RAPID PROTOTYPING FOR HETEROGENEOUS MULTICOMPONENT SYSTEMS: AN MPEG-4 STREAM OVER AN UMTS COMMUNICATION LINK

M. RAULET[1,2], F. URBAN[1], JF NEZAN[1], C. MOY[3], O. DEFORGES[1], Y. SOREL[4]

(1) IETR/Image group Lab
UMR CNRS 6164/INSA
20, av des Buttes de Coësmes,
35043 RENNES, France
{jnezan, odeforge,furban}@insa-rennes.fr

(2) Mitsubishi Electric ITE
Telecommunication Lab
1 Allée de Beaulieu,
35 000 RENNES, France
raulet@tcl.ite.mee.com

(3) IETR/Automatic & Communication Lab
UMR CNRS 6164/Supelec-SCEE team
Avenue de la Boulaie - BP 81127
35511 Cesson-Sévigné, France
christophe.moy@rennes.supelec.fr

(4) INRIA Rocquencourt
AOSTE
BP 105
78153 Le Chesnay, France
yves.sorel@inria.fr

ABSTRACT

Future generations of mobile phones, including advanced video and digital communication layers, represent a great challenge in terms of real-time embedded systems. Programmable multicomponent architectures can provide suitable target solutions combining flexibility and computation power. The aim of our work is to develop a fast and automatic prototyping methodology dedicated to signal processing application implementation on parallel heterogeneous architectures, two major features required by future systems. This paper presents the whole methodology based on the SynDEx CAD tool, that directly generates a distributed implementation onto various platforms from a high level application description, taking real-time aspects into account. It illustrates the methodology in the context of real-time distributed executives for multi-layer applications based on an MPEG-4 video codec and a UMTS telecommunication link.

Key words: rapid prototyping - multicomponent - DSP - FPGA - UMTS - MPEG-4

1. INTRODUCTION

New embedded multimedia systems, such as mobile phones, require more and more computation power. They are increasingly complex in design and have a shorter time to market. Computation limits of critical parts of the system (i.e. video processing, telecommunication physical layer) are often overcome thanks to specific circuits [1]. Nevertheless, this solution is not compatible with short time designs or the system's growing need for reprogramming and future capacity improvements. An alternative can be provided by programmable software (DSP: Digital Signal Processor, RISC: Reduced Instruction Set Computer, CISC: Complex Instruction Set Computer) or programmable hardware (FPGA: Field Programmable Gate Arrays) components since they are more flexible. Efficiency loss can be counterbalanced by using multicomponent architectures to satisfy hard real-time constraints. The parallel aspect of multicomponent architectures (programmable software and/or programmable hardware components interconnected by communication media) and possibly its heterogeneity (different component types) raise new problems in terms of application distribution. Real-time executives developed for single processor applications can hardly take advantage of multicomponent architectures: handmade data transfers and synchronizations quickly become very complex and result in lost time and potential deadlocks. A suitable design process solution consists of using a rapid prototyping methodology. The ultimate objective is then to go from a high-level description of the application to its real-time implementation on a target architecture [2] as automatically as possible. The aim is to avoid disruptions in the design process from a validated system at simulation level (monoprocessor) to its implementation on a heterogeneous multicomponent target. Performances of the process can be evaluated by different aspects:

- maximal independence with regards to the architecture,
- possibility of handling heterogeneous multicomponent architectures,
- maximal automation during the process (distribution/scheduling, code generation including data transfers and synchronizations),
- efficiency of the implementation both in terms of execution time and resource requirements,
- reduced design time,
- enhanced quality and robustness of the final executive.

The methodologies generally rely on a description model, which must match the application behavior. These applications are a mixture of transformation and reactive operators [3]. A transformation operator is based on the data-driven process: input data is transformed into output data. A reactive operator is one, which is event-driven and has to continually react to stimuli. In practice, systems are a combi-

nation of both. Nevertheless an important distinction can be made between systems with deterministic scheduling whose operators are mainly transformation-oriented, and systems with highly dynamic behavior whose operators are mostly reactive-oriented. For the first class of system (including signal, image and communication applications), DFG (Data Flow Graphs) have proven to be an efficient representation model. They enable automatic rapid prototyping and lead to optimized scheduling [4].

This paper deals with a rapid prototyping methodology based on the SynDEx tool, which is suitable for transformation-oriented systems and heterogeneous multicomponent architectures. Major contributions concern two points:

- method and tool, more specifically about automatic distributed code generation from SynDEx,
- a complex multi-layers application including video and digital communication layers, going from its high-level description to its distributed and real-time implementations on heterogeneous platforms.

SynDEx automatically generates synchronized distributed executives from both application and target architecture description models. These executives specify the inner component scheduling and global application scheduling, and are expressed in an intermediate generic language. These executives have to be transformed to be compliant with the type of component and communication media so that they automatically become compilable codes. In this article, we will focus on this mechanism based on the concept of *SynDEx kernels*, and detail new developed kernels enabling automatic code generation on various multicomponent platforms.

The design and the distributed implementation of a multilayer application composed of a video (MPEG-4) and a digital communication layer (UMTS) illustrate the methodology. An MPEG-4 coding application provides the UMTS transceiver with a video coded bitstream whereas the associated MPEG-4 decoder is connected to the UMTS receiver in order to display the video. The result is a complete demonstration application with automatic code generation over several kinds of processors and communication media.

The digital communication layer under investigation is a UMTS FDD (Frequency Division Duplex) uplink transceiver [5]. UMTS is the European and Japanese selected standard for 3G. It has already spread to many areas of the world, but is not yet predominant. 3G should enable us to benefit from new wireless services requiring quite a high data rate up to 2Mbps. Typical targeted applications go from wireless internet to video streaming, and also include high-speed picture exchanging and of course voice.

MPEG-4 is the latest multimedia compression standard to be adopted by the Moving Picture Experts Group (MPEG) [6]. The prototyping of MPEG-4 video codecs over multicomponent platforms and their optimizations are studied in the IETR Image Group Laboratory. A part of the project has already been presented in [7]. We will therefore focus on the coupling between the UMTS and MPEG-4 sub-systems rather than describe the video codec in detail.

