APPLICATION NOTE : INTEGRATION OF MPEG-4 VIDEO TOOLS ONTO SUNDANCE MULTI-DSP ARCHITECTURES

J.F. Nezan, M. Raulet, O. Déforges

IETR (CNRS UMR 6164)
INSA Rennes
20 av des Buttes de Coësmes
35043 Rennes Cedex
France

ABSTRACT

Real-time signal, image and control applications have very important time constraints, involving the use of several powerful numerical calculation units. The aim of our work is to develop a fast and automatic prototyping process dedicated to parallel architectures made of both PC and several last generation Texas Instruments digital signal processors: TMS320C6X DSP. Our methodology provides an alternative for 3L users who need to develop static executives instead of dynamic ones. The process is based on SynDEx, a CAD software improving algorithm implementation onto multi-processor architectures by finding the best matching between an algorithm and an architecture. SynDEx kernels have been developed for automatic PC and DSP dedicated code generation with new SynDEx features. A full Mpeg-4 coding/decoding application shared onto PC, Sundance SMT335 and SMT361 illustrates the results.

1. INTRODUCTION

Although new mono-processor architectures (Personal Computer) provide ever-increasing computational power, they cannot cope with the ever-increasing complexity of some control, signal and image processing applications. Parallel architectures are needed to satisfy real-time constraints (computation load balancing), as well as to take into account the distributed nature of the resources (sensor/actuator, computation, memory) of real-time applications.

The goal of a prototyping methodology is to go from a high level description of the application to its implementation onto a target architecture. Performances of the process can be evaluated by different aspects:
- maximal independency with regards to the architecture,
- possibility to handle heterogeneous multi-component architectures,
- maximal automation during the process (partitioning, code generation),
- efficiency of the implementation both in terms of executive time and resource requirements.

This paper presents a prototyping methodology based on SynDEx which satisfies the previous criteria. Although SynDEx is basically a CAD tool for partitioning and for code generation, we demonstrate that SynDEx can also be directly used as the front-end of the process for functional check. Furthermore, we show that data parallelism can be easily highlighted at a high level of description for distributed implementation purposes, through new features of SynDEx associated with the data flow graph formalism. The code automatically generated by SynDEx is static, leading to an optimal use of the DSP and its DMA transfers. The off-line scheduling is specially suitable for fix data driven scheduling applications like video algorithms. The generated code is compiled and loaded on the platform directly from Code Composer Studio. So that the developer can control data transfers, synchronizations and can also modify them directly if desired. The time to market is reduced allowing algorithm descriptions fast reuse. An application can directly be projected onto several multi-DSP architectures.

The methodology is illustrated with an Mpeg-4 video coder/decoder automatically implemented onto PC + Sundance Multi C6x platforms.

2. FAST PROTOTYPING METHODOLOGY

2.1. SynDEx V6 software

SynDEx (www-rocq.inria.fr/syndex/) is an academic system level CAD software freely downloadable and developed in INRIA Rocquencourt, France. It supports the AAA methodology Adequation Algorithm Architecture) for distributed processing.

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 and the available parallelism of the hardware architecture. A SynDEx application (fig.1) is made of an algorithm graph (operations that the application has to execute) which specifies the potential parallelism, and an architecture graph (multi-components target, i.e. a set of interconnected processors and specific integrated circuits), which specifies the physical parallelism.

”Adequation” means an efficient mapping. Performing an adequation consists in executing heuristics which seek for an optimized implementation of a given algorithm onto a given architecture. The result of graph transformations is a latency optimized distributed executive. An implementation consists in distributing (allocating parts of algorithm onto components) and scheduling (giving a total order for the operations distributed onto a component) the algorithm onto the architecture. The scheduling is performed off-line. Results can be visualized and analyzed thanks to a timing diagram (fig.1).

