A DSP Based Multi-Format Video Decoder for an IP Set-Top Box

Fernando Pescador, Matías J. Garrido, Rafael Antoniello, Cesar Sanz, Eduardo Juarez and Carlos Santos*

Department of Sistemas Electrónicos y de Control.
Universidad Politécnica de Madrid.
Ctra. de Valencia Km. 7
28031 Madrid. Spain.

*SIDSA
PTM Torres Quevedo, 1
Tres Cantos
28760 Madrid. Spain.

Email: pescador@sec.upm.es

Abstract[1]

In this paper, the implementation of a digital signal processor (DSP) based multi-format decoder for an IP set-top box is described. Using several software optimization techniques, the multi-format decoder has been fitted into a TMS320DM641 DSP @ 480 Mhz. Starting from a native C code implementation, a six-step software optimization process has been applied to improve the decoder performance. Currently, 30% and 54% of the DSP capacity is used for MPEG-2 MP@ML and MPEG-4 ASP@L5 decoding, respectively. As a result, the cost of the whole system is lowered since the DSP has enough room left to run other IP set-top box tasks such as transport stream parsing, audio decoding, audio and video presentation and a user interface. Nowadays, our middle term goal is to fit other video decoders (H.264 and HD MPEG-2), other audio decoders (AC3 and AAC), a Real Time Operating System (RTOS) and a complete user interface in the DSP.

1. Introduction.

Set-top boxes (STBs) are becoming key devices in home entertainment networks, not only to receive digital television (DTV), but also as residential gateways to deliver multiple services as well [1]. To gain in flexibility and modularity in home networks, the functionality of STBs may be distributed between a main device and several peripherals, all interconnected by an Ethernet network. As can be seen in Figure 1, the main device works as a gateway between cable, terrestrial or satellite DTV distribution systems and the home network. The peripheral devices are IP DTV decoders. Since a user may have one of these decoders placed close to each TV set, they are also called IP STBs [2].

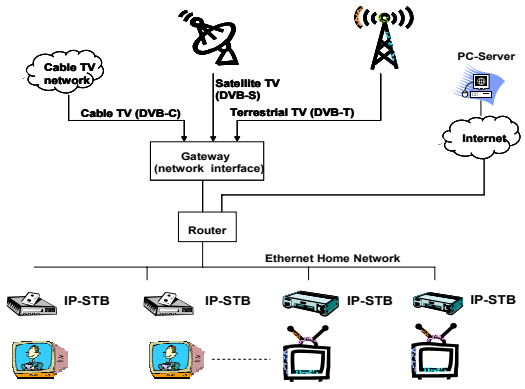


Figure 1. TV distribution in home networks.

[1] This work is partially supported by the ELITE project "Video Decoders Based on TI DSPs" and a grant TIC2003-09687-C02-01 from the *Ministerio de Ciencia y Tecnología* (MCYT) of the Spanish Government.

As a consumer electronics device, an IP STB must be cheap and versatile. The high computational costs of video decoder implementations constraints STB designs. Non-programmable decoders [3] cannot cope with the quick evolution of audio and video decoding algorithms. In addition, the latest generation of digital signal processors (DSPs) [4-6] can support inexpensive and flexible decoders. As a consequence, much effort has been done in the design of DSP-based video decoders in the last years [7-13]. Our short term goal is to develop an IP-STB with audio and video multi-format decoders. Unlike the work done in [13], our approach lowers system costs by taking advantage of the DSP capacity left to implement the rest of STB functions.

In this paper, the design and optimization of a multi-format video decoder based on a TMS320DM641 DSP @ 480 MHz is described. Performance results are given for SD MPEG-2 decoding while preliminary results are outlined for MPEG-4 ASP decoding. The optimization achieved in the video decoder has allowed the design of an IP STB whose main tasks have been implemented within a single DSP.

The rest of the paper is organized as follows. In section 2, the IP STB architecture (hardware and software) is explained for reference. In section 3, the video decoder software optimization process is described. In section 4, profiling results for all functional blocks in the MPEG-2 decoding process are outlined. In section 5, a testbench to analyze the IP-STB real-time performance is presented. In section 6, preliminary results for MPEG-4 ASP decoding are shown. Finally, section 7 is devoted to conclusions and future work.