The paper is organized as follows: section 2 introduces the SynDEx tool and the AAA methodology. Our contribution in terms of prototyping platforms and executive kernels is described in section 3. The UMTS description according to the AAA methodology and its implementations are explained in section 4. The methodology is illustrated and validated by the application (MPEG-4 + UMTS) described in section 5 what allows to reach a new stage in the relevance of the method. Finally conclusions and open issues encountered during the application development are given in section 6.

2. SYNDEX OVERVIEW

SynDEx[1] is a free academic system level CAD (Computer Aided Design) tool developed in INRIA Rocquencourt, France. It supports the AAA methodology (Adequation Algorithm Architecture [8, 9]) for distributed real-time processing.

2.1. Adequation Algorithm Architecture (AAA)

A SynDEx application (Fig.1) comprises an algorithm graph (operations that the application has to execute), which specifies the potential parallelism, and an architecture graph (multicomponent [10] target, i.e. a set of interconnected processors and specific integrated circuits), which specifies the available parallelism. “Adequation” means efficient mapping, and consists of manually or automatically exploring the implementation solutions with optimization heuristics [9]. These heuristics aim to minimize the total execution time of the algorithm running on the multicomponent architecture, taking the execution time of operations and of data transfers between operations into account. These execution times are determined during the characterization process, which associates a list of characteristics, such as execution times, necessary memory, etc with each (operation, processor)/(data transfer, communication medium) pair respectively.

An implementation consists of both performing a distribution (allocating parts of the algorithm on components) and scheduling (giving a total order for the operations distributed onto a component) the algorithm on the architecture. Formal verifications during the adequation avoid deadlocks in the communication scheme thanks to semaphores inserted automatically during the real-time code generation. Moreover, since the *Syn*chronized *D*istributed *EX*ecutives (SynDEx) are automatically generated and safe, part of the

[1] www.syndex.org

tests and low-level hand-coding are eliminated, decreasing the development lifecycle.

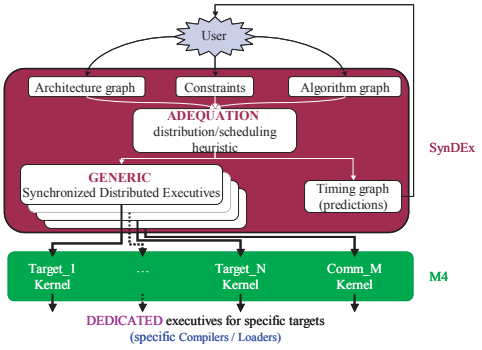

Fig. 1. SynDEx utilization global view

SynDEx provides a timing graph, which includes simulation results of the distributed application and thus enables SynDEx to be used as a virtual prototyping tool.

In the AAA methodology, an algorithm is specified as an infinitely repeated DFG. Each edge represents a data dependence relation between vertices, which are operations; operation stands for a sequence of instructions, which starts when all its input data is available and which produces output data at the end of the sequence. In SynDEx, there is an additional notion of reference. Each reference corresponds to the definition of an algorithm. The same definition may correspond to several references to this definition. An algorithm definition is a repeated DFG similar to those in AAA, except that vertices are references or ports so that hierarchical definitions of an algorithm are possible.

2.2. Automatic Executive Generation

The aim of SynDEx is to directly achieve an optimized implementation from a description of an algorithm and of an architecture. SynDEx automatically generates a *generic executive*, which is independent of the processor target, into several source files (fig.1), one for each processor [11]. These generic executives are static and are composed of a list of macro-calls. The M4 macro processor transforms this list of macro-calls into compilable code for a specific processor target. It replaces macro-calls by their definition given in the corresponding *executive kernel*, which is dependent on a processor target and/or a communication medium. In this way, SynDEx can be seen as an off-line static operating system that is suitable for setting data-driven scheduling, such as signal processing applications [12, 13].

SynDEx kernels are available for several processors, such as the TI[2] TMS320C6x (C62x, C64x) and the Virtex FPGA families, and for several communication media such as links SDBs (Sundance Digital Buses - Sundance High Speed FIFOs), CPs (Comports - SUNDANCE FIFOs), BIFOs (BIFIFOs - Pentek FIFOs), PCI Bus, TCP bus presented in the following section.

2.3. Design Process

Our previous prototyping process integrated AVS[3] (Advanced Visual Systems) as a front-end [14] for functional checking. AVS is a software designed for DFG description and simulation. The application was constructed by inserting existing modules or user modules into the AVS workspace, and by linking their inputs and outputs. The validated DFG was next converted into a new DFG by a translator to be compliant with SynDEx algorithm input. The main advantage was 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 with AVS before the step of the implementation.

Although SynDEx is basically a CAD tool for distribution/scheduling and code generation, here we demonstrate that SynDEx can also be directly used as the front-end of the process for functional checking (as it is possibly done with AVS). This is made possible thanks to our kernels presented in section 3. The design process is now based on a single tool and is therefore simpler and more efficient. SynDEx therefore enables full rapid prototyping from the application description (DFG) to final multiprocessor implementation (Fig.2) in three steps:

Step 1: the user creates the application DFG using SynDEx. Automatic code generation provides a standard C code for a single host computer (PC) implementation (SynDEx PC kernel). In this way, the user can design and check each C function associated with each vertex of its DFG, and can check the functionalities of the complete application with any standard compilation tools. With automatic code generation, visualization primitives or binary error rate computation can be used for easy functional checking of algorithms. The user can easily check his or her own DFG on a cluster of PCs interconnected by TCP Buses. With this cluster, the user can emulate his or her embedded platform thanks to SynDEx distributed scheduling.

Step 2: the developed DFG is then used for automatic prototyping on monoprocessor targets so that to chronometric reports are automatically inserted by the SynDEx code generator. Each duration associated with each function (i.e. vertex) executed on each processor of the architecture graph is automatically estimated using dedicated temporal primitives.

