real-time processing is achieved for this layer.

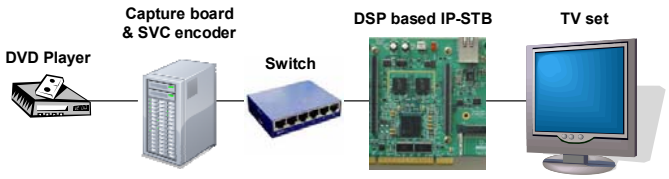

Fig. 10. IP-STB testbench in an actual environment

In Table VII, the measured decoder performance using an actual soccer match is shown. Four sequences with spatial and quality scalability types have been encoded using a GOP of 10 frames. Two of them use CABAC with 3 P-frames and 6 B-frames in each GOP and the other two sequences use CAVLC without B-frames. All the sequences have 1500 frames. The *cycles* column in *layer 0* and *layer 1* shows the average number of clock cycles ($\times 10^6$) needed to decode a picture from a layer of the SVC sequence.

TABLE VII DECODER REAL-TIME BENCHMARK.

Sequence	Layer 0 (Base layer)			Layer 1 (Enhancement layer)		
	cycles	size	Kbps	cycles	size	Kbps
Spatial CAVLC	3.8	QCIF	104	18.7	CIF	512
Quality CAVLC	10.7	CIF	256	25.5	CIF	512
Spatial CABAC	5.1	QCIF	109	23.9	CIF	512
Quality CABAC	15.2	CIF	256	30.7	CIF	512

Table VIII shows the average CPU clock cycles ($\times 10^6$) per frame used by the IP-STB *video dec* task and the frames per second (fps) that can be decoded with the DSP at 600 MHz. The overhead of the other IP-STB tasks is less than 4×10^6 CPU clock cycles per frame. The test sequences are the same as those referenced in Table VI.

TABLE VIII TASKS PERFORMANCE.

	Spatial CAV.		Quality CAV.		Spatial CAB.		Quality CAB	
	Lay0	Lay1	Lay0	Lay1	Lay0	Lay1	Lay0	Lay1
Video dec	4.5	22.3	11.9	30.4	7.7	28.7	18.9	36.5
fps	75.9	23.6	39.7	17.7	56.0	18.8	27.0	15.0

VI. CONCLUSIONS AND FUTURE WORK

The Open SVC Decoder has been ported to a latest generation DSP and has been integrated into an IP-STB prototype with a performance of 15 CIF fps. Now our previous experience [13] [26] is currently been applied to the optimization of this decoder. Extrapolating the results we have obtained for other decoders, real-time operation for SD pictures can be forecasted.

ACKNOWLEDGMENT

The authors would like to thank Ernesto Seisdedos from Grupo de Diseño Electrónico y Microelectrónico (UPM) and Mickaël Raulet from IETR/Image Group Lab for their contributions to this work.

REFERENCES

[1] J-R Ohm, "Advances in Scalable Video Coding". *Proceedings of the IEEE*, vol. 93, nº 1 pp. 42-56, Jan. 2005.
[2] ISO/IEC 13818-2 (ITU-T Rec. H.262). Generic coding of moving pictures and associated audio information: Video. 1995.
[3] ISO/IEC 14496-2. Information technology. Coding of audio visual objects. Part 2: Video. 1998.
[4] ISO/IEC 14496-10. Information technology. Coding of audio-visual objects. Part 10: Advanced Video Coding. Dec. 2005.
[5] Joint ITU-T Rec. H.264 | ISO/IEC 14496-10 / Amd.3 Scalable Video Coding, November 2007.
[6] Joint Scalable Video Model JSVM-9.9, Available in CVS repository at Rheinisch-Westfälische Technische Hochschule (RWTH) Aachen.
[7] IMEC. http://www2.imec.be/be_en/press/imec-news/archive-2008/imec-speeds-up-scalable-video-decoder-svc-with-factor-20.html
[8] Open SVC Dec. http://sourceforge.net/projects/opensvcdecoder
[9] Texas Instruments. TMS320DM642 Video/Imaging Fixed-Point Digital Signal Processor. http://focus.ti.com/docs/prod/folders/print/tms320dm642.html
[10] ADSP-BF533 High Performance General Purpose Blackfin Processor. http://www.analog.com/en/epProd/0,,ADSP-BF533,00.html
[11] F. Pescador, C. Sanz, M.J. Garrido, C. Santos y R. Antoniello. "A DSP based IP set-top box for home entertainment". IEEE Trans. on Consumer Electronics Vol. 52, Issue 1, Feb. 2006 pp. 254-262.
[12] F. Pescador, M. J. Garrido, C. Sanz, E. Juárez, D. Samper and R. Antoniello. "MPEG-4 SP/ASP decoder for a DSP-based Multi-Format IP Set-Top Box". Annual Conference of the IEEE Industrial Electronics Society (IECON06). Paris (Francia).2006.
[13] F. Pescador, C. Sanz, M.J. Garrido, E. Juárez y D. Samper. "A DSP Based H.264 Decoder for a Multi-Format IP Set-Top Box". IEEE Trans. on Consumer Electronics Vol. 54, Issue 1, Feb. 2008 pp. 145-153
[14] H. Schwarz, D.Marple and T.Wiegand. "Overview of the Scalable Video Coding Extension of the H.264/AVC Standard". IEEE Trans. on Circuits and Systems for Video Tech. Vol 17, Nº9, pp 1003-1120. Sept 2007
[15] M. Pelcat, M. Blestel, M. Raulet. "From AVC decoder to SVC: Minor impact on a data flow graph description" PCS2007. June 2007.
[16] Scalim@ges project: http://www.images-et-reseaux.com/en/les-projets/fiche-projets-finances.php?id=125
[17] SVC4QoE: http://www.images-et-reseaux.com/en/les-projets/fiche-projets-finances.php?id=203
[18] ScalNet: http://www.scalnet.info/system/web/default.aspx
[19] Simple Direct Media Layer (SDL). http://www.libsdl.org/
[20] DM6437 Digital Video Development Platform. http://www.spectrumdigital.com/product_info.php?cPath=37&products_id=196.
[21] Texas Instruments. DaVinci DSPs. http://focus.ti.com/docs/prod/folders/ print/tms320dm6437.html
[22] Texas Instruments. "TMS320 DSP-BIOS User's guide" (SPRU303B – May. 2000). http://focus.ti.com/lit/ug/spru303b/spru303b.pdf.
[23] Code Composer Studio. http://focus.ti.com/dsp/docs/dspsupportatn.tsp?sectionId=3&tabId=415&familyId=44&toolTypeId=3
[24] Mainconcept Scalable Video Coding. http://www.mainconcept.com/site/developer-products-6/pc-based-sdks-20974/svc-tech-preview-22033/information-22036.html.
[25] Decklink Studio. http://www.decklink.com/products/decklink/
[26] F. Pescador, G. Maturana, M.J. Garrido, E. Juárez y C. Sanz. "An H.264 video decoder based on a latest generation DSP". IEEE Trans. on Consumer Electronics Vol. 55, Issue 1, Feb. 2009 pp. 145-153.