2. IP STB Architecture.

The IP STB hardware architecture [14] can be seen in Figure 2. The TMS320DM641 DSP interfaces with an Ethernet port, two external memories, a video encoder, a digital to analog audio converter, an infrared receiver and a JTAG emulator using a minimum amount of glue hardware.

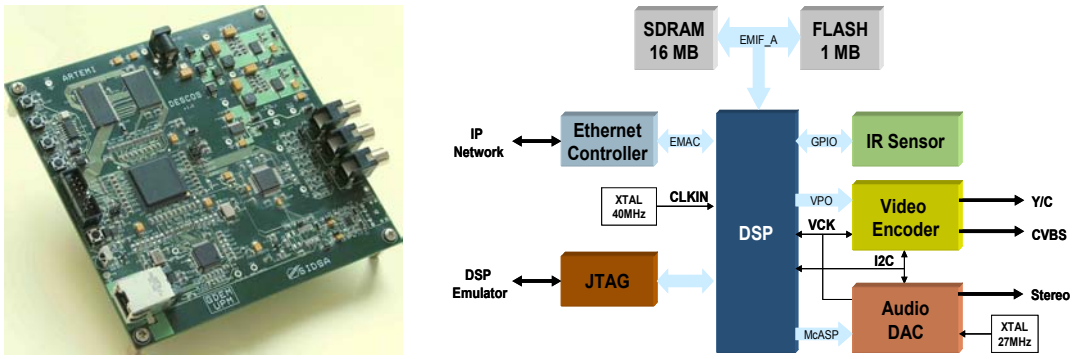


Figure 2. Board (a) and block diagram (b) of the IP STB.

To implement all IP STB tasks, the system is embedded into a multi-task software architecture that has been built using the RF5 [15] (see Figure 3). A real-time kernel [16] schedules the tasks execution and allows inter-tasks communication using queues and mailboxes. The system is composed of six tasks: transport stream demultiplexing, audio and video decoding, audio and video presentation and user application. The video decoder is integrated in the video decoding task. This task reads one frame from a transport task output buffer, decodes it, and stores the result in an image buffer. The video algorithm has been designed to be eXpress-DSP Compliant [17].

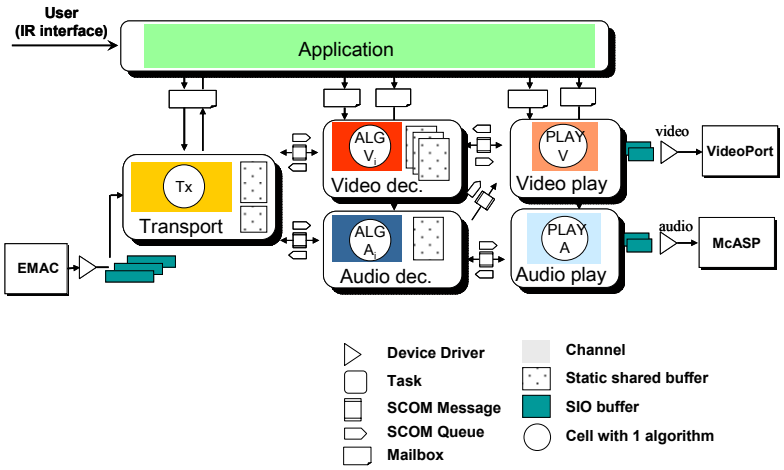


Figure 3. Software architecture (see notation details in [15]).

3. MPEG-2 video decoder optimization.

The first version of the video decoder was developed in native C code implementing only the MPEG-2 algorithm. Initially, the code was located entirely in external memory and spent about 2.000 million clock cycles per frame to decode MPEG-2 MP@ML streams.

The optimization process was developed in six steps:

1. **First step: memory management optimization.** After several iterations, the 128 kB L2 memory was configured as a 32 kB 2nd level cache and a 96 kB SDRAM internal memory. All decoder functions were fitted in the latter. With these changes, an improvement of about 95% over the initial version was achieved.