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

[2]Texas Instrument

[3]www.avs.com

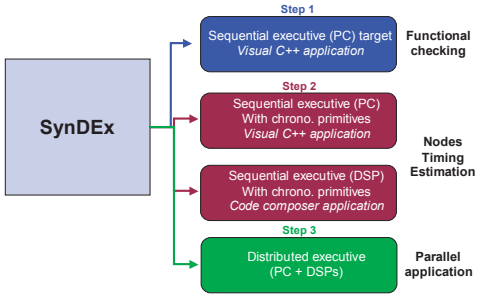

Fig. 2. SynDEx utilization global view

SynDEx. Then SynDEx tool executes an adequation (optimized distribution/scheduling) and generates a real-time distributed and optimized executive according to the target platform. Several platform configurations can be simulated (processor type, their number, and also different media connections).

The main advantage of this prototyping process is its simplicity because most of the tasks performed by the user concern the description of an application and a compiling environment. Only a limited knowledge of SynDEx and compilers is required. All complex tasks (adequation, synchronization, data transfers and chronometric reports) are executed automatically, thus reducing the "time to market". The user can rapidly explore several design alternatives by modifying the architecture graph or adding constraints.

3. SYNDEX EXECUTIVE KERNELS

As described above, the SynDEx generic executive will be translated into a compilable language. The translation of SynDEx macros into the target language is contained in library files (also called kernels). The final executive for a processor is static and composed of one computation sequence and one communication sequence for each medium connected to this processor. Multicomponent platform manufacturers must insert additional digital resources between processors to make communication possible. Thus, SynDEx kernels depend on specific platforms.

3.1. Development Platforms

Different hardware providers (Sundance, Pentek) were chosen to validate automatic executive generation. Many component and inter-component communication links are used in their platforms, ensuring accuracy and the generic aspect of the approach. The use of several hardware architectures guarantees generic kernel developments.

Sundance[4] **platform:** a typical Sundance device is made up of a host PC with one or more motherboards, each supporting one or more TIMs (Texas Instrument Module). A TIM is a basic building block from which you build your system. It contains one processing element, which is not necessarily a DSP but an Input/Output device, or an FPGA. A TIM also provides mechanisms to transfer data from module to module. These mechanisms, such as SDBs (200MB/s), CPs (20MB/s), or a global bus (to access a PCI bus up to 40MB/s), are implemented on the TIMs using FPGAs.

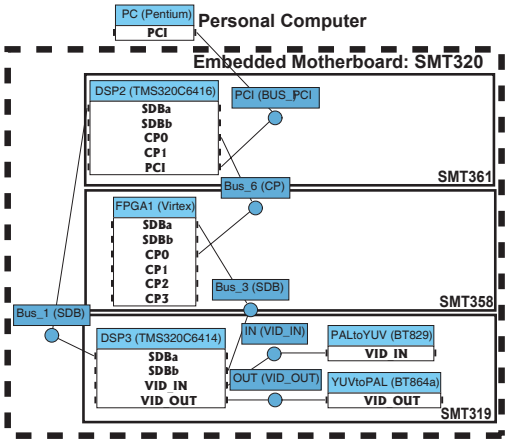

Fig. 3. Example of Sundance architecture topology

The SMT320 motherboard (Fig.3) is modular, flexible and scalable. Up to four different modules can be plugged into the SMT320 and connected using CP or SDB cables. The SMT361 TIM with a TMS320C6416 (400Mhz) is very suitable for imaging processing solutions as the TMS320C64xx has special functions for handling graphics. The SMT319 TIM is a framegrabber, which includes a TMS320C6414 and two non programmable devices: a BT829 PAL to YUV encoder and a BT864 YUV to PAL decoder. These two devices are connected to the TMS320C6414 DSP thanks to two FIFOs, which are equivalent to SDBs with the same data rate. An SMT358 is composed of a programmable Virtex FPGA (XCV600) which integrates specific communication links and specific IP blocks (computation).

Pentek[5] **platform:** The Pentek p4292 platform (Fig.4) is made up of four TMS320C6203 DSPs. Each DSP has three communication links: two bi-directional (300Mhz) inter-DSP links and one for the Input/Output interface. The four DSPs are already connected to each other in a ring structure. Some daughter boards may be added to the p4292 thanks to the VIM (Velocity Interface Mezzanine) bus, such as analog to digital converters (ADC p6216), digital to analog con-

[4] **http://sundance.com/**

[5] **http://www.pentek.com/**

verters (DAC p6229), or FPGAs (XC2V3000, XC2Vx Virtex2 family).

This stand-alone Pentek platform is connected to an Ethernet network. This allows TCP/IP (1.5MB/s) communications between DSPs and any computer in the network in order to check a binary error rate, or to visualize a decoded image. However this Bus's throughputs will not authorize the transfer of uncompressed data.

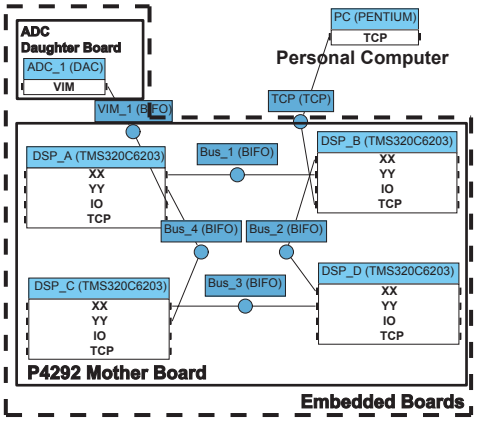

Fig. 4. Pentek 4292 mother board and its daughter board

3.2. Software component kernels

Most of the kernels are developed in C language so that they can be reused for any C software programmable device. These kernels are similar for the host computer (PC) and the embedded processors (DSPs). The generated executive is composed of a sequential list of function calls (one for each DFG operation). This kind of executive and the fact that the adapted C compiler for DSPs has really improved in terms of resource use mean that the gap between an executive written in C and an executive written in assembly language is narrow. The user can design each function associated with each vertex of its DFG in C or assembly language for better results [15].

