Video Processing on the DaVinci Platform

Alejandro A. Ramírez-Acosta[2], Mireya S. García-Vázquez[1], and Gustavo L. Vidal-González[1]

[1] Centro de Centro de Investigación y Desarrollo de Tecnología Digital (CITEDI), Av. del Parque No. 1310, Mesa de Otay, Tijuana, Baja California, C.P. 22510, México

[2] MIRAL R&D, Imperial Beach, USA

{mgarcia, vidal}@citedi.mx, ramacos10@hotmail.com

Abstract. Nowadays, the complexity of embedded systems has increased dramatically, making design process more complex and time consuming. This situation has caused significant delays in introducing new products to market and serious economic problems for several companies. Thus, the integrated circuit manufacturers have revised; redesigned or abandoned the traditional paradigms of the design of electronic circuits and systems. This effort allows the emergence of applications based on design platforms (platform-based design, PBD). This paper describes the implementation of a MPEG-4 Advanced Simple Profile video encoder prototype based on Xvid software, which is ported to the ARM9 platform-based architecture of the evaluation development DaVinci platform DVEVM355 of Texas Instruments (TI). Our encoder implementation is under eXpressDSP Digital Media (xDM) standard of TI. The importance of our work is that the implemented encoder based on xDM standard can be integrated with other software to build a multimedia system based on either DVEMs platform in a very short time. The experimental evaluation of our MPEG-4 ASP-Xvid encoder's performance demonstrate high performance and efficiency compared to the MPEG-4 Simple Profile video encoder of TI.

Keywords: Video, embedded system, DaVinci platform, platform-based design, MPEG-4 codecs, DVEVMs.

1 Introduction

Thanks to the advent of microprocessor technology, the applications based on platform-based design [1] and the emergence of standards bodies in the area of video coding such as ISO/IEC [2] and ITU-T [3], the multimedia embedded systems are a reality. For instance, recently, embedded multimedia system based on advanced digital media processors (DMP) have been integrated into commercial products to implement coding

tools in MPEG-4 real time environment [4]. The MPEG-4 includes many standard coding tools [5, 6].

Texas Instruments (TI) as an industry leader has generated DaVinci technology for embedded application development [7]. This includes hardware, software and development tools. DaVinci technology belongs to development applications platform-based design [1]. The design platforms from TI, called Digital Video Evaluation Module (DVEVM) under DaVinci technology [7], allow to create a SoC (System on Chip, [8]) for video applications such as videophones, IP set-top-box, digital cameras, IP cameras, DVRs, portable media players, media gateways, medical images, etc.

Nowadays, the complexity of embedded systems has increased dramatically, making design process more complex and time consuming. Therefore the use of standardized tools and methodologies such as DaVinci technology significantly help designers, developers, integrators and researchers to develop complex embedded systems in shorter times. In this paper we describe the use of standardized tools and DaVinci technology for the implementation of a MPEG-4 Advanced Simple Profile video encoder prototype based on Xvid software, which is ported to the ARM9 platform-based architecture of the evaluation development DaVinci platform DVEVM355 of Texas Instruments (TI). Our encoder implementation is under eXpressDSP Digital Media (xDM) standard of TI. The importance of our work is that the implemented encoder based on xDM standard can be integrated with other software to build a multimedia system based on either DVEMs platform in a very short time.

The paper is organized as follows: In section 2 the main features of the MPEG-4 standard and ASP profile are presented. In section 3, DaVinci development environment is described. The methodology for integrating a modular MPEG-4 ASP Xvid encoder in DaVinci architecture DVEVM355 is given in section 4. The results of the performance of MPEG-4 ASP-Xvid encoder ported to the ARM9 platform-based DaVinci architecture of DM355 are set out in section 5. Conclusions are drawn in section 6.