2. **Second step: function optimization.** Optimized functions taken from the DSP vendor were adapted to implement several functional blocks (variable length decoding, inverse quantization and inverse discrete cosine transform). As a consequence, an improvement of 50% above the previous results was achieved.

3. **Third step: assembly coding.** The execution was optimized by increasing the parallelism in critical parts of the algorithm. These parts were re-encoded in the DSP assembly language and parallelized by hand. The performance obtained is better than the one obtained by the linear assembler compiler. The re-encoded functionalities were:

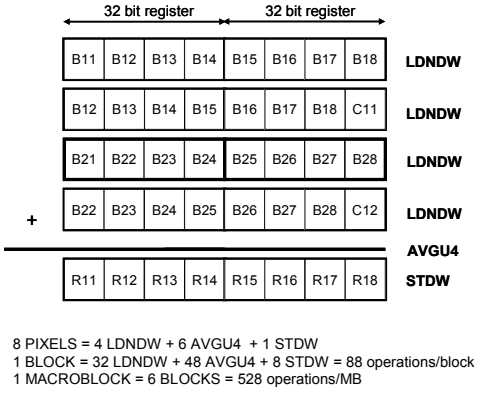
   a) ½ pixel interpolation arithmetic. Considering the worst case, to perform half-pel motion compensation for an 8×8 block, 640 operations are needed[2]. Using DSP specific assembly instructions, these operations were done using 88 instructions[3] (see Figure 4-a).

   b) Prediction error addition and saturation. To perform the motion compensation, IDCT 16-bit pels and reference 8-bit pels must be added. In order to achieve this operation in an efficient way, packing and unpacking DSP specific instructions were used. The UNPACK4 instruction was used to store two 8-bit reference pels in a 32-bit register as two 16-bit half words. Zero-padding was applied to the most significant bits of each 8-bit value. The SPACKU4 instruction saturates four signed 16-bit values to unsigned 8-bit values and stores them out as packed

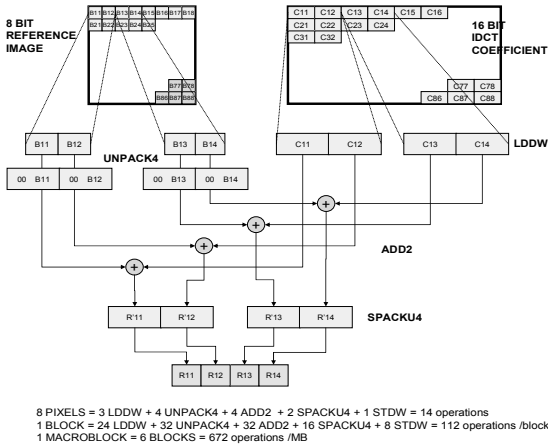
[2] 256 data loads (four pels are loaded to reconstruct each block pel), 256 additions, 64 shift operations and 64 byte stores are required.
[3] 4 load instructions (LDNDW), 6 average instructions (AVGU4) and 1 storage instruction (STDW) are needed to compute 8 pels in a block.

unsigned bytes in a 32-bit register. Using DSP specific assembly instructions, these operations were done using 112 instructions[4] (see Figure 4-b).

The implemented motion compensation function is capable of processing a complete 4:2:0 structured block in 79 CPU cycles including function call and data set-up. This represents 1.24 (79 CPU_cycles / 64 pixels_per_block) cycles per pixel processed.
Also in this step, the code and data sizes were adjusted to reduce the number of cache misses. All these changes achieved a 20% reduction in the clock-cycle count over previous results.



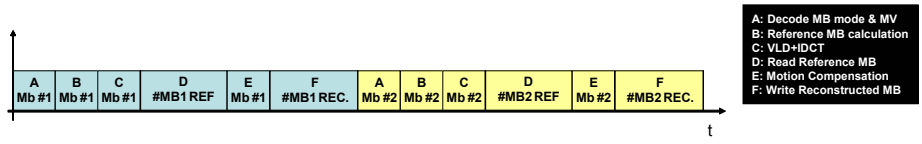
a) Operations needed to perform ½ pel interpolation for an 8×8 reference block.