SynDEx creates a macrocode made of several interleaved schedulers: one for computation and the others for communications allowing parallelism of those actions. We have chosen to use multi-channel enhanced DMA (Direct Memory Access) transfers, thus maximizing parallelism and timing performance. Data transfers are executed in parallel with computation minimizing communication duration. DMA and CPU have their own bus to access the internal memory, therefore bus conflicts only appear when CPU and DMA access an external memory. As all data buffers are in internal memory, memory bus conflicts are null between CPU and DMA accesses. Communication overhead is only due to DMA setup which is negligible to take transfers into account (a few assembly instructions)[16].

The development of an application on TI processors can be hand-coded with TI RTOS (Real-Time Operating System) called DSP/BIOS [17]. DSP/BIOS is well-suited for multithread monoprocessor applications. Several processors must be connected to improve computational performances and reach real-time performances. In this case, the multithread multiprocessor 3L diamond[6] RTOS is more appropriate for this situation than DSP/BIOS. Applications are built as a collection of inter-communicating tasks. The mapping and scheduling of each task are chosen manually. Then data transfers and synchronizations are implemented by the RTOS using precompiled libraries. 3L enables multiprocessor application development easier, faster and suited to dynamic communications between tasks. Data transfers are realized using DMA, but without any computation parallelism which is nearly equivalent to *polling technique*.

Data transfers in a signal processing application are generally statically defined both in terms of data type and number so that their description with a DFG is suitable. The execution of DFG operations is also well-defined so that data transfers can be implemented with static processes. As static processes are faster than dynamic ones, *SynDEx kernels are developed without any RTOS.* That is to say that the SynDEx generic executive is not transformed into dynamic RTOS functions but into specific static optimized functions.

3.3. Communication media kernels

With AAA methodology, two different models are possible for communication media between processors: the SAM (Single Access Memory) and RAM (Random Access Memory, shared memory) models.

The SAM model corresponds to FIFOs in which data are pushed by the producer if it is not full, and then pulled by the receiver if the FIFO is not empty. Synchronizations between the two processors are hardware signals (empty and full flags) and are not handled by SynDEx semaphores. The data must be received in the same order as it is sent. Most of our kernels are designed according to this model. SDBs, CPs and BIFO_DMAs enable parallelism between calculation and communications, whereas TCP and BIFO do not enable it (data polling mechanism).

The RAM model corresponds to an indexed shared memory. A memory space is allocated and an interprocessor synchronization semaphore is created for each item of data that has to be transferred. This mechanism allows the destination processor to read data in a different order to which it has been written by the source processor. Interprocessor synchronizations are handled by SynDEx. The first imple-

[6] http://www.shen.myby.co.uk/threel/

mentation of the RAM model, through the PCI bus, is described in the following section.

A PCI transfer kernel, for communications between a DSP on Sundance platforms and the host computer, is first developed with the SAM model. First the host and DSP must be synchronized. Each data transfer therefore encloses two synchronizations because the PCI bus does not have hardware signals like a usual FIFO (full or empty flag). The receiver must first wait for the sender to write new data in the PCI memory. Then, the receiver can read data from the PCI memory and send an acknowledgement back to the sender. This "rendez-vous as soon as possible mechanism" induces idle or wait states but is mandatory to ensure the medium is ready for the next transfer and to guarantee transfer order. PCI communications using the SAM model reach a maximum transfer rate of 16MB/s. This mechanism drastically slows down PCI transfers. In addition, a shared buffer is actually allocated to the PC's RAM by the PCI bus driver. Therefore, a new PCI kernel implementing the RAM model has been developed and the transfer rate has been improved (up to 40MB/s). Each item of data that has to be transferred has its own address allocation in the PCI memory and corresponding semaphores, which allows several buffers to be written before the first one is read. This results in less wait states and more time for computation. The PCI scheduler is controlled by interrupt when using this model. Consequently, communications and computations can be concurrent on the DSP, thus reducing overall execution time.

3.4. Hardware component kernel

Moreover an FPGA kernel for programmable hardware components has been developed in HDL (Hardware Description Langage) and could be considered as a coprocessor in order to speed up a specific function of the algorithm. This kernel handles automatic integration of inter-component communication syntheses and instantiates a specific IP (Intellectual Properties) block.

Programming of a communication link depends on its type but also on the processor. Previous works have already validated these libraries [18] however they need to evolve with processors or communication links (depending on provider's additional logic).

3.5. Kernel organization

The libraries are classified to make developments easier and to limit modifications when necessary. As shown in Figure 5, these files are organized in a hierarchical way. An application-dependent library contains macros for the application, such as the calls of the algorithm's different functions. A generic library contains macros used regardless of the architecture target (basic macros). The others are architecture-dependent: processor or communication type dependent. Processor-dependent libraries contain macros related to the real-time kernel, such as memory allocations, interrupt handling or the calculation sequence. Communication type-dependent libraries contain macros related to communications: send, receive and synchronization macros, communication sequences. As different processor types (with different programming of the link) can be connected by the same communication type, one part per processor type can be found in one library. The right part of the file is used during the macro-processing.

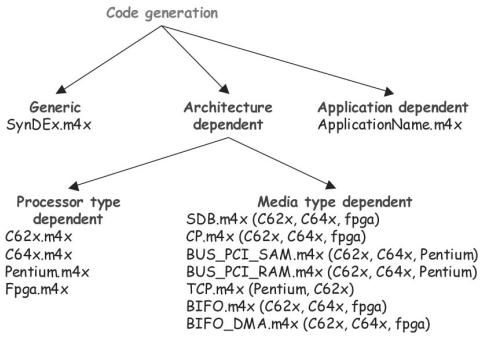

Fig. 5. SynDEx kernel organization

Kernels have been developed for every component of the platforms described in section 3.1. When SynDEx is used for a new application, only the application-dependent library needs to be modified by the user. Architecture-dependent libraries are added or modified when a new architecture is used (a processor or a medium that does not have its kernel).