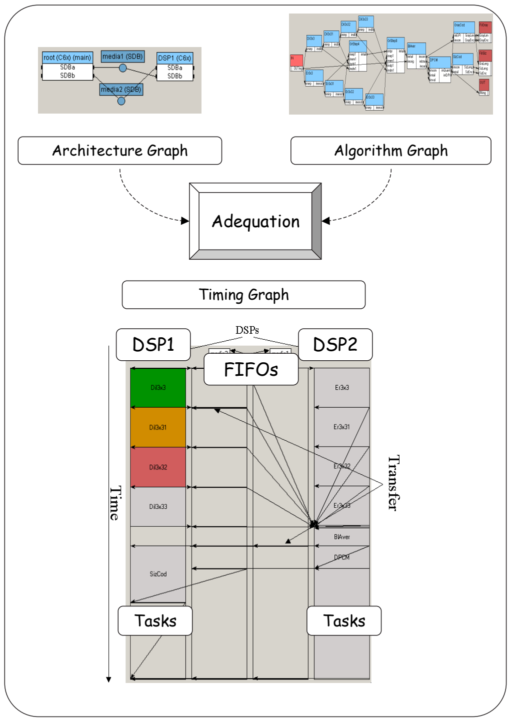

Fig. 1. SynDEx description graphs and the resulting timing diagram.

2.2. Automatic Executive Generation

The goal of SynDEx is to directly perform an implementation from an algorithm specification. It automatically generates a “generic executive” independent of the processor target into several source files (fig.2), one for each processor. Moreover, it generates automatically a makefile for automating the specific compilation chain of the architecture. These generic executives are dedicated to the application without any support of RTOS. They are composed of a list of macro-calls. The macro processor M4 transform this list of macro-calls into compilable code for the specific processor target. It replaces macro-calls by their definition given in the corresponding “executive kernel”, which is dependent of the processor target.

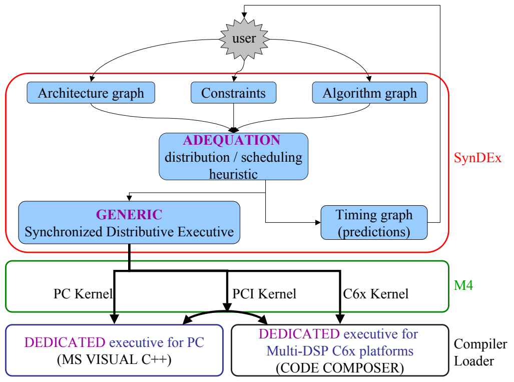

Fig. 2. SynDEx utilization global view

SynDEx kernels are available for instance for the following processors: Analog Device ADSP 21060, SHARC, Motorola MPC 555 et MC 68332, Intel i80x86 et i8096, Unix/Linux workstations, Texas Instruments TMS320C40. We will introduce in this paper several kernels developed for C6x automatic code generation.

2.3. Methodology

Our previous prototyping process integrated AVS as a front-end. AVS is a software designed for algorithm data flow graph (DFG) specification and simulation. The DFG was validated thanks to AVS visualization tools and was automatically transformed to be compliant with SynDEx algorithm input.

The work presented here is based on the use of SynDEx for the specification and for the simulation of the algorithm graph without any commercial tool like AVS. Hence, SynDEx allows the full rapid prototyping starting from the application specification (DFG) to the final multiprocessor implementation (fig.2) in four steps:

Step 1: the digital image designer creates the DFG of his application with SynDEx. The automatic code generation provides a standard C code for a single PC implementation. In this way, the user can design each C function associated with each vertex of its DFG, and can check the functionalities of the complete application with standard compilation tools like Visual C++. The automatic code generation allows the use of visualization primitives (instead of AVS visualization tools) for an easy functional check of image and video algorithms.

Step 2: the DFG developed is then used for the automatic prototyping onto monoprocessor targets with chronometrical reports inserted automatically by the SynDEx code generator. The execution duration associated to each function (i.e vertex) executed on each processor of the architecture graph (PC and C6x) is automatically estimated through dedicated temporal primitives.

Step 3: the user can easily use these durations to characterize the algorithm graph by entering these values in SynDEx.

Step 4: SynDEx generates a real-time distributed and optimized executive, where chronometrical report are not inserted, 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, because most of the tasks realized by the user concern the application specification with his usual compiling environment. The required knowledge of SynDEx and of compilers (Visual C++ and Code Composer) is limited to simple operations. All complex tasks (adequation, synchronization, data transfers and chronometrical reports) are executed automatically.