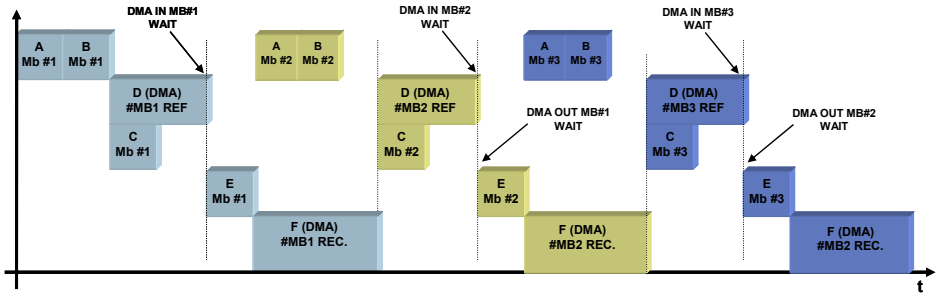
b) Optimized packaging/unpackaging DSP operations used to perform the motion compensation.

Figure 4. Assembly functions.

4. **Fourth step: DMA transfers.** DMA was used to perform data transfers mainly in the motion compensation block. Special care was taken with both, data alignment and balance among DMA queues to avoid CPU stalls. Figure 5-a shows a macroblock execution time diagram without DMA. In this time diagram all decoding phases are executed in series. Using the DMA is possible to parallelize the CPU execution and the DMA transfers (Figure 5-b). These changes improved the previous results in about 65%.



a) Serial execution



b) Parallel execution using DMA

Figure 5. Execution time diagram.

[4] 3 load instructions (LDNDW), 2 pack instructions (PACK4), 4 unpack instructions (UNPACK4), 4 add instructions (ADD2) and 1 storage instruction (STDW) are needed to compute 8 pels in a block.

5. **Fifth step: Double buffering and loop optimization.** This previous time diagram presents two limitations:

    1. Reference macroblock. Before starting the phase E of a macroblock, it is necessary that the DMA associated to the reference macroblock (phase D) is finished.

    2. Reconstructed macroblock. Before starting the phase E of the macroblock #n, it is necessary that the DMA transfer of the reconstructed macroblock #n-1 (phase F) is finished. It is worth noting that a wait is needed because the DMA transfer and the phase E of the macroblock #n use the same intermediate buffer.

    In order to solve the first problem, the decoding loop has been re-scheduled to reduce the amount of time the CPU is waiting for the DMA transfers to be completed. To solve the second problem a double buffer was included for the reconstructed macroblock. This double buffer allows to calculate the motion compensation of macroblock #n while the reconstructed macroblock #n-1 is been transferred. Figure 6 shows the execution time diagram. This diagram shows that the CPU is executing instructions continually without waiting for DMA completion.

    These changes improved the previous results in about 13%.