2 MPEG-4 ASP

The MPEG-4 standard and the rapid proliferation of the existing IP (Internet Protocol) networks present new opportunities and challenges to the scientific community. The MPEG-4 standard provides standardized technological elements enabling the integration of production, distribution and access to content. Examples of this are: digital television, graphic interactive applications and interactive multimedia. In addition, the standard tries to avoid a multitude of formats and owner players, which do not have interoperability capability. The MPEG-4 standard is organized into multiple parts, two of these parts define the video coding schemes: Part 2 (ISO/IEC 14496-2) [5] define the MPEG-4 visual and Part-10 (ISO/IEC 14496-10) [6] define MPEG-4 AVC (Advanced Video Coding). The MPEG-4 part-2, offers a variety of tools for visual coding. These tools are combined in subgroups called Profiles. The most common profiles for video coding are: *Simple*

Visual Profile (SP) and the *Advanced Simple Visual Profile (ASP)*. These profiles are the best choice for coding an entire image. The standard also defines Levels. These levels subdivide the profiles in terms of complexity as the image resolution, bitrates or buffer requirements. The Simple Visual Profile provides efficient, error resilient coding of rectangular video objects, suitable for applications on mobile networks, such as IMT2000 [9]. The *Advanced Simple Profile* looks much like Simple in that it has only rectangular objects, but it has a few extra tools that make it more efficient: B-frames, ¼ pel motion compensation, extra quantization tables and global motion compensation.

3 DVEVM355 Evaluation Module

DaVinci technology [7] is a signal processing based solution tailored for digital video applications that provides video equipment manufacturers with integrated processors, software, tools and support to simplify the design process and accelerate innovation. DaVinci technology refers to the *DM* platform of media processors with their associated development tools, software components, and support infrastructure including third party companies.

The DM3x generation, including DM355 processors [10], is ideal for applications compliant to MPEG-4 standard (video-playback devices such as IP cameras, video doorbells, video conferencing, digital signage, portable media players and more). This processor, due to its parallelism, it can be used to process multiple blocks of an image simultaneously. This improves system performance and overall performance compared to the serial processors.

The DVEVM355 evaluation module [11] is based on the TI TMS320DM355 processor [10]. DM355 is a multimedia processor with ARM9EJ processor and hardware video accelerator (fixed function co-processor, MJCP) and a set of peripherals for multimedia products. To harness the processing power of the DM355, we integrate MPEG-4 ASP encoder for the DM355 ARM processor.

The following figure shows the software components used for application development with the DVEVM kit.

In Figure 1, everything runs on the ARM. The application handles I/O and application processing. To process video, image, speech, and audio signals, it uses the VISA APIs provided by the Codec Engine. The Codec Engine, in turn, uses xDM-based codecs (enCOders/DECoders).

xDM, which stands for xDAIS for Digital Medial [12] is a standard API (Application Programmer Interface) that is wrapped around all signal processing functions, especially encoders/decoders. In the figure, this is represented as a container that holds a codec in VISA API. The container, which represents the code interconnections in the xDM standard and the Codec Engine negotiate the execution stages and the resources required by the encoder/decoder. VISA API (Video, Imaging, and Speech & Audio) abstracts the details of signal processing functionality, and allows an application developer to leverage

the benefits of these functions without having to know their details. The API VISA allows managing a common format to call the encoder/decoder through the codec engine. Codec Engine (CE) [13] is a piece of software of the technology DaVinci that manages the system resources and translates VISA calls into xDM calls.

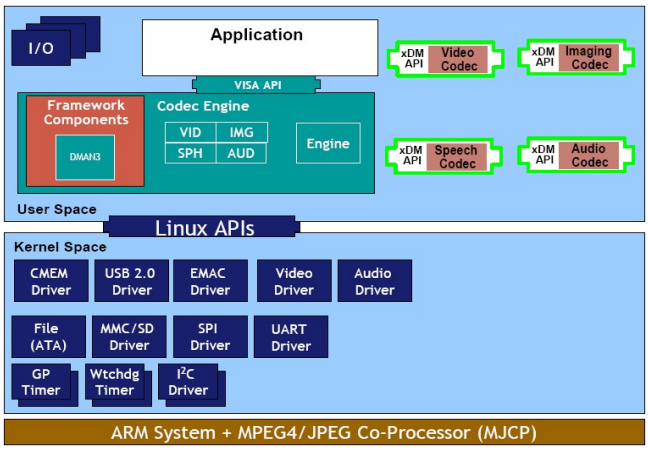

Fig. 1. Software components.