4. UMTS APPLICATION

UMTS is much more challenging than previous 2G systems, such as GSM. In particular, UMTS signals have a 3.84MHz bandwidth compared with 270kHz for GSM. Both application and signal processing layers are very demanding. This partially explains the delay in the effective arrival of UMTS on the market. It presents a very interesting case study for high efficiency multi-processing heterogeneous implementations. This becomes even more relevant in a Software Radio [19] context, which aims to implement as much radio processing as possible in the digital domain, and especially onto processors and reconfigurable hardware. The advantages firstly consist of easing the system design while privileging fast software instead of heavy low level hardware development. Secondly the system supports new services and features thanks to software adaptation capability during system operation [20].

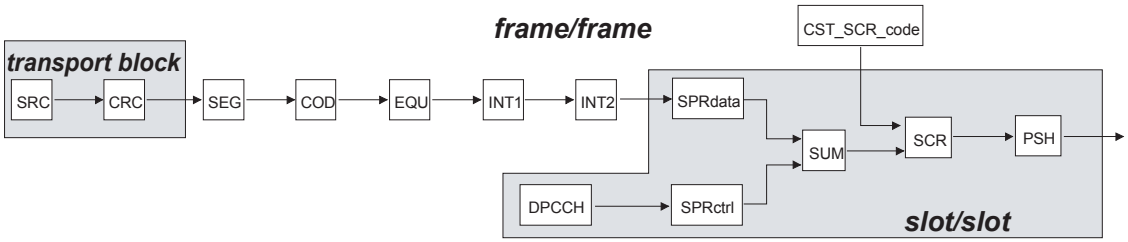

Fig. 6. UMTS FDD transmitter (Tx)

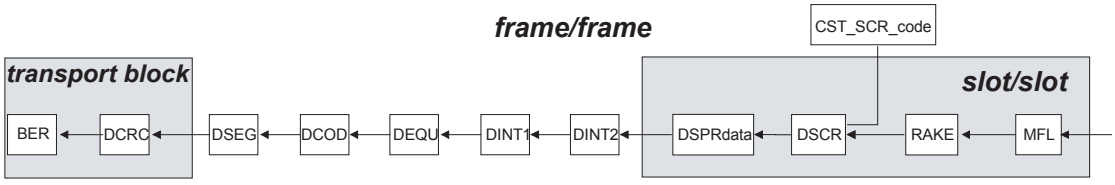

Fig. 7. UMTS FDD receiver (Rx)

4.1. General description

UMTS FDD physical layer algorithms explained in [5] are implemented for baseband from cyclic redundancy check (CRC) to pulse-shaping (PSH) (Table 1) for the transmitter as shown in the DFG in Figure 6. This does not represent a total real UMTS since synchronization is artificial and no propagation channel is used (the link is completely digital). Data may be generated by an arbitrary source (SRC Fig.6: not in the standard) for bit error rate verifications or extracted from a real application, such as a video stream, to make demonstrations.

SRC	Source (Pseudo random generator)
CRC	Add of cyclic redundancy check bits
SEG	Segmentation
COD	Channel coding
EQU	Equalization
INT1	First interleaving
INT2	Second interleaving
SPRdata	Spreading of information bits
SPRctrl	Spreading of control bits
SUM	Creation of a complex signal
CST_SCR_code	Generation of the scrambling code
SCR	Scrambling
DPCCH	Generation of control bits
PSH	Pulse-shaping

Table 1. Legenda of UMTS FDD transmitter

Link characteristics in the measured version are as follows:

- 1 transport channel,
- 1 physical channel,
- no channel coding,
- spreading factor of 4,
- data rate of 950kbps.

The receiver [5] extracts the information necessary for the application using the scheme represented in Figure 7 (Table 2).

MFL	Matched filter
RAKE	Simplified Rake (One perfectly synchronized finger)
CST_SCR_code	Generation of the scrambling code
DSCR	Descrambling
DSPRdata	Despreading of information bits
DINT2	Deinterleaving 2
DINT1	Deinterleaving 1
DEQU	Equalization inverse operation
DCOD	Channel decoding
DSEG	transport block extraction
DCRC	Analysis of cyclic redundancy check bits
BER	Bit error rate

Table 2. Legenda of UMTS FDD receiver

The number of operations effectively in use is much greater than the figures shown as most of them are duplicated several times. The generation of a 10ms frame

(composed of 15 slots) requires the instantiation of approximately 140 operations for Tx and 240 for Rx in this version, which is a minimum. The granularity of the operations has the same level of complexity as a FFT, FIR or a memory reorganization.

4.2. FIR implementation

The filter operation is of particular interest because its implementation complexity makes it very resource consuming. This is a FIR (Finite Impulse Response) with a raised-root cosine impulse response specified by the UMTS standard at both transmitter baseband output and receiver baseband input. Here, the impulse response is symmetric around its center; this characteristic can be exploited to minimize the number of memory accesses, the required memory for storing the filter coefficients and the number of multiplication operations. In order to obtain a convenient rejection of contiguous bands, the filter impulse response is spread over 16 chips and consequently has 33 taps with an oversampling of 2. The same coefficients are used for Tx and Rx.

Equation 1 gives us the representation of a FIR filter with an odd number of coefficient where h is the real coefficient vector of the filter impulse response (filter taps), K is number of coefficients (or taps), $x[n]$ and $y[n]$, the n^{th} input and output complex data samples respectively.

$$y[n] = h\left[\frac{K-1}{2}\right] \cdot x\left[n - \frac{K-1}{2}\right] + \sum_{k=0}^{(K-1)/2-1} h[k] \cdot (x[n-k] + x[n-K+1+k]) \quad (1)$$