3. SYNDEX KERNELS DEVELOPED

Video algorithms can be characterized by a fix data driven scheduling, while requiring high performance processing. SynDEx does not need any RTOS like 3L-diamond product. It generates dedicated executives based on off-line scheduling leading to the minimum overhead in space and time. Moreover SynDEx avoids inter-processor communication deadlocks. We have developed SynDEx executive kernels for the C6X DSP family, in order to automatically generate a distributed and optimized static executive of the specified algorithm onto those processors and to automatically generate data transfers and synchronizations between several C6x and PC.

3.1. Sundance Platforms

The chosen platforms enable the user to obtain a coherent and modular target architecture. A first platform is composed of a Sundance motherboard and two SMT335 TIM modules. The SMT335 TIM consists of a Texas Instruments TMS320C6201 (64 KB of internal program memory, 64 KB of internal data memory) running at 200MHz. Modules are populated with 512KB of synchronous burst SRAM and 16MB of synchronous DRAM.

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 their related data) needed in video coding.

A second platform with two SMT361 is used. Each SMT361 handles a TMS320C6416 running at 400Mhz, and can be connected with communication ports and SDB the same way as SMT335.

3.2. C6x characteristics

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. The several peripherals available for the C6x devices, such as various memory configurations, ports, timers, direct memory access, and power down logic, make it very interesting in real time and embedded parallel architectures. C6x peripherals are fully programmable in C language.

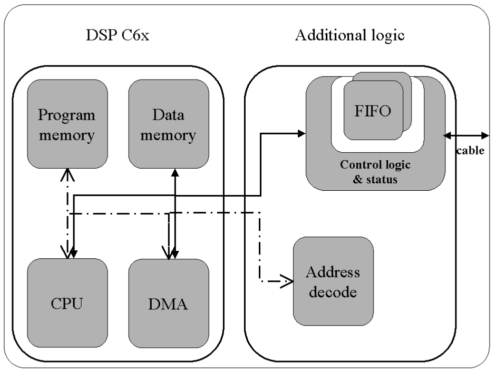

Fig. 3. Additional logic for C6x communications

C40 DSPs contain communication ports (CPs), providing inter C40 communication. This specific interface allowed to construct parallel processor systems very easily. C6x DSPs no longer contain CPs. Manufacturers must insert additional digital resources between two C6x in order to make their communication possible (fig.3). So, inter C6x communications are architecture dependant. This, and because it was written in the C40 specific assembly language, make that the SynDEx C40 executive kernel can no longer be used for C6x.

Additional logic added by manufacturers is generally a FIFO buffer which communicate with another FIFO connected to the other DSP. The size, the control and status information of FIFOs can change from a platform to another one. A FIFO can create interruptions in the DSP program, for instance at the end of a transfer. The number and the meaning of those interruptions can change too. FIFO's specifications are often done by the manufacturers with examples. Automatic code generation can not be architecture independent. Developed kernels are available for Sundance SMT335 and SMT361.

3.3. C6x Kernel

We have created M4 libraries for C6x forming the SynDEx C6x Kernel. By this way the executive generated by SynDEx can be translated by the macro-processor GNU M4 [6] into a C source code for C6x. The main kernel is called C6x kernel, and communication kernels (SDB, CP, TCP and PCI are developed) have to be developed for a specific communication link.

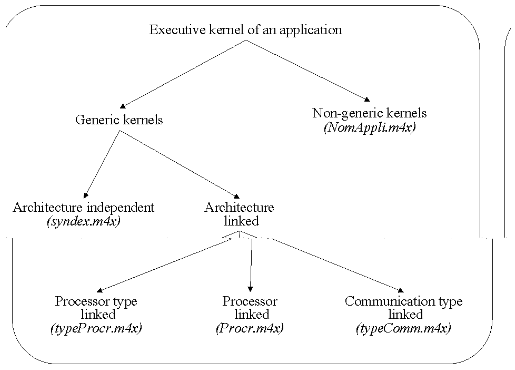

Fig. 4. Executive kernel organization

Code Composer Studio software accepts C source code and produces assembly language source code. Thanks to its optimizer, the programmer can reach high-performances without using assembly language. We decided to develop C6x Kernel in C language, in order to make it partly reusable for any C programmable DSP, and because it is not an important waste of time on a complete application.