4 MPEG-4 ASP Encoder in DVEVM355

DaVinci technology consists of modular software components, which perform various tasks for an application. The encoders/decoders are called in the application using standard APIs. These APIs provide the interface with different modules that contain the encoder/decoder instructions [14] (cf. section 3). To evaluate the performance of our MPEG-4 encoder ported in the embedded platform, we develop an application in DVEVM355 that encodes a video sequence with the MPEG-4 ASP-Xvid encoder. This application is based on standards VISA, xDM and codec engine of the DaVinci technology to develop embedded software.

The Xvid encoder [15] is an implementation non-compliant to standard MPEG-4 part-2 [5]. This encoder is under open source license known as GNU GPL (General Public License) [15]. We performed the integration of the Xvid 1.3.0-rc1 encoder in the ARM9 platform-based architecture under TI DaVinci standards [10-13]. The operating system used is MontaVista's embedded Linux distribution Pro v4.0.1.

The Xvid encoder was developed in *ANSI C* code following a modular structure [15]. The modularity of the code allows the reuse of software components in certain operating systems and processor architectures. The Xvid encoder has many tools [15] that are not covered by the MPEG-4 ASP encoder. Thus, we can say that Xvid encoder is close to

compliant, but no really compliant. For this reason, one of our contributions in this work is the Xvid encoder adaptation being compliant to the MPEG-4 ASP [5] standard. With this, we only integrate the tools provided in the standard for MPEG-4 ASP encoder. The coding tools of the Xvid encoder that satisfy the MPEG-4 ASP profile are: ASP@Level 1 (ASP@L1), ASP@L2, ASP@L3, ASP@L4, ASP@L5; I-Frames/Keyframes, P-Frames/directional encoding, B-Frames/Bi-directional encoding, Quarter Pixel Motion Estimation search precision (QPEL), Global Motion Compensation (GMC), Interlace coding, and H.263/MPEG/Custom Quantization.

Another interesting contribution of our work is the porting of the Xvid encoder to the ARM9 CPU; this porting takes advantage of the TI DaVinci standards and the DaVinci TI DVEVM355 platform. The method to porting the Xvid encoder in the ARM9 platform-based architecture of DM355 is divided in two parts:

- Xvid encoder configuration to the ARM9 platform-based architecture.
- Porting of Xvid encoder to the xDM TI standards.

4.1 Xvid Encoder Configuration to the ARM9 Platform-based Architecture

The Xvid encoder is configured to be executed on the ARM9 processor architecture into the DM355 of TI. The parameters for the set up are: target platform *"ARM generic"*; type of compiler *"ARM_V5T_LE-GCC"*, the company Montavista Linux; host platform *"x86_64-pc-linux"*.

4.2 Porting of Xvid Encoder to the xDM TI Standards

Once that the Xvid encoder is integrated with the coding tools under the MPEG-4 ASP profile and its configurations to be executed in the DM355 ARM processor, we develop the MPEG-4 ASP-Xvid encoder implementation under DaVinci TI standards. The methodology is based on the recommendations of R. Pawate [14]. We used the APIs VISA, xDM and the Codec Engine for the management of MPEG-4 ASP-Xvid encoder being compatible with the software components of the DVEVM355 platform. The implementation is described in the following lines:

Figure 2 shows the outline to follow to implement an encoder/decoder under DaVinci technology. The codec engine software basically translates these *create*, *control*, *process* and *delete* APIs to their respective xDM APIs, while managing the system resources and inter-processor communication.