A real filter (i.e. filter whose coefficients are real) applied to complex data is very frequent in *baseband* (BB) processing and consists of applying the same filter independently to the real and imaginary parts of the data samples. In our case we are interested in fixed point implementations so care must be taken to avoid overflow while preserving signal quality (in terms of SNR). The filter at Tx is called *pulse shaping* (PSH) and at Rx *matched filtering* (MFL). At Tx PSH and oversample (which consists of inserting zero between binary digits) operation can be combined in order to minimize computation. In this case we obtain:
if n is even

$$y[n] = \sum_{k=0}^{(K-1)/4} h[2k] \cdot (x[n-k] + x[n-(K-1)/2+k])$$

if n is odd

$$y[n] = h\left[\frac{K-1}{2}\right] \cdot x\left[n - \frac{K-1}{2}\right] + \sum_{k=1}^{(K-1)/4-1} h[2k] \cdot (x[n-k] + x[n-(K-1)/2+k])$$

The nature of a FIR operation is particularly suited to FPGA implementations but can also be implemented on DSP processors. A specific characteristic of the DSP is that it has a MAC (Multiply ACcumulate) or a VLIW structure to support filtering computing in one clock cycle. The TMS320C6x family, based on VLIW architecture, has six adders and two multipliers, which operate in parallel and complete execution in one clock cycle. A fixed-point multiply-accumulate takes two instructions: multiply on one cycle and accumulate on the next. Thanks to pipelining, it is possible to effectively compute two multiply-accumulates per cycle.

The performance then directly depends on filter length and processor clock frequency as each tap is processed sequentially. In an FPGA, it is possible to parallelize part or all of these operations, depending on the available gate surface. FIR implemented in the FPGA is a distributed arithmetic (DA) filter[21]. Features of this FIR are not multipliers, but only ROM (Read Only Memory) and accumulators. The complexity of this filter only depends on the number of bits per sample, not on the number of taps.

In the particular case of C6x, it is possible to use a data buffer organization of the FIR as shown in Figure 8. FIR is a typical case where functional units in the microprocessor datapath can speed up processing. Data is processed in blocks. The interface consists of an input data buffer, the coefficient buffer and an output data buffer.

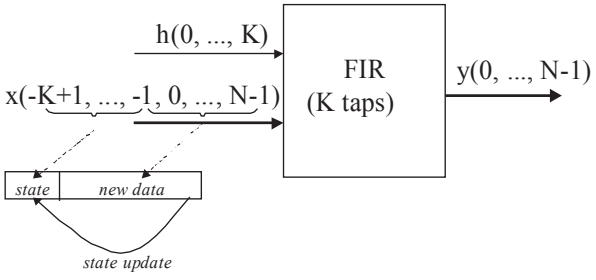

Fig. 8. Data management for DSP implementation of a FIR

The algorithm for each input sample performs the function of $y[n]$ in a for-loop. At the end of each block processing operation, the filter state is updated by copying the last K input data into a state buffer (Fig. 8). For the sake of processing efficiency, it is assumed that the input data buffer is stored in a memory after the state data buffer so that negative indices of the input data buffer point to the state buffer data.

	C62x 300Mhz	*C64x 400Mhz*	*XC2Vx 100Mhz*
Time/slot (μs)	576	320	338

Table 3. Timing of PSH (input: 2560 samples)

	C62x 300Mhz	*C64x 400Mhz*	*XC2Vx 100Mhz*
Time/slot (μs)	1130	640	338

Table 4. Timing of MFL (input: 5120 samples)

In Tables 3 and 4, the differences in timing between C62x and C64x (without taking clock rates into account) is due to the fact that compilers are not the same for each processor and that those DSPs have different internal architectures. In an FPGA (XC2Vx), this FIR operation could be more parallelized giving better acceleration to the detriment of the gate surface. However, these time values are sufficient to get a Tx or Rx real-time application, that is why we use the same FIR implementation for PSH and MFL. An elementary oversampling function just has to be added before PSH. On the contrary to FPGAs, we take advantage of the FIR features (cf. 4.2) on DSPs to optimize and divide by 2 the computation complexity of PSH at Tx, so that 576μs versus 1130μs are obtained on C62x and 320μs versus 640μs on C64x.

4.3. Tx and Rx implementations

Four different implementations (Table 5) of a UMTS transmitter have been automatically tested using SynDEx: three are implemented on Pentek platform and one on Sundance platform. A transmitter application must last under 10ms to be real-time.

Principally due to PSH (Table 8, timing PSH ratio compared to a Tx implementation), the first transmitter implementation onto the Pentek platform did not reach real-time with one C62x DSP however it is possible to parallelize PSH in order to process half of the samples on two processors. Before filtering, two buffers of 1296 samples (as described in Figure 8) must be created. Each block processing operation overlaps 16 transient samples. The length of this PSH is reduced by 1.5 when transfers are taken into account.

Furthermore code generation and kernels can be used to quickly shift to another platform. UMTS prototyping on the Sundance platform required indeed few hours to reach to a real-time transmitter application, thanks to our previous works (UMTS algorithm description, SynDEx code generation and kernels) on Pentek platform. This is a tremendous proof of the portability capabilities offered by the methodology.

UMTS Rx has been implemented according to three different configurations (Table 6). A real-time application has been achieved on the Pentek platform with one DSP and one FPGA. MFL parallelization is also possible on several DSPs on Pentek platform, however more than two DSPs are added

	Sundance	*Pentek*		
Configuration	1*C64x	1*C62x	2*C62x	1*XC2Vx 1*C62x
Time/frame	9. 5ms	11.8ms	8.5ms	9.6ms
PSH ratio	50%	73%	53%	52%

Table 5. Tx timings and PSH ratio

compared with one FPGA in the previous configuration. A configuration with 4 DSPs requires many transfers in the Pentek ring structure, thus not reducing MFL computation length by too much.

	Sundance	*Pentek*	
Configuration	1*C64x	1*C62x	1*XC2Vx 1*C62x
Time/frame	15.9ms	20.2ms	9.9ms
MFL ratio	60%	84%	32%

Table 6. Rx timings and MFL ratio

5. MPEG-4 OVER UMTS: A MULTI-LAYER SYSTEM

MPEG-4 is the latest compression standard. An MPEG-4 codec can be divided into ten main parts (e.g. system, visual and audio) with different timing requirements and execution behaviors. Each part is divided into profiles and levels for the use of the tools defined in the standard. Each profile (at a given level) constitutes a subset of the standard so that MPEG-4 can be seen as a toolbox where system manufacturers and content creators have to select one or more profiles and levels for a given application. The application handled here is an MPEG-4 part 2 codec developed in our laboratory, which is based on the Xvid[7] Codec. This MPEG-4 codec has also been tested on several distributed platform configurations [7] (Multi-DSP implementation). Here, our aim is to interface UMTS with MPEG-4 to provide a bitstream to the UMTS application.