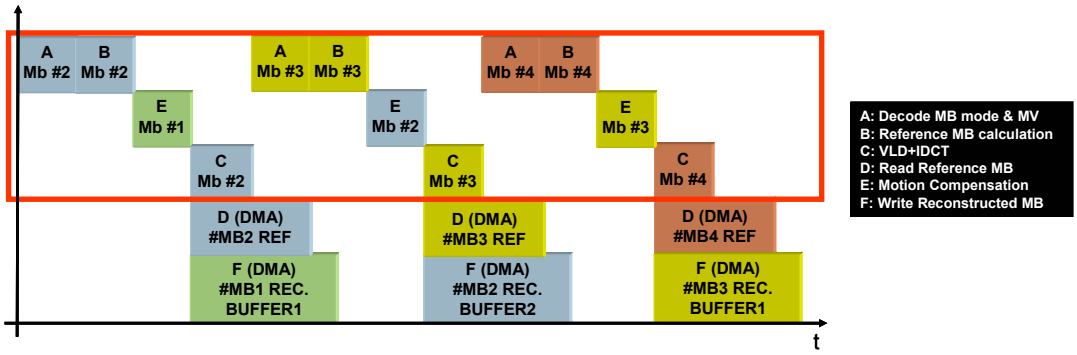


Figure 6. Execution time diagram

6. **Sixth step: slice processing.** Further optimization was obtained by reducing the number of DMA requests to store the reconstructed macroblocks in the image buffer. Before this step, for a D1 resolution frame, the number of DMA requests for a reconstructed frame was 4860[5]. In this step, a new internal buffer to store a line of macroblocks was defined. Therefore, only one DMA request per line was needed. In addition, more optimized one dimension DMA transfers were done instead of bi-dimensional ones. This change improved the previous results in 5%.

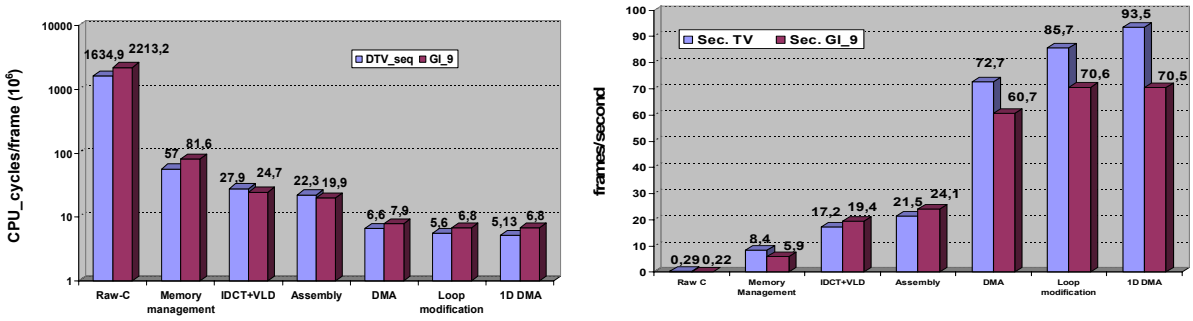
These six steps have been completed for the MPEG-2 decoder. The results are reported in section 4. Now, the same optimization techniques are been used for the MPEG-4 ASP decoder. Preliminary results are given in section 6. In addition, HD MPEG-2 and H.264 decoders are been developed using the same methodology.

[5] 720/16 mb/line × 576/16 mb/column × 3 components/mb=4860.

4. MPEG-2 decoding performance data.

The decoder performance has been measured using two video sequences. The first sequence, DTV_seq, is a video elementary stream extracted from a DTV emission with a 720x576 resolution, a 3 Mbps average bit rate and a ratio of one I frame out of two P and B frames (20% I, 40% P and 40% B frames). The second sequence is one of the MPEG-2 reference streams [18], GI_9, with a 720x480 resolution, a 15 Mbps average bit rate and a ratio of 6.25% of I frames, 18.75% of P frames and 75% of B frames. Figure 7-a shows the average number of system clock cycles needed to decode a frame from both the DTV_seq and the GI_9 streams, after each one of the six optimization steps.

Figure 7-b shows, only for reference, the average number of D1 resolution frames/sec that can be decoded with a 480 MHz CPU clock.



a) Average number of CPU_cycles/frame b) Number of images/second @480MHz

Figure 7. MPEG-2 video decoder performance.

Only for the fully optimized decoder, the number of CPU clock cycles needed to carry out the different parts of the algorithm is reported for the DTV_seq and the GI_9 streams in Table 1.

	DTV_seq (CPU CLKs•10^6)	GI_9 (CPU CLKs•10^6)
VLD and IQ	0.67	1.86
IDCT	1.08	0.82
Motion compensation	2.88	3.72
Other functions	0.50	0.40
TOTAL	5.13	6.80

Table 1. CPU clock cycles spent in the different decoding processes.

Table 2-a compares the results of our MPEG-2 decoder with the implementation proposed in [7] [6] using a DTV sequence. Table 2-b compares our MPEG-2 decoder with the implementation proposed in [8][7].