The process and control API of VISA are a direct reflection of the low-level process and control functions of the xDM algorithm (codec1). As a result, Texas Instruments providing low-level control of codecs along with high level abstraction of the details. The figure 3 shows the specific VISA and xDM APIs for the video encoder/decoder. For a

given class, say Video, the signature of these APIs is held constant. This enables an integrator and developer to easily replace one encoder/decoder with another.

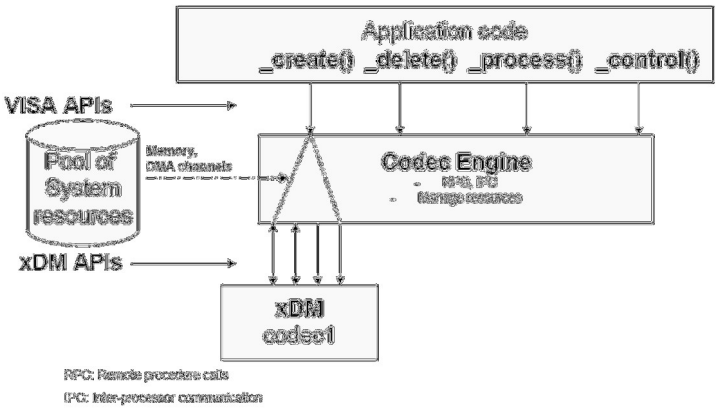

Fig. 2. Management of an encoder/decoder under DaVinci technology.

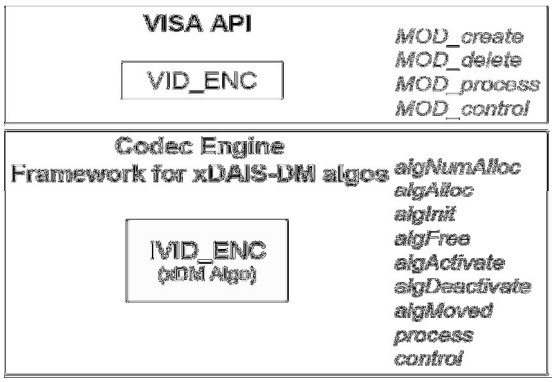

Fig. 3. VISA Abstracts details of xDM algorithms for video encoder/decoder.

To use the followings APIs: VISA, xDM and of the Codec Engine, the application is divided into four logical blocks:

- Parameter setup,
- Algorithm instance creation and initialization,
- Process call – xDM 1.0,
- Algorithm instance deletion.

4.2.1 Parameter Setup

The video encoder/decoder requires various configuration parameters to be set at initialization. Our application obtains the required parameters from the encoder configuration. The coding parameters are assigned to *VIDENC1_XVID_PARAMS* structure, which is an extended version of the generic structure *IVIDENC1_Params*. The generic structure is used by video codecs for the API VISA. We verify that the parameters are within the ranges set by the MPEG-4 ASP standard. We also assign memory space E/S for the data of the entry frame and for the data of the coded frame. The allocation of contiguous memory space is done by the function: *Memory_contigAlloc()*. After successful completion of the above steps, the application does the algorithm instance creation and initialization.

4.2.2 Algorithm Instance Creation and Initialization

In this logical block, the application accepts the various initialization parameters and returns an algorithm instance pointer. *The VIDENC1_XVID_Create() API* creates an instance of xDM encoder MPEG-4 ASP-Xvid and allocates the required resources for the encoder MPEG-4 ASP-Xvid to run. *VIDENC1_XVID_Create() API*, using the Codec Engine, queries the xDM encoder for the resources that it needs, and based on the encoder requirements, it allocates them. The following APIs are called in sequence:

- *algAlloc()* – to query the algorithm about the memory requirement to be filled in the memory records.
- *xvid_global(XVID_GBL_INIT)* – to initialize the global variables of the algorithm.
- *xvid_encore(XVID_ENC_CREATE)* – to initialize the algorithm.
- *algInit()* – to initialize the algorithm with the memory structures provided by the application.

