INTEGRATION OF MPEG-4 VIDEO TOOLS ONTO MULTI-DSP ARCHITECTURES USING AVSYNDEX FAST PROTOTYPING METHODOLOGY

Jean-François NEZAN, Mickaël RAULET, Olivier DEFORGES
IETR / INSA Rennes
CNRS UMR 6164
20, av des Buttes de Coëmes, CS 14315
35043 Rennes Cedex, France
Contact : jnezan@insa-rennes.fr
Phone number : +33/0 223 238 459 Fax : +33/0 223 238 262

ABSTRACT

Mpeg-4 is a response to the growing need for coding method that can facilitate access to visual objects in natural and synthetic moving pictures. Future real time audio-visual applications using Mpeg-4 will have very important time constraints, that can be achieved with the use of several calculation units. Sequential software solutions actually developed for single processors can hardly be projected onto multiprocessor architectures, leading to extra load of source code and calculations, but also to a sub-optimal use of the architecture parallelism. A functional data flow description of the application is then a well suited front-end for optimal multi-components implementation. This paper presents an Mpeg-4 decoder with such description formalism, allowing incremental building, and easy handing-over up to date of the algorithms. Furthermore, we show that the use of our AVSynDEx methodology enables its optimized implementation onto a multi-C6X platform.

1. INTRODUCTION

The Moving Picture Experts Group (MPEG) [2] is the ISO/IEC working group in charge with the development of compression, decompression, treatment and representation of both video and audio documents. Mpeg gives a framework for those applications. Software solutions based on standard monoprocessor architectures are able to handle applications such as data storage or audio document manipulation. However, Mpeg-4 tools provide solutions for many other real time applications manipulating sounds, images and video from both natural and synthetic origin. Such applications, especially those including interactions with users, need very high computational performances, involving the use of dedicated technology.

Nowadays, a single chip including all Mpeg-4 tools can't be realized because of their high complexity [3,4]. Some are developed to accelerate repetitive calculations like DCT or complete image coding in special Mpeg-2 cases. Furthermore, an application will have to switch between these tools, involving the use of programmable targets. So their integration into platforms requires many specialized people, especially because platforms created are mixed, made up of standard programmable processors (software part) and those chips (specific hardware part). Such platforms are usually dedicated to a single application. We present here the AVSynDEx methodology (concatenation of AVS+SynDEx) aiming to the fast prototyping of multiprocessor platforms which could reach those high performances, and Mpeg-4 real time decoding results onto a multi-C6X architecture.

The front-end of AVSynDEx [1] is the AVS object-oriented visual programming interface for the description of the algorithm data flow graph. The only solutions actually proposed are sequential software programs that can't be projected easily onto multiprocessor architectures, whereas such functional data flow graphs are better suitable. The behavioral description is validated thanks to AVS visualization tools and is then automatically transformed to be compliant with Syndex entrance, a CAD software which evaluates and generates the tasks scheduling over the target multi-DSP architecture. The functional description of the application can be projected onto many multiprocessor platforms.

One main advantage is that AVSynDEx is based on the use of available and efficient CAD tools established along the design process so that most of the implementation tasks become self-running. Only a signal-processing contributor is needed, all the other specialized manual tasks being transparent in this prototyping methodology, so that the implementation time is reduced. The high level development allows a maximal adaptation between the Mpeg-4 standard and the application, and also an easy handing-over up to date of the algorithms. Tasks are automatically shared between processors, leading to high computational performances. Furthermore, a same programmable platform can be used for several applications, increasing the platform profitability.

This paper is organized as follows : in Section 2, Mpeg-4 specificity is studied. In section 3, we describe the AVSynDEx methodology. Results of an Mpeg-4 decoding process implementation are developed in section 4. Finally, conclusions and perspectives are given in section 5.