	[7]	Our implementation
Mcycles/frame	6.2	5.13

a) Average CPU clock cycles to decode a DTV frame

	[8]	Our implementation
Mcycles/frame	9.14	6.8

b) Average CPU clock cycles to decode a frame of GI_9

Table 2. Average CPU clock cycles spent to decode a frame.

The performance results of the optimized decoder allow to integrate other functionalities in the DSP, as we have done in the IP-STB prototype described in section 2.

[6] This measure assumes an average of 20 B frames, 8 P frames and 2 I frames out of every 30 frames.

[7] Actually, in **¡Error! No se encuentra el origen de la referencia.**, two implementations are presented. Our proposal is better than the first of such implementations (see Table 2-b). The second implementation is better than our proposal in terms of speed but also needs more DSP resources. The performance is analyzed using the reference stream GI_9 [18].

Finally, several reference sequences[8] have been decoded using our decoder and the ISO reference decoder [19]. The decoded sequences have been compared by computing the PSNR. In all cases, the average PSNR is greater than 50dBs.

5. IP-STB real time testbench.

In this section, the results of two real time tests performed using the IP-STB prototype described in section 2 are reported:

1. In the first one, a PC encapsulates in real time a DVD movie (Star Wars: Episode I. with 5.6 Mb/s average bit rate and 9% I, 32 % P and 59% B frames) into MP2TS over IP and streams it using a multicast IP address to the IP STB.

2. The second one uses a commercial gateway [20] to tune a satellite DTV channel (the channel is EuroSport with 3.2 Mb/s average bit rate and 5.5% I, 27.7% P and 66.6% B frames). Currently, the gateway tunes a transponder and encapsulates each program in an MP2TS over IP. All needed PSI tables are also generated. Each program inside the transponder is encapsulated in a different multicast IP address and a specific Session Announcement Protocol (SAP) packet is generated. The STB decodes the SAP and the PSI tables and plays the user selected program on a TV set.

The testbench can be seen in Figure 8. The gateway is the box at the left side. The IP-STB is the prototype mentioned in section 2, working at 480 MHz clock rate.

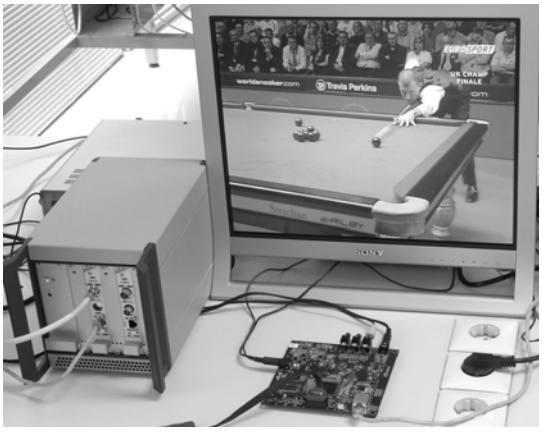


Figure 8. Testbench used in the profiling tests.

Table 3 shows the percentage of the DSP computing power needed by each task playing the movie Star Wars: Episode I and the EuroSport channel. In this table, the column "others" includes the application task, the TCP/IP stack and the RF5 overload.

Task / % DSP	Transp.	videodec.	Audio dec.	Video play	Audio play	others	free
Star Wars I	10.4	32.8	5.0	0.1	0.7	11.3	39.7
EuroSport	6.0	28.5	5.0	0.1	0.7	11.3	48.4

Table 3. CPU Percentage used and free space

These results show that there is enough free DSP computational power to support a RTOS (e.g. Linux) and a complete user interface using the same DSP.

[8] The sequences used are tcela-9, tcela-10 and conf4.

6. MPEG-4 preliminary results

Real-time profiling tests have been done showing that the decoder performs in real-time (D1 resolution @ 25 fps) using about 54% of the DSP capacity (average bit rate: 3.5 Mbps; 6.3% I, 31.2% P, 62.5% B frames). These preliminary results are currently being improved.

