# RAPID PROTOTYPING FOR AN OPTIMIZED MPEG4 DECODER IMPLEMENTATION OVER A PARALLEL HETEROGENOUS ARCHITECTURE

*Ventroux N.[1], Nezan J.F.[1], Raulet M.[1,2], Déforges O.[1]*

[1] CNRS UMR 6164 IETR laboratory / INSA Rennes
20, av des Buttes de Coëmes, CS 14315, 35043 Rennes Cedex, France
[2] MITSUBISHI ELECTRIC ITE-TCL, 1 allée Beaulieu, CS 10806, 35708 Cédex 7, Rennes, France
Contact : {jnezan, mraulet, odeforge}@insa-rennes.fr

## ABSTRACT

Sequential Mpeg-4 solutions actually developed for single processors try to integrate the most functionalities as possible in an unique software, and are generally oversized compared with the actual service requirement. Moreover, they can hardly be projected onto multiprocessors targets, leading to an extra load of source code and calculations, but also to a sub-optimal use of the architecture parallelism. This paper introduces a distributed Mpeg-4 application, where the system part is hosted by a standard PC, and the video decoder is supported by a multi-DSPs board. In particular, we present our AVSynDEx methodology allowing both an incremental building, an easy update on the video decoder description, and a quasi-automatic implementation onto a multi-C6x platform.
We also define a global scheduler managing the parallel execution of the video and system applications.

## 1. INTRODUCTION

Multimedia applications introduce a much higher degree of complexity to traditional digital signal algorithms as they manipulate sounds, images and video from both natural and synthetic origins. The required computational performances should always be improved for a real time use.

Mpeg-4 is a new multimedia standard adopted by the Moving Picture Experts Group (MPEG) [1]. It defines not only the way to decode audio and video, but also how to represent, combine and synchronize them in the bitstream. An Mpeg-4 coder/decoder can then be divided in ten main parts (system, visual and audio for instance) with different timing performances requirements and execution behaviors. The system part is event driven performing interactions with the user or channels, but demanding a low computation power. On the other hand, audio and more particularly video parts can be characterized by a fix data driven scheduling, while requiring high performance processing. In terms of Real Time Operating System (RTOS), the system part can be supported by a preemptive on-line scheduling, while audio and video can be more efficiently implemented by a non-preemptive off-line one [2].

This paper presents a partial solution for a whole Mpeg-4 decoder distributed over a PC-Multi-DSPs architecture. An optimized video decoder has been completely implemented onto the multi-DSPs board, through our prototyping methodology depicted in the following. A simple system runs on the PC, providing bitstreams and displaying rebuilt images. The global application is then distributed and supported by an off-line scheduling. Communication and synchronization primitives via PCI bus have been developed to insure a parallel execution.

## 2. MPEG4 ORGANIZATION AND DESIGN

Mpeg-4 [3] is a toolbox which can address various services at different bitrates and complexity. In a real time context, calculations and resources have to be minimized by restricting the code to its necessary requirements. Mpeg-4 introduces the concepts of complexity and services through profiles and levels. A profile is a defined subset of the entire bitsream syntax. For instance, a profile can deal with visual documents, another one with audio data. For each profile, functionalities and complexity are fixed by a level, defined as a set of constraints imposed on parameters in the bitstream. So Mpeg-4 diagrams can be optimized and adapted to the application according to a specific profile@level.
Data flow graphs (DFG) are modular, meaningful ways to design audio and video parts. The construction of libraries composed of basic functions (bitstream reading, DCT, …), allows an iterative bottom-up description, and well adapted to a defined profile@level. Moreover, a DFG exhibits the potential parallelism, and the data

dependencies given by the arcs between nodes are sufficient to express the whole scheduling of this class of applications. A DFG is then not only a suitable solution from a functional point of view, it also constitutes the description input for our prototyping process.

## 3. AVSYNDEX METHODOLOGY

AVSynDEX [4] is a full rapid prototyping going from a DFG functional description of the application to the multi-components implementation.