2. MPEG-4 CODING OF AUDIO-VISUAL OBJECTS

2.1 Generalities

Since 1988, three standards have been developed : Mpeg-1, Mpeg-2 and Mpeg-4 [2]. This later builds on the proven success of three fields : digital television, interactive graphics applications (synthetic content), and interactive multimedia (World Wide Web, distribution of and access to content). MPEG-4 provides the standardized technological elements enabling the integration of the production, distribution and content access paradigms of the three fields. Mpeg-4 provides a toolbox for next generation multimedia applications.

Mpeg-4 provides the way to represent units of aural, visual or audiovisual content, called "media objects". These media objects can be of natural or synthetic origin, this means they could be recorded with a camera or microphone, or generated with a computer.

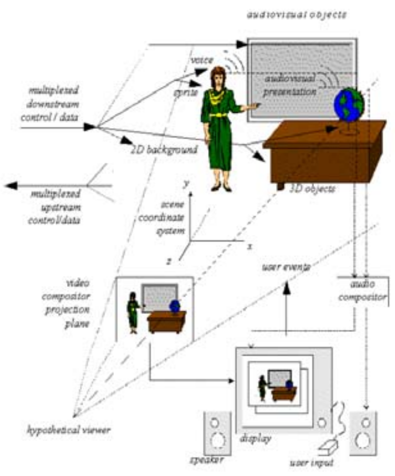


Fig 1 : Mpeg-4 audiovisual scene example

Audiovisual scenes composed of these objects can be created and manipulated. Mpeg-4 describes the composition of these objects to create audiovisual scenes (Fig 1). Media objects associated data are multiplexed and synchronized in the bitstream, so that they can be transported over network channels providing a QoS appropriate for the nature of the specific media objects. Another important MPEG-4 functionality is to provide interaction with the audiovisual scene generated at the receiver's end.

2.2 Mpeg-4 organization : profiles – levels

Many Mpeg-4 diagrams are defined in the standard according to bit rate, document complexity, use of the document. The first software programs developed at the moment try to integrate the most Mpeg-4 services as possible. But for real time applications especially in an embedded context, this approach is not suitable : resources and calculations must be minimized to reach higher performances. A methodology enabling to implement minimum Mpeg-4 services has to be used. Mpeg-4 is developed in this mind being divided into parts, each of them divided into profiles and levels [2].

A profile is a defined subset of the entire Mpeg bitstream syntax. For instance, a profile can deal with visual documents, another one with audio documents. In each profile, it is possible to require a very large variation in the performance of encoders and decoders depending on the values taken by parameters in the bitstream. In order to deal with this problem, levels are defined within each profile. A level is a defined set of constraints imposed on parameters in the bitstream.

By this way, Mpeg standards are intended to be generic, in the sense that they serve a wide range of applications, bit rates, resolutions, qualities and services. Mpeg-4 is also divided into parts. The first one is the system part, it defines the framework for integrating the natural and synthetic components of complex multimedia scenes. The Systems level shall integrate the elementary decoders for media components specified by MPEG-4 other parts : Audio, Video, Synthetic and Natural Hybrid Coding (SNHC), and Intellectual Property Management and Protection (IPMP), providing the specification for the parts of the system related to composition and multiplex.

The developer has to choose a set of those profile@levels but not the whole standard according to the application's functionalities and complexity. In the meantime, the way to process each step is not defined in the standard, only global decoding schemes are normalized. So compression diagrams can be optimized and adapted to the application (rapidity, code length, resources). The use of Mpeg-4 libraries constituted of modules interconnected in a functional data flow graph for the description of a specific application is well suited in this context. A given Mpeg-4 functionality can be handled by several modules with special specificities. An application functional description with a data flow graph is then adapted for the Mpeg-4 profile and level organization.