4.2.3 Process Call – xDM 1.0

After algorithm instance creation and initialization, the application does the following:

- Sets the dynamic parameters (if they change during run time) by calling the *VIDENC1_XVID_control()* function.
- Sets the input and output buffers descriptors required for the *VIDENC1_XVID_process()* function call. The input and output buffer descriptors are obtained by calling the *VIDENC1_XVID_control()* function.
- Call the *VIDENC1_XVID_process()* function to encode a single frame of data. In this function, we call the function *xvid_encore(XVID_ENC_ENCODE)* which coding the frame at the input buffer with MPEG-4 ASP-Xvid encoder. When the process function ends, the output buffer is updated with the encoded picture.

The *VIDENC1_XVID_control()* and *VIDENC1_XVID_process()* functions should be called only within the scope of the *algActivate()* and *algDeactivate()* *xDAIS* functions which activate and deactivate the algorithm instance respectively. Once an algorithm is

activated, there could be any ordering of *VIDENC1_XVID_control()* and *VIDENC1_XVID_process()* functions. The do-while loop encapsulates frame level *VIDENC1_XVID_process()* call and updates the input buffer pointer every time before the next call. The do-while loop breaks off either when an error condition occurs or when the input buffer exhausts.

4.2.4 Algorithm Instance Deletion

Once encoding is complete, the application must release the resource for the encoder MPEG-4 ASP-Xvid and delete the current algorithm instance.

5 Results

This section presents the evaluation of MPEG-4 ASP-Xvid encoder ported in the embedded Linux operating system under ARM9 platform-based architecture of DM355. In order to evaluate our MPEG-4 ASP-Xvid encoder porting, we develop an application in DVEVM355 that encodes a video sequence with the MPEG-4 ASP-Xvid encoder. In order to have a reference for comparing our encoder's performance, we conducted the evaluation of the MPEG-4 SP encoder of TI for DM355 [16] under the same conditions. The test database consists of three video sequences *(Bus, Foreman and News)*. These sequences are used as reference by the integrators developers, and the scientific community to evaluate the encoder's performance. These sequences have great information in texture, movements ranging from low to high level. Each video has the following parameters: 10 seconds of video, CIF (Common Interface Format) resolutions (352x288), YUV 4:2:0P, 30 fps. The coding scheme was tested with different constant bitrates: CBR: 64kbps, 128Kbps, 384Kbps, 512Kbps and 1024Kbps. The GOP size for both encoders is 12 (frames). The GOP structure for the MPEG-4 ASP-Xvid encoder is IBBPBBPBBPBB, while for the MPEG-4 SP encoder of TI is IPPPPPPPPPPP. According to the above parameters, the three sequences were coded for both encoders under DaVinci platform. Then, the coded sequences were decoded in the host computer. To evaluate the performance of our integration of MPEG-4 ASP-Xvid encoder in the DVEVM355 architecture, according to the standard xDM of TI, we consider Peak Signal to Noise Ratio of the Y-plane of the frame (Y-PSNR) measure. Equation 1 describes Y-PSNR mathematically.

$$d\left(I, I^{'}\right) = 10 \cdot log_{10} \frac{(Max_I)^2 \cdot m \cdot n}{\sum_{i=0}^{m-1} \sum_{j=0}^{n-1} \left(I(i,j) - I^{'}(i,j)\right)^2} \quad (1)$$

where $d(I, I')$ is the Y-PSNR value, Max_I indicates the largest possible pixel value, I is the original frame and I' is the decoded frame, m,n are the frame size mxn (352x288). The Y-PSNR is obtained of the average of each video sequence. The other evaluation metric used is based on compression ratio.

The figures 4, 5 and 6 depict the result of average Y-PSNR value of Bus, Foreman and News video sequences with different bitrates.