The methodology permits to merge the design of very different (heterogeneous) parts of the system in terms of hardware processing support (PC, DSP, FPGA) as well as processing nature. A conventional methodology would require different environments, which is a cause of bugs and incompatibility at the integration step. This causes delays in the best case, and could even completely question the design in the worst case. Our approach permits to gather the different parts of the design very early in the design flow and anticipate integration issues. Nevertheless MPEG-4 over UMTS

[7] www.xvid.org

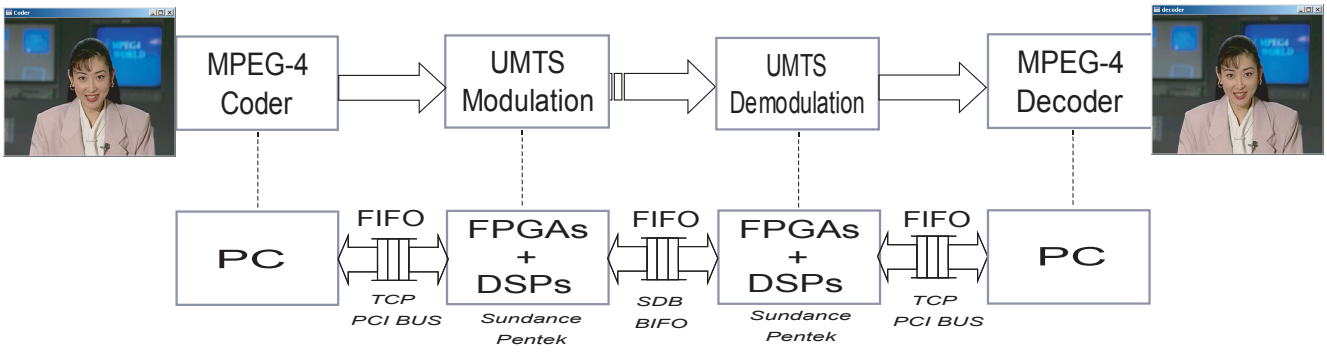

Fig. 9. MPEG-4 over UMTS

arises a new difficulty: the complete application is a multi-layer system (two layers MPEG-4 and UMTS) with different data periodicities between layers. A consequence is that the whole application can not be represented by a single DFG. The solution consists of breaking up the UMTS physical layer and the video codec layer into four algorithm sub-graphs. Then these sub-graphs (coder, decoder, modulation, demodulation) have been implemented onto several processors connected each other with media (FIFO) following the topology of figure 9.

The MPEG-4 codec is not embedded here: firstly, TCP throughputs on the Pentek platform do not enable uncoded or uncompressed data to be transferred, and secondly too few Sundance TIMs are available in our laboratory to embed a complete application with UMTS + MPEG-4. Our real-time MPEG-4 codec provides the maximum data rate supported by our UMTS transceiver (950kbps). An MPEG-4 bitstream, coded on a PC, is sent via a UMTS telecommunication link to another PC to be decoded. Once the communication transceiver has been implemented on a platform, it can be viewed as a communication medium equivalent to a FIFO.

So the platform integrating the MPEG-4 codec could be described as two PCs interconnected by a UMTS communication medium. A FIFO is used to connect asynchronous applications (codec to UMTS communication link). Asynchronous means different periodicities and different data exchange formats. A codec cycle corresponds to one image processing operation producing a variable compressed bitstream in a variable time (about 40ms). A UMTS cycle executes one fixed-size frame in 10ms. FIFO material signals (empty and full flags) ensure the self-regulation of the global system (UMTS+MPEG-4). Two implementations of this global system have been rapidly done onto two platforms thanks to developed kernels as described in Figure 9. The global system runs in real-time on Pentek platform and is not far from real-time on Sundance platform (Rx is in 16ms and must be 10ms). The first implementation of the global application on Pentek platform take quite a long time (two months) to find and solve the multi-layer issue, but this implementation is instantaneously transposed on Sundance platform, which exactly illustrates the efficiency and the pertinence of the approach.

6. CONCLUSIONS AND OPEN ISSUES

The design process proposed in this paper covers every step, from simulation to integration in digital signal application development. Compared with a manual approach, the use of our fast prototyping process ensures easy reuse, reduced time to market, design security, flexibility, virtual prototyping, efficiency and portability.

On the one hand, we have shown how SynDEx is capable of manually or automatically exploring several implementation solutions using optimization heuristics, and on the other hand, how it automatically generates dedicated distributed real-time executives from kernels dependent on the processors and the media. These executives are dedicated to the application because they do not use any support of RTOS, and are generated from the results of the adequation taking operation and data transfer distribution and scheduling into account while providing synchronizations between operations and data transfers and between consecutive repetitions of the DFG. The kernels enable recent multiprocessor platforms to be used and also enable the process to be extended to heterogeneous platforms. It was tested on several different architectures composed of TI TMS320C6201, TMS320C6203, TMS320C6416 DSPs, Xilinx Virtex-E and Virtex-II FPGAs, and PCs.

The calculations and data transfers are executed in parallel. RAM and SAM communication models have been tested for PCI transfers. Higher transfer rates are reached using the RAM model enabling real-time video transfers between a PC and a DSP platform.

Several complex tasks are performed automatically, such as distribution/scheduling, code generation of data transfers

and synchronizations. So the development of a new application is limited to the algorithm description and to the adaptation of kernels for platforms or components. Furthermore, as the C language is used and there is a large number of tested topologies, developed DSP kernels can easily be adapted to any other DSP and communication media.

The current MPEG-4+UMTS application is still in progress to achieve wireless communication. A new version already integrates the channel coding (Turbo code) steps, which only slightly increases the overall complexity.