2.3 Mpeg-4 part 4 : conformance testing

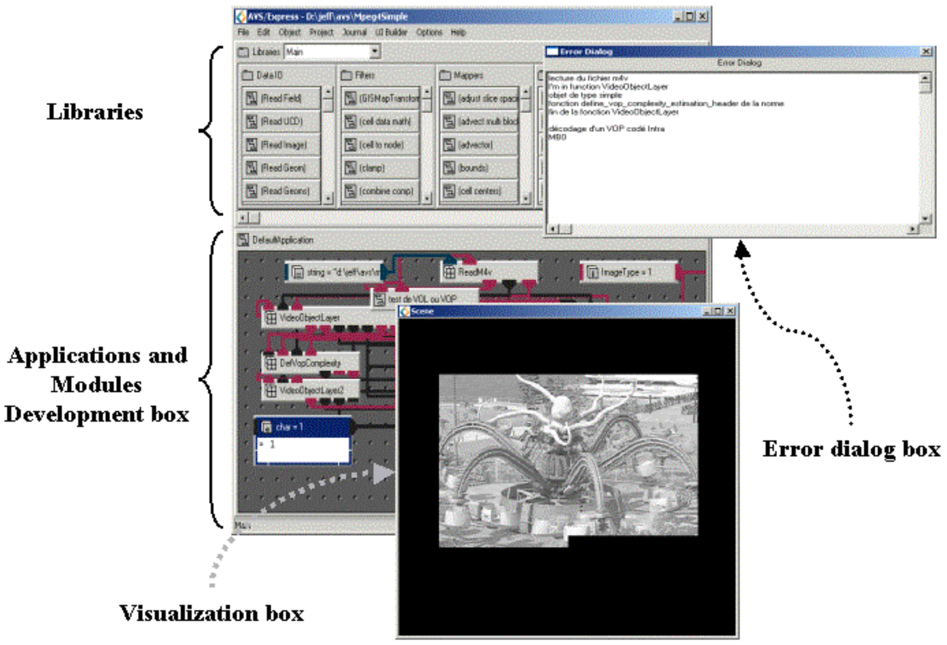
Mpeg-4 parts 1 (system), 2 (visual) and 3 (audio) specify a multiplex structure and coded representations of audio-visual information. The flexibility is obtained by including parameters in the bitstream that define the characteristics of coded bitstreams, specifying for example the picture size or the audio sampling frequency. The fourth Mpeg-4 part specifies how tests can be designed to verify whether bitstreams and decoders meet the Mpeg-4 requirements. The conformance testing part allows to test each subset of the standard in a coding or a decoding application. a Mpeg-4 data flow graph application is divided in functional blocks that can be tested independently thanks to adapted conformance testing. A complete application development can be done in an incremental way, checking in several conformance points if results fit to Mpeg-4 requirements.

So Mpeg-4 does not only describe coding and decoding processes, but it gives also many application development tools. The AVSynDEx methodology we used in our development is fully coherent with this Mpeg-4 specificity.

3. AVSYNDEX PROTOTYPING METHODOLOGY

3.1 AVS : Advanced Visual System

AVS [5] is a multi-platform, component-based software environment building application with interactive visualization and graphic features. AVS employs an innovative object-oriented visual programming interface to create, modify, and connect application components.



```
lecture du fichier m4v
I'm in function VideoObjectLayer
objet de type simple
fonction define_vop_complexity_estimation_header de la norme
fin de la fonction VideoObjectLayer

décodage d'un VOP codé Intra
MB0
```

Fig 2 : AVS development environment

The top section of the AVS Network Editor, shown Fig 2, consists of libraries of objects. The bottom section is the workspace where the user drags and drops objects and draws connection lines between objects to assemble an application or to create new objects. Objects are stored in new libraries for future applications. AVS visualization tools and the error dialog box are used to observe during the calculations. Objects created are linked to C source code developed, compiled and debugged with Microsoft Visual C++ tools.

We have developed an semi-automatic translator generating a global SynDEx description with the data flow graph created with AVS, and “cleaning” the source C functions in order to get a standard C code.

3.2 SynDEx

SynDEx [6] (SYNchronized Distributed Executive) is a free academic CAD software system, meaning Synchronized Distributed Execution. It supports the AAA methodology (Adequation Algorithme Architecture) for distributed processing, which has been developed in INRIA Rocquencourt, France. The goal of adequation, (French word meaning an efficient matching) is to find the best matching between an algorithm and an architecture. The AAA optimization heuristic handles heterogeneous architectures and inter-processor communications.