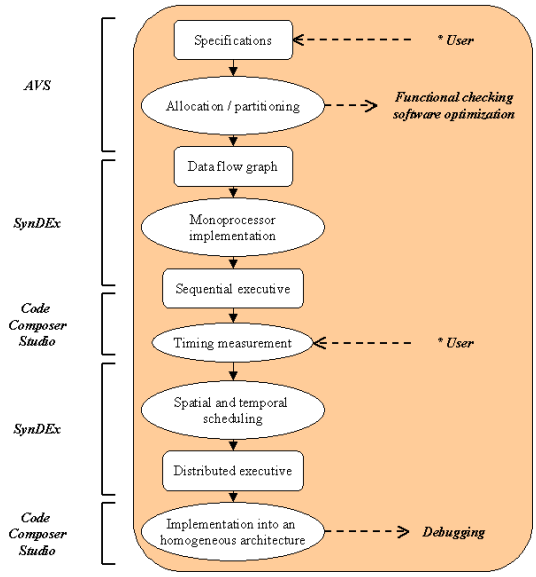


**Figure 1** : AVSynDEx methodology

The digital image designer creates the data flow graph by means of the graphical development tool AVS (**A**dvanced **V**isual **S**ystem). This CAD software enables the user to achieve a functional validation of the application. Then, a quasi-automatic translator converts this representation into a new data flow graph compatible with SynDEx (**Syn**chronised **D**istributed **Ex**ecutive). This last tool schedules and distributes the data flow graph according to the parallel architecture and generates an optimized distributed executive. This executive is loaded onto the platform by using the classical loader of the target processor. A first mono-processor implementation is necessary in this process to perform chronometrical measures of each task of the graph. Then each node of the DFG can be valued by its execution time, so that SynDEx is able to find the best distribution of the application. The process is summarized figure 1.

### 3.1. AVS

AVS is a high-level environment for the development and the functional validation of graphical applications [5]. It provides powerful visualization methods. The AVS environment contains several module libraries dedicated to the application developments. An application is constructed as a DFG by inserting existing modules or user modules into the workspace, and linking their inputs and outputs. Each module can calls a C, C++ or Fortran function and the associated library files. Only C functions are considered here. Hierarchical representations are also supported by AVS. Arcs of the DFG express both data transports and execution ranking of the nodes.

A main advantage is the automatic visualization of intermediate and resulting images at the input and output of each module. This characteristic enables the image-processing designer to check and validate the functionality of the application before the implementation step.

### 3.2. SynDEx

This free tool is an academic environment designed and developed at INRIA Rocquencourt France and several national laboratories take part in this project as we do. SynDEx is an efficient environment, which uses the AAA [6] methodology to generate a distributed optimized executive dedicated to parallel architectures. The purpose of this methodology is to find the best matching between an algorithm and a specific architecture while satisfying constraints. This methodology is based on graph models to exhibit both the potential parallelism of the algorithm (algorithm graph) and the available parallelism of the hardware architecture (architcture graph). The result of graph transformations is a latency optimized distributed executive. Results can be visualized and analyzed thanks to a timing diagram.

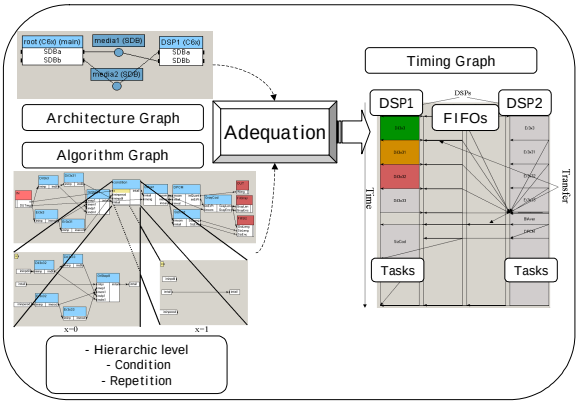


**Figure 2** : SynDEx description graphs and the resulting timing diagram

SynDEx avoids the use of a RTOS like 3L diamond product, implementing the minimum custom-built off-line scheduling. So spatial and temporal additional costs are minimized. Moreover, the order of the algorithm tasks is guaranteed and dead-locking is avoided.