7. Conclusions and future work

In this paper, the implementation of a DSP-based multi-format video decoder for an IP STB has been described. The software is capable to decode MPEG-2 MP@ML and MPEG-4 ASP@L5 streams, using around 30% and 54% of the capacity of a TMSDM641 DSP @480MHz, respectively. These results have allowed the integration of the main IP-STB functionalities in a single DSP. An IP-STB prototype has been designed and fully tested in a real environment using DVD media and digital TV channels. Now, the efforts are oriented towards including other video formats such as HDTV and H.264, other audio formats such as AC3 and AAC, a RTOS and a complete user interface.

8. References

[1] F.T.H. den Hartog et al. "Convergence of Residential Gateway Technology: Analysis of Evolutionary Paths", IEEE Consumer Communications and Networking Conference, pp.1-6, Jan. 2004

[2] C. Luo et al. "Design and implementation of multiplexing rate control in broadband access network TV transmission system", IEEE Trans. on Consumer Electronics, Vol. 50, pp. 849 - 855, Aug. 2004.

[3] Q. Peng and J. Jing. "System-on-chip design for TV-centric home networks". IEEE Consumer Communications and Networking Conference, pp. 501 - 506, Jan. 2004.

[4] TMS320DM641 DSPs Data Manual". (SPRS222D). http://focus.ti.com/lit/ds/symlink/tms320dm641.pdf

[5] Philips Semicond. Nexperia. http://www.semiconductors.philips.com/products/nexperia/home/index.html

[6] Equator Technologies. Video centric SOCs. http://www.equator.com/productsservices/videocentricsocs.htm

[7] S. Cacopardi et al. "A DSP based MPEG-2 video decoder for HDTV or multichannel SDTV". IEEE Workshop on Multimedia Signal Processing, pp. 134-137, Dec. 2002.

[8] S. Arora et al. "Multi Channel MPEG-2 Decoder Implementation for the DM642 Multimedia Processor". GSPx 2003 Conference Paper. http://www.techonline.com/pdf/pavillions/gspx/475.pdf.

[9] http://www.techonline.com/pdf/pavillions/gspx/475.pdf

[10] C. Pretty and J. G. Chase. "Reconfigurable DSPs for efficient MPEG-4 video and audio decoding". Proc. of the IEEE International Workshop on Electronic Design, Test and Applications, pp. 63-67, Jan 2002.

[11] C. Basoglu et al. "The Equator MAP-CA DSP: an end-to-end broadband signal processor VLIW". IEEE Trans. on Circuits and Systems for Video Technology, vol. 12, issue 8, pp. 646-659, Aug. 2002.

[12] J.-H. Kuo et al. "A low-cost media-processor based real-time MPEG-4 video decoder". IEEE Trans. on Consumer Electronics, pp. 1488-1497, Nov. 2003.

[13] Y.-S. Tung et al. "DSP-Based Multi-Format Video Decoding Engine for Media Adapter Applications". IEEE Trans. on Consumer Electronics, Volume 51, Issue 1, pp. 273 - 280, Feb. 2005.

[14] F. Pescador et al. "A DSP Based IP Set-Top Box for Home Entertainment". IEEE International Conference on Consumer Electronic, pp. 271-272, Jan. 2006.

[15] Reference Frameworks for eXpressDSP Software: RF5 an Extensive, High-Density System (SPRA795A). http://focus.ti.com/lit/an/spra795a/spra795a.pdf.

[16] TMS320 DSP-BIOS user's guide (SPRU423B). http://focus.ti.com/lit/ug/spru423b/spru423b.pdf.

[17] TMS320 DSP Algorithm Standard API Reference (SPRU360C). http://focus.ti.com/lit/ug/spru360c/spru360c.pdf.

[18] ISO13818-4. Information Technology – Generic coding of moving pictures and associated audio information. Conformance testing. (Dec. 1998).

[19] ISO13818-5. Information technology – Generic coding of moving pictures and associated audio information–Software simulation. (1998).

[20] SIDSA. Ether TV. http://www.sidsa.com/DATASHEETS/SIDSA_EtherTV_brochure_big_box.pdf.