On the one hand, SynDEx uses a material graph, which models the multiprocessor architecture. Fig 3 shows an architecture made of two processors (“U1” and “U2”) connected each other with a single TCP link (called “TCP”). On the other hand, a software graph describes the dataflow graph. A software graph is constituted of interconnected vertices. There are five types of vertex: constant, sensor (input of the algorithm), actuator (output of the algorithm), memory and operation. Only an operation may contain a hierarchy, that is to say it, in turn, may be specified as a graph of other constants, sensors, actuators, memories, operations, and moreover input and output ports. The algorithm graph can also contain condition nodes. By this way, complete algorithm functional descriptions can be modeled.

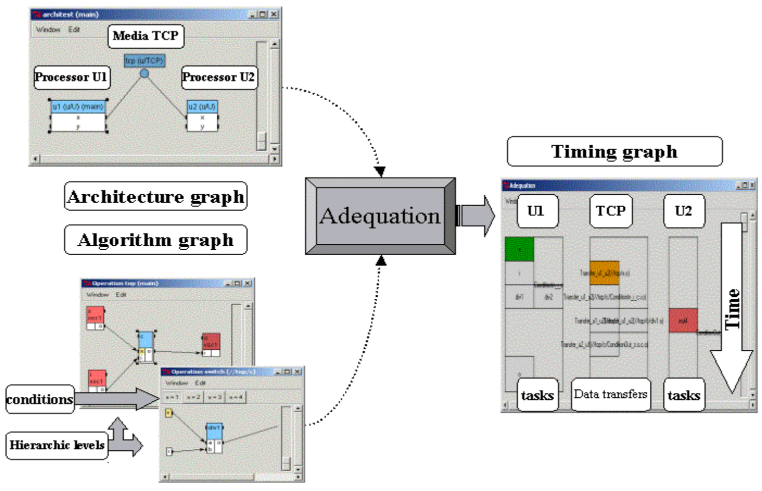


Fig 3 : SynDEx description graphs and the resulting timing graph

SynDEx generates an executive into several source files, one for each processor of the architecture, and another one for automating the architecture specific compilation chain. The code generated can insert chronometrical reports for heuristic optimization or not, for the final implementation. The main advantage of Syndex is to avoid the use of operating system like 3L diamond product, implementing the minimum custom-built static executive needed for a given application. Spatial and temporal additional costs are minimized, whereas the order of algorithm tasks is guaranteed and locking is avoided.

Finally, SynDEx carries out the placement and partitioning, according to the time spent for data transfers between processors, and for each task of the algorithm. The result can be visualized and analyzed thanks to the timing diagram generated by SynDEx.

SynDEx is able to handle different processors: Analog Device ADSP 21060, SHARC, Motorola MPC 555 et MC 68332, Intel i80x86 et i8096, Unix/Linux workstations, Texas Instruments TMS320C40. This latter was very used, but its computational performance is no longer efficient enough for new applications. From 150 to 600 Mhz clock rates, C6X are using VelociTI Advanced Very-Long-Instruction-Word (VLIW) architecture, in order to supply up to eight 32-bit instructions to the eight functional units every clock cycle. C6x are high-performance DSPs. We coupled SynDEx advantages to C6x DSP power to create its SynDEx automatic code generator [7]. Communications are done with multi-channel DMA transfers, maximizing this parallelism and timing performances.