### 3.3. Material platform

Our target architecture is made of a sundance motherboard with two SMT335 TIM modules. Each module contains a Texas Instrument TMS320C6201 running at 200MHz. An FPGA manages global bus accesses and implements six communication ports (20 MB/s each) and two Sundance Digital Buses (SDB) achieving high-speed data transfers (200 MB/s each). The mother board owns a 32 bits PCI bus (PCI-X 2.0 version, 33 MHz) in order to dialog with the PC-Host, but it is generally used as a stand-alone platform.

SynDEx generates a macro-code independent of the material target. Communication and synchronization primitives have to be defined for each processor type. Then, the M4 macro-processor transforms the macro-code into a compilable one. In order to get an automatic implementation onto the multi-DSPs architecture from SynDEx, we have realized these primitives for TMS320C6201 DSP and for our platform (management of SDB buses).

## 4. MPEG-4 DECODER

### 4.1. Application representation

We realized a video Mpeg-4 natural texture decoder [7]. The granularity level of the description has an important impact on the final implementation. Mpeg-4 natural texture coding tools divide pictures into macroblocks, composed of four 8x8 blocks of Y channel, and the associated 8x8 blocks of chromatic component (U/V). All the Mpeg-4 algorithm descriptions are given at this block level. Therefore, this is also the finest granularity adopted in our design. The hierarchical representation involves four main layers :

- *block-level* : expressing the sequence of processing from the multiplexed bistream to the final decoded block (see figure 3).
- *macroblock-level* : realizing the macroblock decoding by 6 block-level instances with an independent order (U/V) or a dependent one (Y),
- *image-level* : reconstructing a decoded image by a temporal recursive iteration of macroblock-level graph,
- *sequence-level* : reconstructing a decoded sequence by a temporal recursive iteration of image-level graph.

Both AVS and SynDEx support conditional node executing one or another sub-graph depending on a condition. For example, the type of image I (intra) or P (predicted) can address a specific optimized sub-graph.

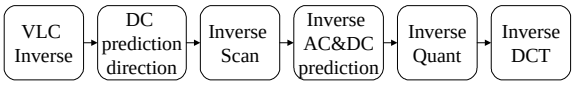


**Figure 3** :Block-level decoding process

### 4.1. Mono and multi-DSPs implementations

A first mono-processor has been automatically generated from SynDEx onto a TI C6201 DSP. The decoder has been checked with the set of video sequences provided by the Mpeg-4 part 4 document (conformance testing). The average decoding time is 95 ms for a CIF image (288x354), and 35 ms for a QCIF one (144x177), while conformance tests specify the upper limit of real time processing to 100 ms for a CIF, and 40 ms for a QCIF image.

The prototyping process implements directly the C functions used at the functional checking. We have only tried to optimize the mapping of the code onto the VLIW architecture of C6X by avoiding interleaving loops, conditional tests and dynamic memory allocation. The timing diagram of SynDEx exhibits bottleneck of the application, and optimization at the function-level can be done. For instance, we have exchanged our IDCT function by a fast version one available in Texas Instrument libraries.

By reporting the chronometrical measures of each task, SynDEx realizes the partitioning over the two DSP. The best distribution is obtained by taking advantage of both spatial redundancy of the graph (several block-level sub-graphs at the macroblock-level) and temporal redundancy (recursive iterative macroblock-level graph). The speed-up factor is 1.82 (upper limit : 2) compared to a mono-processor implementation.

## 5. DISTRIBUTED HOST-MULTI-DSPS APPLICATION

### 5.1. Global application scheduling

The PCI bus of the mother board can be used in order to exchange data and to share processing with the host. We designed low level synchronized communication primitives. A global off-line scheduler of the distributed application (system+video) can be easily deduced (ref. fig 4). Actually the system part consists only in reading the whole sequence bitstream, sending the bitstream to the DSP board, receiving the series of decoded image and displaying them. The integration of a full system will only modify the bitstream capture task, remaining the global behavior.

The host software is a Visual C++ application. Communications sequences have been added manually in

both the PC and multi-DSPs codes. The display uses the Microsoft DirectX library which permits high-speed accesses to the video card. Nonetheless, frame rates depend on graphics acceleration capabilities of the card. With an old-technology video card, time for formatting and displaying a CIF picture is about 40 ms. Bitstream decoding and image display are done simultaneously, ensuring Host and DSP's parallelism execution.