Based on the video quality measure results, the MPEG-4 ASP-Xvid encoder ported to the DM355 maintained better performance than the MPEG-4 SP encoder of TI for the three sequences, which contain different levels of motion.

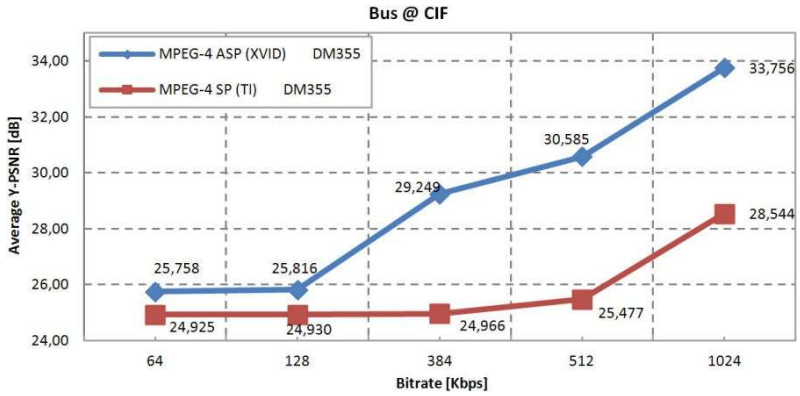

Fig. 4. Coding efficiency: comparison between MPEG-4 SP of TI encoder and our integration of MPEG-4 ASP-Xvid encoder using Bus video sequence.

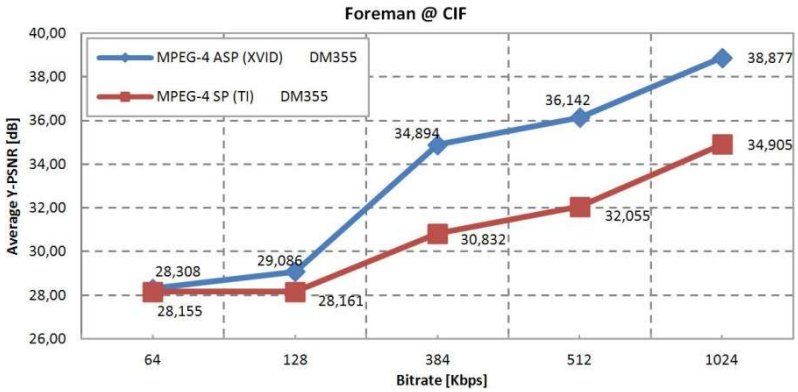

Fig. 5. Coding efficiency: comparison between MPEG-4 SP of TI encoder and our integration of MPEG-4 ASP-Xvid encoder using Foreman video sequence.

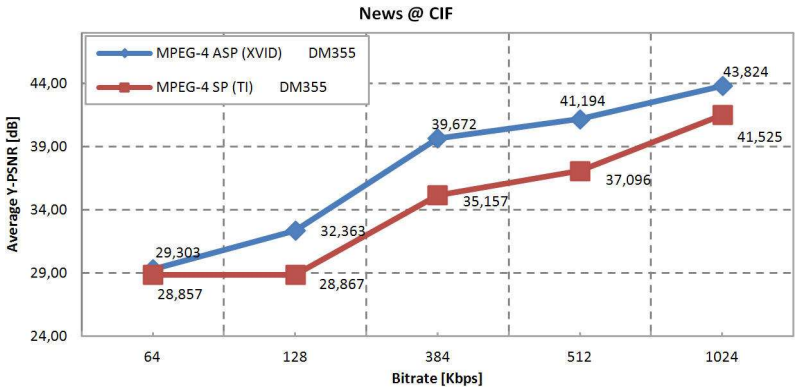

Fig. 6. Coding efficiency: comparison between MPEG-4 SP of TI encoder and our integration of MPEG-4 ASP-Xvid encoder using News video sequence.