This approach ensures fast prototyping of digital signal applications over heterogeneous parallel architectures in many technological fields. Other applications have already taken advantage of it. A SynDEx description of a MC-CDMA (probably planned as 4G) application has been developed by the IETR SPR laboratory [22]. LAR codec is a video codec studied in the IETR Image Group Laboratory. A similar scheme (Figure 9) has already been tested on different configurations: LAR over MC-CDMA, MPEG-4 over MC-CDMA and LAR over UMTS.

The complex MPEG-4+UMTS application stresses that a multi-layer system presents some specific characteristics in terms of data flow. In the future, this case study may be capitalized on creating in SynDEx new hierarchical models of architecture graphs in such a way the physical layer (telecommunication link) may appear as a particular medium. Another issue is the memory allocation in SynDEx. At each output of each vertex, SynDEx creates an allocation. At this time, memory allocations are reordered and reused manually to give an optimal solution. Current works deal with an automatic solution, based on graph coloring techniques and life memory allocation.

7. REFERENCES

[1] A. M. Eltawil, E. Grayver, H. Zou, J. F. Frigon, G. Poberezhskiy, and B. Daneshrad, "Dual Antenna UMTS Mobile Station Transceiver ASIC for 2Mb/s Data Rate," *IEEE International Solid-State Circuits Conference*, 2003.

[2] K. Keutzer, S. Malik, R. Newton, J. Rabaey, and A. Sangiovanni-Vincentelli, "System Level Design: Orthogonalization of Concerns and Platform-Based Design," *IEEE Transactions on Computer-Aided Design of Circuits and Systems*, vol. 19, no. 12, December 2000.

[3] T. A. Henzinger, C. M. Kirsch, M. A. Sanvido, and W. Pree, "From control models to real-time code using Giotto," *IEEE Control Systems Magazine*, vol. 23, no. 1, pp. 50–64, 2003.

[4] S. S. Bhattacharyya, P. K. Murthy, and E. A. Lee, "Software synthesis from dataflow graphs," *Kluwer*, 1996.

[5] *3GPP - TS 25.213 v3.3.0: Spreading and Modulation FDD*, release 1999.

[6] F. Pereira and T. Ebrahimi, "The MPEG-4 Book," *Prentice Hall*, July 2002.

[7] N. Ventroux, J.-F. Nezan, M. Raulet, and O. Déforges, "Rapid Prototyping for an Optimized Mpeg-4 Decoder Implementation over a Parallel Heterogeneous Architecture," in *28th IEEE International Conference on Acoustic, Speech and Signal Processing (ICASSP)*, Hong-Kong, April 2003, Conference cancelled - Invited paper, ICME 2003.

[8] Y. Sorel, "Massively Parallel Systems with Real Time Constraints, the Algorithm Architecture Adequation Methodology," in *Conf. on Massively Parallel Computing Systems*, Ischia, Italy, May 1994.

[9] T. Grandpierre, C. Lavarenne, and Y. Sorel, "Optimized Rapid Prototyping for Real-Time Embedded Heterogeneous Multiprocessors," in *7th International workshop on Hardware/Software Co-Design CODES'99*, Rome, Italy, May 1999, pp. 74–78.

[10] Y. Sorel, "Real-Time Embedded Image Processing Applications using the AAA Methodology," in *IEEE International Conf. on Image Processing*, Lausanne, Switzerland, September 1996.

[11] T. Grandpierre and Y. Sorel, "From Algorithm and Architecture Specification to Automatic Generation of Distributed Real-Time Executives: a Seamless Flow of Graphs Transformations," in *First ACM and IEEE International Conference on Formal Methods and Models for Codesign, MEMOCODE'03*, Mont Saint-Michel, France, June 2003.

[12] F. Balarin, L. Lavagno, P. Murthy, and A. Sangiovanni-Vincentelli, "Scheduling for Embedded Real-Time Systems," *IEEE Design and Test of Computers*, vol. 15, no. 1, pp. 71–82, January-March 1998.

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

[14] V. Fresse, O. Déforges, and J.-F. Nezan, "AVSynDEx: A Rapid Prototyping Process Dedicated to the Implementation of Digital Image Processing Applications on multi-DSPs and FPGA Architectures," *EURASIP journal on Applied Signal Processing, special issue on*

Implementation of DSP and Communication Systems, , no. 9, pp. 990–1002, September 2002.

[15] Texas Instruments, "TMS320C6000 Optimizing Compiler User's Guide," *reference spru187l*, March 2004.

[16] Y. Le Méner, M. Raulet, J.-F. Nezan, A. Kountouris, and C. Moy, "SynDEx Executive Kernel Development for DSP TI C6x Applied to Real-Time and Embedded Multiprocessors Architectures," in *XI European Signal Processing Conference (EUSIPCO)*, Toulouse, France, September 2002.

[17] Texas Instruments, "TMS320 DSP/BIOS User's Guide," *reference spru423b*, September 2002.

[18] F. Nouvel, S. Le Nours, and I. Herman, "AAA methodology and SynDEx tool capabilities for designing on heterogeneous architecture," *Conference on Design of Circuits and Integrated Systems*, November 2003.

[19] A. Kountouris, C. Moy, and L. Rambaud, "Reconfigurability: a Key Property in Software Radio Systems," *First Karlsruhe Workshop on Software Radios, Karlsruhe, Germany*, March 2000.

[20] C. Moy, A. Kountouris, and A. Bisiaux, "HW and SW Architectures for Over-The-Air Dynamic Reconfiguration by Software Download," *Software Defined Radio workshop of RAWCON'03*, August 2003.

[21] S. A . White, "Applications of Distributed Arithmetic to Digital Signal Processing: Tutorial Review," *IEEE ASSP Magazine*, July 1989.

[22] S. Le Nours, F. Nouvel, and J.F. Helard, "Example of a Co-Design approach for a MC-CDMA transmission system implementation," *Journees Francophones Adequation Algorithme Architecture (JFAAA)*, December 2002.