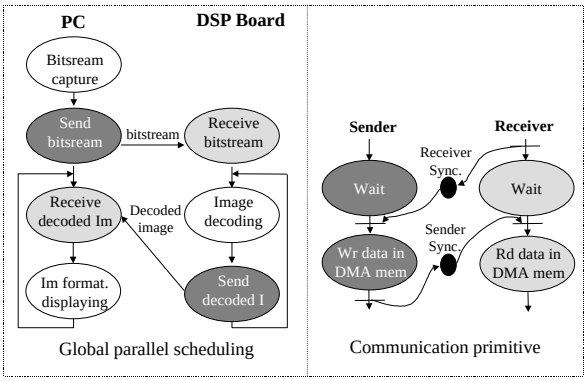


**Figure 4** :Synchronized distributed execution and communication primitives

### 5.2. Communication primitives

A maximum DMA transfer rate of 30 Mbytes/s can be reached, but, the synchronization of the Host and DSP must be first ensured. Thus, each data transfer enclose two synchronizations. The receiver must first prevent the sender that he is ready for receiving data. Then, the sender writes new data in the DMA memory and prevents the receiver when finished. Finally, the receiver recovers data in the DMA memory (Figure 4). PCI communication links have been modeled and maximum transfer rates for sending a CIF picture is about 6.7 ms (15 Mbytes/s).
A protected OS like Microsoft Windows does not allow to access directly to hardware components. We used WinDriver development toolkit to develop our drivers. A main advantage is that applications based on a WinDriver kernel can easily be used under different OS.

Both host and DSPs could be modelized in a unique architecture graph with SynDEx, as well as the PCI bus media. Input and output of the video decoder graph can be then map to the host, and global scheduler including communication links could be automatically generated.

## 6. CONCLUSION AND PERSPECTIVES

This paper has presented a distributed Mpeg-4 decoder implementation along with its design methodology. The system part can be supported by the host processor with a traditional operating system while the video and audio parts are more efficiently implemented using both DSP processors and an off-line scheduling. The multi-DSPs platform can then be seen as a PC co-processor. The video decoder has been developed and integrated thanks to our prototyping process, allowing both a quasi-automatic distributed implementation, and a minimal embedded code.

Low level primitives of synchronization and data communications via PCI bus between the host and the DSP platform have been designed. Then, the two applications (system + video decoder) can be concurrently executed and synchronized by a global off-line scheduler.

Next developments will concern the integration of a full standard system on the PC and new video features on the multi-DSPs board. The PC and PCI media have also to be added on the SynDEx material graph, in order to get an automatic generation of the global scheduler.

**Acknowledgments** : This work is partially supported by Mistubishi ITE in Rennes.

## 7. REFERENCES

[1] JTC1/SC29/WG11. "Mpeg-4 applications. Technical Report N2724", *ISO/IEC*, Octobre 1999.

[2] L. A. Hall, D. B. Shmoys, J. Wein, "Scheduling To Minimize Average Completion Time: Off-line and On-line Algorithms", *Proceedings of the 7th ACM-SIAM Symposium on Discrete Algorithms*, January 1996, pp. 142--151

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

[4] V. Fresse, O. Déforges, J.F. Nezan, "AVSynDEx: A Rapid Prototyping Process Dedicated to the Implementation of Digital Image Processing Applications on Multi-DSP and FPGA Architectures, *EURASIP journal on Applied Signal Processing, special issue on Implementation of DSP and Communication Systems*, No. 9, September 2002, pp 990-1002.

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

[6] T. Grandpierre, C. Lavarenne, and Y. Sorel, "Optimized rapid prototyping for real-time embedded heterogeneous multiprocessors", *Codes'99 7th International Workshop on Hardware/Software Co-Design*, Rome, May 1999.
http://www-rocq.inria.fr/syndex/

[7] J.F. Nezan, M. Raulet, O. Déforges, "Integration of Mpeg-4 Video Tools onto Multi-DSP Architectures using AVSynDEx fast Prototyping Methodology", *IEEE Workshop on Signal Processing Systems (SIPS'02)*, San Diego, October 16-18, 2002.