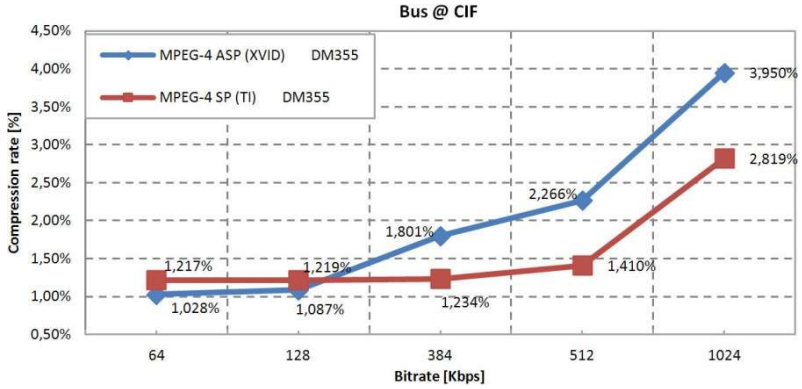

Fig. 7. Compression efficiency: comparison between MPEG-4 SP of TI encoder and our integration of MPEG-4 ASP-Xvid encoder using Bus video sequence.

With respect to overall performance on the three sequences for both encoders, we observe that the best quality is obtained with the News sequence (Fig. 6) behaving with low movement, followed by Foreman (Fig. 5) that presents a middle movement category. Finally, the worst quality is obtained with the Bus sequence (Fig. 4), which presents a high motion category. Based on the compression ratio, figures 7, 8 and 9 show the encoder's performance under different bitrates. The compression ratio represents the size in Kbytes of coded video with respect to the size in Kbytes of the original sequence. The rate of compression is presented as a percentage. With respect to the measure of compression ratio, a value close to zero represents a better performance of the encoder.

With respect to overall performance on the three sequences for both encoders, we can summarize that the best compression is obtained with the News sequence (Fig. 9), then foreman (Fig. 8) and finally, the worst compression is obtained with Bus sequence (Fig. 7).

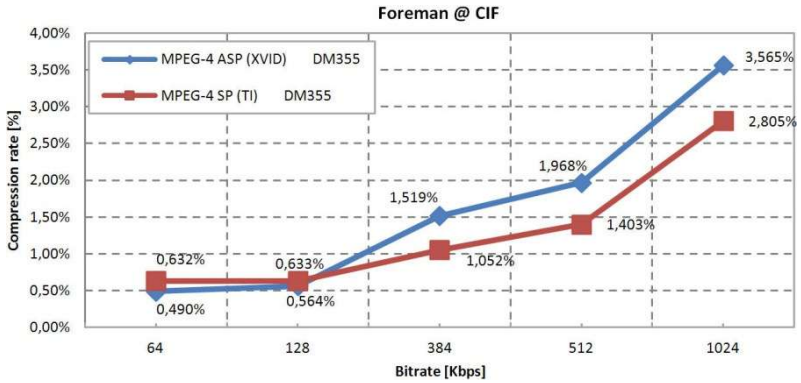

Fig. 8. Compression efficiency: comparison between MPEG-4 SP of TI encoder and our integration of MPEG-4 ASP-Xvid encoder using Foreman video sequence.

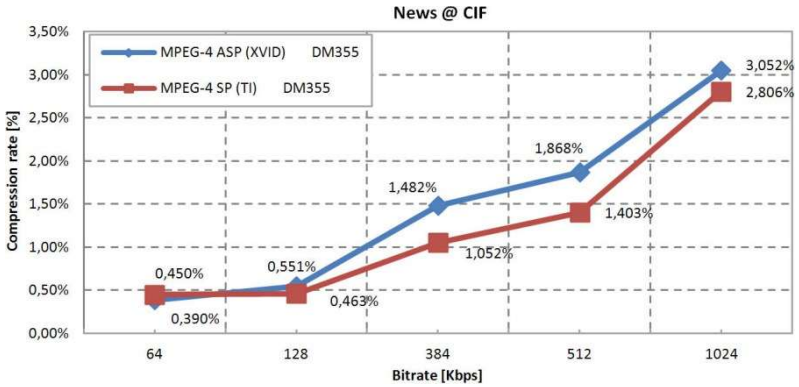

Fig. 9. Compression efficiency: comparison between MPEG-4 SP of TI encoder and our integration of MPEG-4 ASP-Xvid encoder using News video sequence.