3.3 AVSynDEx [1]

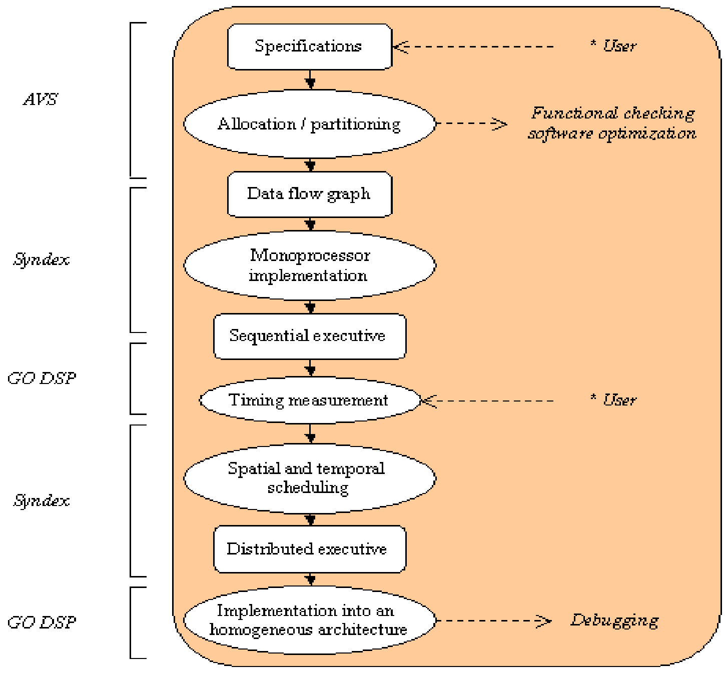

AVS
Syndex
GO DSP
Syndex
GO DSP
Specifications
* User
Allocation / partitioning
Functional checking software optimization
Data flow graph
Monoprocessor implementation
Sequential executive
Timing measurement
* User
Spatial and temporal scheduling
Distributed executive
Implementation into an homogeneous architecture
Debugging

Fig 4 : AVSynDEx methodology

The starting point of the prototyping process is the functional description of the application, with the use of AVS (Fig 4). The functional checking, but also software optimization of modules and applications, are made at an high level, depending on platform resources and time constraints of the application.

This description is then translated into an input file for Syndex, used in a first step to determine the time associated with each function. It creates a monoprocessor implementation with chronometric reports. In the C6x case, reports are done with this one of the two internal timers. The user can easily copy out these times into the software SynDEx graph. Then, Syndex generates a real-time distributed and optimised executive, where chronometrical report are removed, according to the target platform.Several platform configurations can be simulated (processor type, their number, but also different media connections).

The main advantage of this prototyping process is its simplicity, as most of the tasks realized by the users concern the application description with his conventional environment. The required knowledge of SynDEx and the loader is limited to simple operations.

4. MPEG-4 DECODER PROTOTYPING OVER MULTI-C6X DSP PLATFORM

4.1 Material platform

The chosen material platform enables the user to obtain a coherent and modular target architecture. Our platform is made by a SUNDANCE motherboard and two SMT335 TIM modules. The SMT335 TIM (Fig 5) consists of a Texas Instruments TMS320C6201 running at 200MHz. Modules are populated with 512KB of synchronous burst SRAM and 16MB of synchronous DRAM, giving a total memory capacity of 16.5MB.

Fig 5 : Sundance SMT 335 TIM module

A Field Programmable Gate Array (FPGA) is used to manage global bus accesses and implement six communication ports (20 MB/s). A Field Programmable Gate Array (FPGA) is used to manage global bus accesses and implement six communication ports (20 MB/s) and two Sundance Digital Buses (SDB). SDBs are 16-bit data parallel links achieving high-speed data transfers (200 MB/s each), an important point in regards with the large quantities of data (images and relative data) needed in video coding.

4.2 Modules and application libraries

We have created video Mpeg-4 texture decoding libraries made of modules and applications with AVS. Each application is built with modules, which can be involved in several applications, for instance coding and decoding processes. Each module executes a C routine like motion estimation, IDCT, inverse quantification. A module can also be built as a macro, composed of several other modules.