C6x kernel organization :
The C6x executive kernel has been divided into several libraries (fig.4), enabling its easy adaptation to a new architecture. A non-generic library contains M4 macros for the application, such as specific input/output functions. Architecture independent library contains macros used whatever the architecture target. The other libraries connected with processor type, processor or communication type dependent are architecture dependent. We developed kernels for two different platforms to ensure that macros can be reused. The adaptation of our work for another multi-C6x architecture is limited to the communication sequence adaptation for a new media, if needed. C6x kernels for our Sundance platforms are available.

Calculation and communication parallelism :
SynDEx macrocode creates two interleaved schedulers: one for computation tasks and the other for communications, allowing parallelism of those actions. We have chosen the use of multi-channel DMA transfers (fig.5), maximizing this parallelism and timing performances. The C6201 DMA contains four different channels, we have linked each

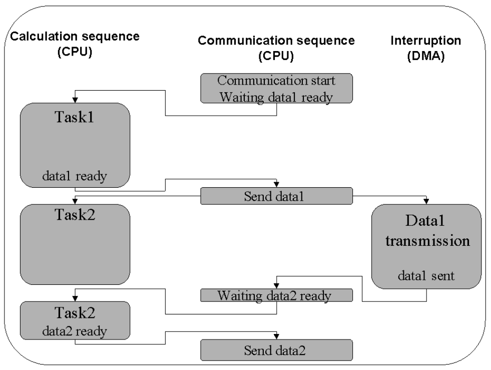

Fig. 5. Parallelism between calculations and communications

of them to one SynDEx communication media. By this way SynDEx architecture graph can have four connections for each C6x, for instance two SDB and two CP links. Our work with the C6416 remains four communication links because the SMT361 have ”only” two SDB, but an extension will be possible to manage more communication media.

We have created communication sequences for each different media, but they are always structured as follow :

- Computation of the number of packets (For a communication, we have to split data into FIFO length packets),
- FIFO selection (depending on the FIFO),
- DMA Configuration,
- Permission of interruption (depending on the FIFO),
- DMA start.

chronometric reports :
The C6x kernel contains macros for the automatic code generation of chronometric reports. It is based on the use of the internal timer of the C6x for a maximum precision. This feature is available for both SMT335 and SMT361.

PCI communications :
Sundance motherboards are PCI cards plugged in PC. The description of a target architecture is described with a graph in SynDEx (fig. 6). The motherboard PCI bus can be used in order to exchange data and to share processing with the PC. In this configuration, the multi-DSPs platform can then be seen as a *PC co-processor*. In terms of Real-Time Operating System (RTOS), the control part (file or network management for instance) can be supported by a preemptive on-line scheduling on the PC, while video algorithms can be more efficiently implemented by a non-preemptive off-line one on the multi-DSPs PCI board. We designed

low-level synchronized communication primitives for the SMT310 motherboard. Our works with the SMT320Q will be available soon.

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 encloses two synchronizations. The receiver must first warn the sender that he is ready for receiving data. Then, the sender writes new data in the DMA memory and warns the receiver when finished. Finally, the receiver recovers data in the DMA memory. PCI communication links have been modeled and maximum transfer rates for sending a CIF picture (Mpeg4 image format: 352*288 pixels) is about 6.7 ms (15 Mbytes/s).

A protected OS like Microsoft Windows do not allow accessing 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 an unique architecture graph with SynDEx, as well as the PCI bus media. Input and output of the video decoder graph can be then mapped on the host, and global scheduler including communication links could be automatically generated with the SynDEx PCI library included in the C6x kernel.

TCP communications :

A TCP kernel has also been developed for the automatic code generation of TCP communications. This kernel can be used for automatic code generation of communication between PC and multi-DSP platforms, but also between several PC. As a result the complete architecture graph shown figure 6 can be described in SynDEx. Two DSP (SMT335 or SMT361) communicate with a PC via the SMT310 motherboard PCI bus, PC itself connected to another PC via TCP (Ethernet network). Note that the second PC can also be connected to another Sundance platform. The code for this architecture will be automatically generated.