6 Conclusions and Future Direction

The main advantage of developing applications platform-based design is the integration of software/hardware components from different manufacturers in a very short time. The

development of a complete software application that constitutes a working embedded system relies on many software components. For this reason, this work leverages the use of an embedded Linux operating system tailored for the DaVinci platform. Based on the analysis of the DaVinci technology, this paper presented a prototype of an MPEG-4 encoder video profile ASP based on Xvid software ported to the architecture ARM9 processor of the DaVinci Platform DVEVM355 of Texas Instruments. We developed an application to evaluate the performance of our MPEG-4 ASP Xvid encoder. This application is based on standards VISA, xDM and codec engine of the DaVinci technology to develop embedded software. The encoder is then evaluated for performance on the platform DVEVM355. Based on testing and performance, the MPEG-4 ASP Xvid encoder ported to the ARM9 platform-based architecture of DM355 is a good candidate to replace the MPEG-4 SP encoder of TI since it shows higher quality video. As future work, we will migrate our algorithm in a fully-programmable DSP to leverage the power of the DSP on the SoC (ARM+DSP) since the co-processor of the DM355 is not a DSP open for programming.

Acknowledgements. This work was supported by IPN-SIP 20110032.

References

1 Sangiovanni-Vincentelli, A., Carloni, L., De Bernardinis, F., Sgroi, M.: Benefits and challenges for platform-based design. In: Proc. DAC, pp. 409–414 (2004)
2 International Organization for Standardization (ISO/IEC). http://www.iso.org (2011)
3 International Telecommunication Union (ITU-T). http://www.itu.int/ITU-T (2011)
4 Chatterji, S., Narayanan, M., Duell, J., Oliker, L.: Performance evaluation of two emerging media processors: Viram and imagine. In: International Parallel and Distributed Processing Symposium, pp. 229–235 (2003)
5 ISO/IEC 14496-2:2004, Information technology – Coding of audio-visual objects – Part 2: Visual (Approved in 05-24) (2004)
6 Wiegand, T., Sullivan, G. J., Bjontegaard, G., Luthra, A.: Overview of the H.264 / AVC Video Coding Standard. IEEE transactions on circuits and systems for video technology (2003)
7 Texas Instruments, http://www.ti.com (2011)
8 Shin, D., Gerstlauer, A., Dömer, R., Gajski, D.: Automatic Network Generation for System-on-Chip Communication Design. In: Proceedings of the International Conference on Hardware/Software Codesign and System Synthesis. Jersey City, New Jersey (2005)
9 IMT-2000.: International Mobile Telecommunications, http://www.itu.int/osg/spu/imt-2000/technology.html (2011)
10 TMS320DM355 Digital Media System-on-Chip (DMSoC) (Rev.G, 24 June). SPRS463G. Texas Instruments (2010)
11 TMS320DM355 Evaluation Module Technical Reference. 509905-0001 Rev. E. Spectrum Digital, Inc. (2008)
12 xDAIS-DM (Digital Media), User Guide. Literature Number: SPRUEC8B. Texas Instruments (2007)

13 Codec Engine Application Developer User's Guide. Literature Number: SPRUE67D, September. Texas Instruments (2007)
14 Pawate, B.I. R.: Developing Embedded Software using DaVinci & OMAP Technology, Synthesis Lectures on Digital Circuits and Systems. Morgan & Claypool Publishers (2009)
15 Xvid Codec. http://www.xvid.org (2011)
16 MPEG-4 Simple Profile Encoder Codec on DM355. User's Guide. Literature Number: SPRUFE4C. Texas Instruments (2008)