Mpeg-4 natural texture coding tools divide pictures into macroblocks, which are 16x16 of Y channel and the corresponding 8x8's in both U and V (chromatic components are sub-sampled by two). Mpeg-4 bitstream has to be demultiplexed (Fig 6), giving coded values for each block of each macroblock of the pictures. Those values are first decoded using variable length code tables defined in the standard. The resulting one dimensional data is then converted into a two-dimensional array of coefficients using the appropriate inverse scan table. The selection of the prediction direction is based on the comparison of the horizontal and vertical DC gradients around the block to be decoded. This direction is used for the prediction of AC and DC coefficients. The reconstructed DCT coefficients are obtained by the mean of the inverse quantization of the two-dimensional array of coefficients. The IDCT calculation is finally used, given texture blocks for the video object plane reconstruction.

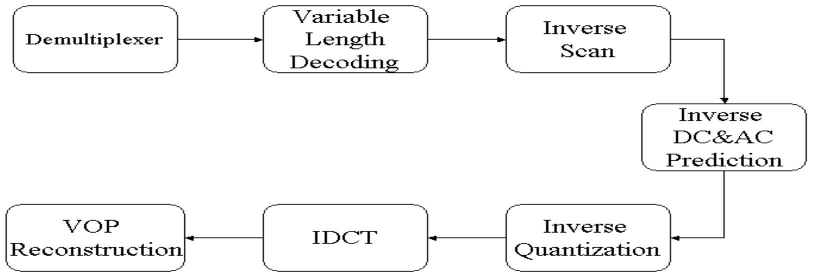


Fig 6 : Mpeg-4 texture decoding process

The first AVS modules created allow to read and manipulate Mpeg-4 visual bitstreams. A first module handles the bitstream, finds start codes and separates configuration information and elementary streams. On the basis of the coded configuration information, several AVS modules extract mpeg-4 configuration variables. The names of modules, and configuration variables on AVS libraries match the description given in Mpeg-4 documents.

Other AVS modules, from both coded elementary streams and configuration variables, realize the decoding operations : inverse VLC, DC prediction direction, inverse scan, inverse AC and DC coefficients prediction, inverse quantization and IDCT. Each module is optimized for the decoding of intra macroblocks, and for a VLIW implementation : interleaving loops, conditional tests leading to pipeline rupture, dynamic allocations and file manipulations are avoided. This high level development is made easier thanks to AVS visualization and debugging tools.

Mpeg-4 algorithm descriptions are given at a block level. So the standard describes the previous neighboring blocks used in coefficient predictions at a block level (Fig 7.a). But algorithms have to be repeated for the four luminance blocks and two chrominance blocks of each macroblock. The functional data flow graph of the decoding process at the macroblock level, using hierarchy, enables to find parallelism between those tasks. Indeed, one processor can handle the treatment of chrominance blocks, while another processor handles luminance blocks for instance. So we defined the adequate previous neighboring blocks for the prediction at the macroblock level (Fig 7.b).

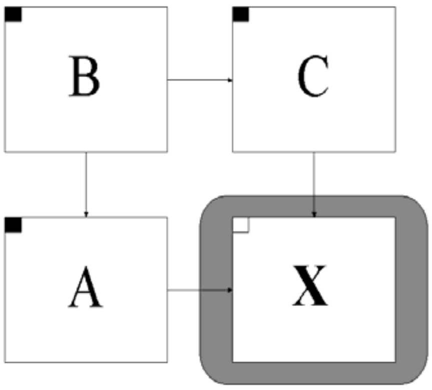


Fig 7a : previous neighboring blocks at the block level

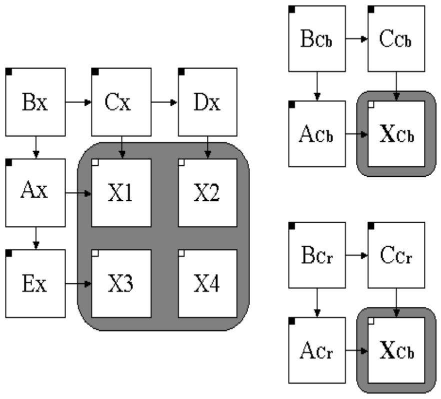


Fig 7b : previous neighboring blocks at the macroblock level

Because of the several prediction schemes, many relative data have to be memorized for each block and each macroblock. Usually software solutions developed store all those data for every blocks and every macroblocks. Resulting source codes need huge resource implementations, whereas the decoding process just need a few of them at a given time : macroblock relative information storage can be limited at three previous block lines (one line for each color component) and four previous blocks in the current line (Ax, Ex, Acb and Acr Fig 7b). This observation leads us to create AVS memorization modules for an optimized block relative information storage.