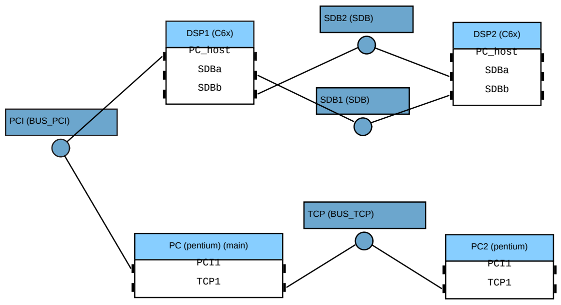

Fig. 6. SynDEx multi-PC-multi-DSP architecture graph

C6x, SDB, CP and TCP kernels have been tested with several applications. From a SynDEx algorithm graph, the code can be automatically generated in several configurations : mono and multi SMT335 connected with up to four links (2SDB and 2CP for instance). The use of multi-SMT361 is possible : SDB and CP kernels can handle EDMA transfers but optimizations are studied.

The PCI kernel is used to automatically generated communication between a PC and SMT335/SMT361 modules on an SMT310 platform. The PCI kernel have to be modified to handle SMT320Q motherboards.

The TCP kernel has been tested between two PC.

4. MPEG4 CODEC

We realized a video Mpeg-4 natural texture decoder. We have created video Mpeg-4 texture decoding libraries made of vertices and applications. Each application is built with vertices, which can be involved in several applications, for instance coding and decoding processes. Each vertex executes a C routine like motion estimation, IDCT, inverse quantification. A vertex can also be built with hierarchy levels, composed of several other vertices. 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.

Mpeg-4 bitstream has to be demultiplexed (fig. 7), 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.

The first vertices created allow to read and manipulate Mpeg-4 visual bitstreams (*"demux" and "configuration decoding"* fig. 7). A first vertex handles the bitstream, finds start codes and separates configuration information and elementary streams. On the basis of the coded configuration information, several vertices extract mpeg-4 configuration variables. The names of vertices, and configuration variables on libraries match the description given in Mpeg-4 documents.

Other vertices, 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

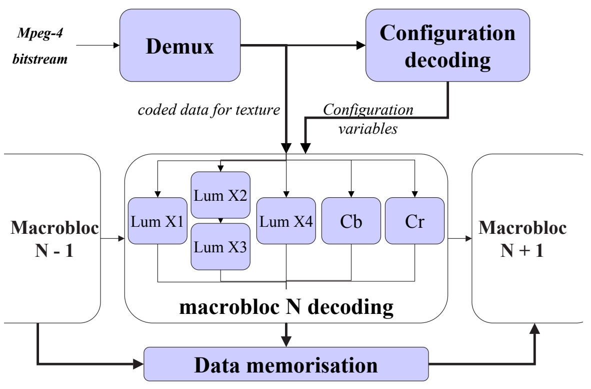

Fig. 7. hierarchical description of the Mpeg-4 decoder

IDCT. Each vertex 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 visualization tools.

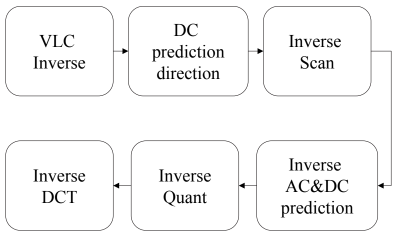

Fig. 8. Mpeg-4 8*8 bloc decoding

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. 8). 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 (*data memorisation* fig. 7).

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. This observation leads us to create memorization vertices 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.

SynDEx supports conditional vertices executing one or another subgraph depending on a condition. For example, the type of image I (intra) or P (predicted) can address a specific optimized sub-graph.

5. MPEG4 DECODING PROTOTYPING RESULTS OVER DSP

We first implement the decoder onto a single SMT335 and a single SMT361. The code is automatically generated from the SynDEx graphs. Chronometrical reports are used to value the time spend for each vertex. Table 1 gives the results for decoding operations at the macroblock level (MacroblockI) or block level (the others).

Functions	Times with an SMT335 micro-sec	Times with an SMT361 micro-sec
MacroblockI	3.2	0.556
InverseVLC	28.5	3.977
DC prediction direction	0.7	0.107
Inverse scan	3.8	0.472
Inverse AC-DC prediction	7.4	0.690
Inverse Quant	11.1	1.721
Inverse DCT	1.8	0.821

Table 1. Chronometric results.

