1 Project context

```
% 1. CONTEXTE ET POSITIONNEMENT DU PROJET
% (1 page maximum) Présentation générale du problème qu'il est proposé de traiter
% dans le projet et du cadre de travail (recherche fondamentale, industrielle ou
% développement expérimental).
```

An embedded system is an application integrated into one or several chips in order to accelerate it or to embedd it into a small device such as a personal digital assistant (PDA). This topic is investigated since 80s using Applications Specific Integrated Circuits (ASIC), Digital Signal Processing (DSP) and parallel computing on multiprocessor machines or networks. More recently, since end of 90s, other technologies appeared like Very Large Instruction Word (VLIW), Application Specific Instruction Processors (ASIP), System on Chip (SoC), Multi-Processors SoC (MPSoC).
During these last decades embedded system was reserved to major industrial companies targeting high volume market due to the design and fabrication costs. Nowadays Field Programmable Gate Arrays (FPGA), like Virtex5 from Xilinx and Stratix4 from Altera, can implement a SoC with multiple processors and several coprocessors for less than 10K euros the piece. In addition, High Level Synthesis (HLS) becomes more mature and allows to automize design and to decrease drastically its cost in terms of man power. Thus, both FPGA and HLS tends to spread over HPC for small companies targeting low volume markets.

To get an efficient embedded system, designer has to take into account application characteristics when it chooses one of the former technologies. This choice is not easy and in most cases designer has to try different technologies to retain the most adapted one.
The first objective of COACH is to provide an open-source framework to design embedded system on FPGA device. COACH framework allows designer to explore various software/hardware partitions of the target application, to run timing and functional simulations and to generate automatically both the software and the synthesizable description of the hardware. The main topics of the project are :

– Design space exploration : It consists in analysing the application runnig on FPGA, defining the target technology (SoC, MPSoC, ASIP, ...) and hardware/software partitioning of tasks depending on technology choice. This exploration is driven basically by throughput, latency and power consumption criteria.
– Micro-architectural exploration : When hardware components are required, the HLS tools of the framework generate them automatically.

At this stage the framework provides various HLS tools allowing the micro-architectural space design exploration. The exploration criteria are also throughput, latency and power consumption.
- Performance measurement : For each point of design space exploration, metrics of criteria are available such as throughput, latency, power consumption, area, memory allocation and data locality. They are evaluated using virtual prototyping, estimation or analysing methodologies.
- Targeted hardware technology : The COACH description of system is independent of the FPGA family. Every point of the design exploration space can be implemented on any FPGA having the required resources. Basically, COACH handles both Altera and Xilinx FPGA families.

As an extension of embedded system design, COACH deals also with High Performance Computing (HPC). In HPC, the kind of targeted application is an existing one running on PC. COACH helps designer to accelerate it by migrating critical parts into a SoC implemented on a FPGA plugged to the PC bus.

COACH is the result of the will of several laboratory to unify their know how and skills in the following domains : Operating system and hardware communication (TIMA, SITI), SoC and MPSoC (LIP6 and TIMA), ASIP (IRISA) and HLS (LIP6, Lab-STIC and LIP). The project objective is to integrate these various domains into a unique free framework (licence ...) masking as much as possible these domains and its different tools to the user.

1.1 Economical context and interest

```
% 1.1. CONTEXTE ET ENJEUX ECONOMIQUES ET SOCIETAUX
% (2 pages maximum)
% Décrire le contexte économique, social, réglementaire. dans lequel se situe
% le projet en présentant une analyse des enjeux sociaux, économiques, environnementaux,
% industriels. Donner si possible des arguments chiffrés, par exemple, pertinence et
% portée du projet par rapport à la demande économique (analyse du marché, analyse des
% tendances), analyse de la concurrence, indicateurs de réduction de coûts, perspectives
% de marchés (champs d'application, .). Indicateurs des gains environnementaux, cycle
% de vie.
```

Microelectronic allows to integrate complicated functions into products, to increase their commercial attractivity and to improve their competitivity. Multimedia and communication sectors have taken advantage from microelectronics facilities thanks to developpment of design methodologies and tools for real time embedded systems. Many other sectors could benefit from microelectronics if these methologies and tools are adapted to their features.

The Non Recurring Engineering (NRE) costs involded in designing and manufacturing an ASIC is very high. It costs several milliars of euros for IC factory and several millions to fabricate a specific circuit. Consequently, it is generally unfeasible to design and fabricate ASICs in low volumes and ICs are designed to cover a broad applications spectrum at the cost of performance degradation.
Today, FPGAs become important actors in the computational domain that was originally dominated by microprocessors and ASICs. Just like microprocessors FPGA based systems can be reprogrammed on a per-application basis. At the same time, FPGAs offer significant performance benefits over microprocessors implementation for a number of applications. Although these benefits are still generally an order of magnitude less than equivalent ASIC implementations, low costs (500 euros to 10K euros), fast time to market and flexibility of FPGAs make them an attractive choice for low-to-medium volume applications. Since their introduction in the mid eighties, FPGAs evolved from a simple, low-capacity gate array technology to devices (Altera STRATIX III, Xilinx Virtex V) that provide a mix of coarse-grained data path units, memory blocks, microprocessor cores, on chip A/D conversion, and gate counts by millions. This high logic capacity allows to implement complex systems like multi-processors platform with application dedicated coprocessors. Using FPGA limits the NRE costs to design cost. This boosts the developpment of methodologies and tools to automize design and reduce its cost.