Mpeg-4 libraries can be used in all applications with natural intra coding or decoding textures, that is to say in most Mpeg-4 profiles. Functionalities have been checked with appropriated bitstreams given in the conformance testing part of the Mpeg-4 standard.

4.3 Monoprocessor implementation

We first implement the developed modules onto a single C6201 DSP. The code is automatically generated thanks to the SynDEx code generator. Chronometrical reports are used to value the time spend for each module. Fig 8 gives the results for decoding operations at the macroblock level (MacroblockI and Inverse VLC) or block level (the others). Those results are first chronometrical reports, the C source code was developed on the AVS environment for the functional verification but can be optimized for C6X [9].

The global decoding process for a QCIF Intra-coded Video Object Plane is then 128,6 ms, including the decoding of the VOL and VOP parameters (1.74 ms).

Functions	Times (microsec)
MacroblockI()	6.3
Inverse VLC	340
DC prediction direction	3.5
Inverse scan	9.3
Inverse AC&DC prediction	47.5
Inverse Quant	66.5
Inverse DCT	5.72

Fig 8 : chronometrical reports

4.4 Multi-DSP implementation

Times founded in the monoprocessor implementation are reported in the SynDEx application description. Syndex finds then the best matching between the algorithm and our architecture. Fig 9 shows a zoom on the resulting timing graph. We can see the partition of a single macroblock calculations on our two C6x DSPs. Luminance and chrominance macroblock operations can be executed separately, X2 and X3 blocks of a same macroblock (Fig 7) too. For each macroblock, the six IDCT calculations can be handled at the end of the macroblock treatment.

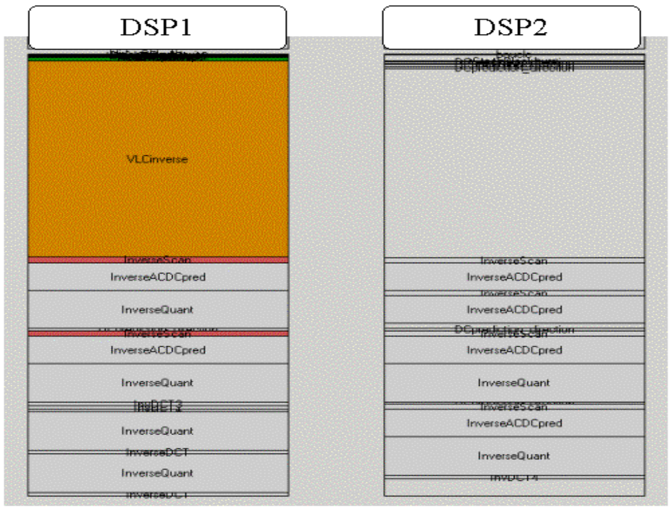


Fig 9 : Mpeg-4 SynDEx resulting timing graph

The resulting time spent to code a QCIF picture is now 82.93 ms instead of 128,6 ms for the monoprocessor implementation, that is to say an 1.55 speed up factor. We can easily see on the timing graph (Fig 9) that DSP2 is waiting for the end of the VLC calculation performed on the DSP1. With an AVS Inverse VLC data flow graph more precise, this calculation would have been shared between the two processors leading to a speed up factor near 2.

In view of the fact that decoding a Predicted Video Object Plane (P-VOP or B-VOP) is faster than the decoding of an I-VOP, future developments will give higher real time performances. Texas Instruments C6201 used for the implementation are running at 200MHz, but the code generated can directly be projected onto new C64x DSPs running at 600 MHz.