The global decoding process for a QCIF (176*144 pixels) Intra-coded Video Object Plane is then 128,6 ms with an SMT335, including the decoding of the VOL and VOP parameters (1.74 ms for a C6201). The program size is smaller than 64 KByte and can be implemented onto internal memory of the C6201. As regards the data, the input coded bitstream and the decoded picture are stored in the external memory (SBSRAM) whereas all the others data are stored in the internal memory. Both data and programm are stored in internal memory when using the SMT361 and its C6416.

Manual optimizations have been performed to minimize the amount of data in the code automatically generated by SynDEx. A PhD is beginning in our laboratory to automate this step directly in the SynDEx code generator.

Results given with an SMT361 are achieved using several optimizations for the C6416 (400 MHz) which should be tested with an SMT335 for better results. At the moment the decoding time for CIF Intra-coded Video Object Plane (384 kbit/s) is then 18.1 ms with an SMT361. For lower bit rates the time of the inverse VLC function is lower and global decoding time can decreased reaching 11 ms.

Times reached with a first automatic code generation was not faster than a mono-processor implementation. It is due to the lack of potential parallelism of a decoding algorithm. In such a decoding most of the tasks have to be realized in a sequential order. A coarser granularity for the algorithm graph have to be tested to improve the adequation.

It is very important to underline that the parallelism in a decoder is not very important in comparison with a coder. Slices for instance can be used to split images into several parts each of them coded by a different DSP. In a project where both the coder and the decoder are controlled, slices can also be used to share the decoder between several DSP.

Currently P-VOP algorithm graph has been successfully implemented on a PC (functional verification step). Work in progress concern memory optimizations for the DSP implantation. The decoding of Predicted Video Object Plane (P-VOP or B-VOP) is known to be faster than the decoding of an I-VOP. So DSP developments will soon give better real time performances.

6. DISTRIBUTED CODEC PROTOTYPING RESULTS OVER PC AND PC/DSP PLATFORMS VIA TCP

In order to get a complete distributed codec, the Xvid Mpeg4 coder is used to provide our decoder a compressed bitstream. The aim is to test a full coding/decoding application where the coder is executed onto a single processor or DSP. In such a case, the methodology is useful for the coder/decoder synchronization and for automatic code generation of the compressed bitstream over TCP and PCI links. The use of a webcam at the coder input and display functions at the decoder output provide us a complete demonstration application.

A first functional check of codec was realized on a PC. Once the functional description is checked, the role of the PC is to code picture from the webcam and send the Mpeg-4 bitstream via TCP link to another PC. The second PC receives the data, send it to the sundance platform for decoding, recover the decoded pictures and finally display the video. The SynDEx algorithm graph used is basic so that the potential parallelism of the application is not yet highlighted. The result is a complete demonstration application with automatic code generation over several physical links. The developer does not have to worry about communication links, synchronizations, image grabbing and display.

7. CONCLUSIONS AND PERSPECTIVES

This paper has presented a distributed Mpeg-4 codec implementation along with its design methodology, allowing both an automatic distributed implementation and a minimal embedded code. A complete demonstration application has been developed and can be adapted to several new platforms (SMT319 or OMAP for instance).

This paper also describes visualization primitives that enable the use of SynDEx as the front-end of the prototyping process instead of AVS. The main advantage is to decrease the complexity of the complete process. Another advantage is that SynDEx can be freely downloaded whereas AVS is a commercial tool. Algorithm mapping, synchronizations and data communications are automatically generated and optimized for the PC-multi-DSP target architecture.

The SynDEx algorithm graph of the Mpeg-4 coder is currently developed with a finer granularity level. By this way optimal speed factor between mono and multi DSP can be reached allowing high quality real time Mpeg-4 encoding in an embedded context. The use of slices for the complete codec will be studied for multi-processor platforms.

The implementation of elementary and regular operations (DCT for instance) onto a non programmable component such as FPGA would give higher performances. We have work in progress in order to study co-design approach with SynDEx for the automatic implementation of these regular operations onto a non programmable component (Virtex XCV400 - Sundance SMT358).

This work is partially supported by Mitsubishi Electric ITE-TCL. A SynDEx description of an UMTS application is actually developed by Mitsubishi and will take benefits of the results presented here.

New results and the list of published articles in national and international workshops are available at ***"http://www.ietr.org/gro/IMA/th3/temp/index.htm"***.