Nowadays, there are neither commercial nor free tools covering the whole design process. For instance, with SOPC Builder from Altera, users can select and parameterize IP components from an extensive drop-down list of communication, digital signal processor (DSP), microprocessor and bus interface cores, as well as incorporate their own IP. Designers can then generate a synthesized netlist, simulation test bench and custom software library that reflect the hardware configuration. Nevertheless, SOPC Builder does not provide any facilities to synthesize coprocessors and to simulate the platform at a high design level (system C). In addition, SOPC Builder is proprietary and only works together with Altera's Quartus compilation tool to implement designs on Altera devices (Stratix, Arria, Cyclone). PICO [CITATION] and CATAPULT [CITATION] allow to synthesize coprocessors from a C++ description. Nevertheless, they can only deal with data dominated applications and they do not handle the platform level. The Xilinx System Generator for DSP [http ://www.xilinx.com/tools/sysgen.htm] is a plug-in to Simulink that enables designers to develop high-performance DSP systems for Xilinx FPGAs. Designers can design and simulate a system using MATLAB and Simulink. The tool will then automatically generate synthesizable

Hardware Description Language (HDL) code mapped to Xilinx pre-optimized algorithms. However, this tool targets only DSP based algorithms.
Consequently, designer developping a embedded system needs to master for example SoCLib for design exploration, SOPC Builde at the platform level, PICO for synthesizing the data dominated coprocessors and Quartus for design implementation. This requires an important tools interfacing effort and makes the design process very complex and achievable only by designers skilled in various domains. COACH project integrates all these tools in the same framework masking them to the user. The objective is to allow **pure software** developpers to realize embedded systems.

The combination of the framework dedicated to software developpers and FPGA target, allows small and even very small companies to propose embedded system and accelerating solutions for standard software applications with acceptable prices. This allows to avoid huge hardware investment in opposite to ASIC based solution.
The combination of the framework dedicated to software developpers and FPGA target can open new markets to small and even very small companies. Such markets we can state HPC (High Performance Computing) and embedded applications. HPC consists in proposing accelerating solutions for standard software applications with acceptable prices, for example, DNA sequencing recognization or DBMS acceleration. Embedded application consists in implementing an application on a low power standalone device, for example distributed intelligent sensors.
This new market may explose like it was done by micro-computing in eighties. This success were due to the low cost of first micro-computers (compared to main frame) and the advent of high level programming languages that allow a high number of programmers to launch start-ups in software engineering.

1.2 Project position

```
% 1.2. POSITIONNEMENT DU PROJET
% (2 pages maximum)
% Préciser :
% -positionnement du projet par rapport au contexte développé précédemment :
%   vis- à-vis des projets et recherches concurrents, complémentaires ou antérieurs,
%   des brevets et standards.
% - positionnement du projet par rapport aux axes thématiques de l'appel à projets.
% - positionnement du projet aux niveaux européen et international.
```

The aim of this project is to propose an open-source framework for architecture synthesis targeting mainly field programmable gate array circuits (FPGA).

To evaluate the different architectures, the project uses the prototyping platform of the SoCLIB ANR project (2006-2009).
The project will also borrow from the ROMA ANR project (2007-2009) and the ongoing joint INRIA-STMicro Nano2012 project. In particular we will adapt existing pattern extraction algorithms and datapath merging techniques to the synthesis of customized ASIP processors.

Regarding the expertise in High Level Synthesis (HLS), the project leverages on know-how acquired over 15 years with GAUT project developped in Lab-STIC laboratory and UGH project developped in LIP6 and TIMA laboratories.
Regarding architecture synthesis skills, the project is based on a know-how acquired over 10 years with the COSY European project (1998-2000) and the DISYDENT project developped in LIP6.
Regarding Application Specific Instruction Processor (ASIP) design, the CAIRN group at INRIA Bretagne Atlantique benefits from several years of expertise in the domain of retargetable compiler (Armor/Calife since 1996, and the Gecos compilers since 2002).

The SoCLIB ANR platform were developped by 11 laboratories and 6 companies. It allows to describe hardware architectures with shared memory space and to deploy software applications on them to evaluate their performance. The heart of this platform is a library containing simulation models (in SystemC) of hardware IP cores such as processors, buses, networks, memories, IO controller. The platform provides also embedded operating systems and software/hardware communication components useful to implement applications quickly. However, the synthesisable description of IPs have to be provided by users.
This project enhances SoCLib by providing synthesisable VHDL of standard IPs. In addition, HLS tools such as UGH and GAUT allow to get automatically a synthesisable description of an IP (coprocessor) from a sequential algorithm.