5. CONCLUSIONS AND PERSPECTIVES

This paper shows the efficiency of the AVSynDEx methodology for multiprocessor platforms. Modules developed for the Mpeg-4 decoding application presented here can be re-used for other Mpeg applications, with the possibility to adapt them and connect each other at a high level. Functionality of a new application can be checked with interactive visualization, graphics features and debugging tools provided by AVS. Next software development will concern the natural and synthetic video coding of Mpeg-4.

To create a digital image processing line, as this Mpeg-4 application, the whole process, starting from the AVS input description to the final execution onto the multiprocessor architecture, is practically automatic. The short implementation time enables the user to do more modifications at a high description level or to change the final partitioning. Our global process enables the user to develop complex applications onto a complex architecture, without any implementation pre-requirements. So, it is quite architecture independent.

Additional logic is always added between two C6x DSPs for the communication and is often integrated in a FPGA. The implementation of elementary and regular operations, especially the IDCT, onto this material part would give higher performances. Since the methodology we used is coherent with our methodology developed for mixed architecture (DSP + FPGA) [8], we plan to implement those operations onto the material part of our platform.

The Sundance platform used can be improved by adding a frame grabber module. A camera, input for an Mpeg-4 coder, or a monitor, output for a decoder, can be connected to the multi-DSP platform. The Sundance platform is connected to the PC processor with a PCI link, like many other multi-DSP platforms. We are studying how to add the PC processor and the PCI media on the SynDEx architecture graph. The algorithm implementations will take advantages from standard PC processor performances and connections : cameras, viewers, user interaction tools, networks but also other multi-DSP platforms.

Acknowledgments

The authors wish to thank the following people for their contribution to the project : Marie Babel (IERT), Yves Sorel (INRIA Rocquencourt). This work has been partially supported by Mitsubishi Electric ITE-TCL, Rennes, France.

References

[1] V. Fresse, M. Assouil, O. Deforges : *Rapid prototyping of image processing onto a multiprocessor architecture*, DSP World ICSPAT, Orlando, Florida, USA, November 1-4 1999.

[2] Signal Processing : Image communication, *special Mpeg-4*. Published by Elsevier Science B.V., January 2000.

[3] C. Miro, A. Lafage, Q.L. Nguyen-Phuc, Y. Mathieu, "Hardware Implementation of Perspective Transformations on Mpeg-4 Video Objects", Proceedings of SPIE, Volume 3655, Media Processors 1999, pp. 102-112.

[4] P. Ruetz, P. Tong, D. Bailey, D. A. Luthi, P. H. Ang : *a high-performance full motion video compression chip set.* IEEE Trans. Circ And Syst. For Video Technol., vol. 2, pp. 111-122, June 1992.

[5] International AVS Center, Manchester Visualization Centre, Manchester Computing, University of Manchester. Available at http://www.iavsc.org.

[6] C. Lavarenne, Y. Sorel : Specification, performance optimization and executive generation for real-time embedded multiprocessor application with Syndex , Proc. Of Real Time Embedded Processing for Space Applications, CNES International Symposium.

[7] Y. Le Mener, M. Raulet, J-F. Nezan, A. Kountouris, C. Moy : *SynDEx executive kernel development for DSPs TI C6X applied to real time and embedded multiprocessors architecture.* EUropean Symposium (EUSIPCO), Toulouse, France, September 2000.

[8] V. Fresse, O. Déforges, J.F. Nezan : *Rapid prototyping for multi-DSP and FPGA architectures: implementation of digital image processing applications by means of AVSynDEx* The European Association For Signal, Speech and Image Processing (EURASIP) Journal on Applied Signal Processing, Implementation of DSP and Communication Systems Special Issue, Oct 2002.

[9] Texas Instruments technical documents: *TMS320C6000 programmer's guide*. Ref spru198f. Available at http://www.dspvillage.ti.com. Feb 01.