The different points proposed in this project cover priorities defined by the commission experts in the field of Information Technolgies Society (IST) for Embedded systems : ■Concepts, methods and tools for designing systems dealing with systems complexity and allowing to apply efficiently applications and various products on embedded platforms, considering resources constraints (delais, power, memory, etc.), security and quality services■.
Our team aims at covering all the steps of the design flow of architecture synthesis. Our project overcomes the complexity of using various synthesis tools and description languages required today to design architectures.

2 Scientific and Technical Description

2.1 State of the art

```
% 2. DESCRIPTION SCIENTIFIQUE ET TECHNIQUE
% 2.1. ÉTAT DE L'ART
% (3 pages maximum)
% Décrire le contexte et les enjeux scientifiques dans lequel se situe le projet
% en présentant un état de l'art national et international dressant l'état des
% connaissances sur le sujet. Faire apparaître d'éventuels résultats préliminaires.
% Inclure les références bibliographiques nécessaires en annexe 7.1.
```

Our project covers several critical domains in system design in order to achieve high performance computing. Starting from a high level description we aim at generating automatically both hardware and software components of the system.

2.1.1 High Performance Computing

Accelerating high-performance computing (HPC) applications with field-programmable gate arrays (FPGAs) can potentially improve performance. However, using FPGAs presents significant challenges [1]. First, the operating frequency of an FPGA is low compared to a high-end microprocessor. Second, based on Amdahl law, HPC/FPGA application performance is unusually sensitive to the implementation quality [2]. Finally, High-performance computing programmers are a highly sophisticated but scarce resource. Such programmers are expected to readily use new technology but lack the time to learn a completely new skill such as logic design [3].
HPC/FPGA hardware is only now emerging and in early commercial stages, but these techniques have not yet caught up. Thus, much effort is required to develop design tools that translate high level language programs to FPGA configurations.

```
[1] M.B. Gokhale et al., Promises and Pitfalls of Reconfigurable
Supercomputing, Proc. 2006 Conf. Eng. of Reconfigurable
Systems and Algorithms, CSREA Press, 2006, pp. 11-20;
http://nis-www.lanl.gov/~maya/papers/ersa06_gokhale_paper.
pdf.
[2] D. Buell, Programming Reconfigurable Computers: Language
Lessons Learned, keynote address, Reconfigurable Systems
Summer Institute 2006, 12 July 2006; http://gladiator.
ncsa.uiuc.edu/PDFs/rssi06/presentations/00_Duncan_Buell.pdf
[3] T. Van Court et al., Achieving High Performance
with FPGA-Based Computing, Computer, vol. 40, no. 3,
pp. 50-57, Mar. 2007, doi:10.1109/MC.2007.79
```

2.1.2 System Synthesis

Today, several solutions for system design are proposed and commercialized. The most common are those provided by Altera and Xilinx to promote their FPGA devices.
The Xilinx System Generator for DSP [http ://www.xilinx.com/tools/sysgen.htm] is a plug-in to Simulink that enables designers to develop high-performance DSP systems for Xilinx FPGAs. Designers can design and simulate a system using MATLAB and Simulink. The tool will then automatically generate synthesizable Hardware Description Language (HDL) code mapped to Xilinx pre-optimized algorithms. However, this tool targets only DSP based algorithms, Xilinx FPGAs and cannot handle complete SoC. Thus, it is not really a system synthesis tool.
In the opposite, SOPC Builder [CITATION] allows to describe a system, to synthesis it, to programm it into a target FPGA and to upload a software application. Nevertheless, SOPC Builder does not provide any facilities to synthesize coprocessors. Users have to provide the synthesizable description with the feasible bus interface.
In addition, Xilinx System Generator and SOPC are closed world since each one imposes their own IPs which are not interchangeable. We can conclude that the existing commercial or free tools does not coverthe whole system synthesis process in a full automatic way. Moreover, they are bound to a particular device family and to IPs library.

2.1.3 High Level Synthesis

High Level Synthesis translates a sequential algorithmic description and a constraints set (area, power, frequency, ...) to a micro-architecture at Register Transfer Level (RTL). Several academic and commercial tools are today available. Most common tools are SPARK [HLS1], GAUT [HLS2], UGH [HLS3] in the academic world and catapultC [HLS4], PICO [HLS5] and Cynthesizer [HLS6] in commercial world. Despite their maturity, their usage is restrained by :

- They do not respect accurately the frequency constraint when they target an FPGA device. Their error is about 10 percent. This is annoying when the generated component is integrated in a SoC since it will slow down the hole system.
- These tools take into account only one or few constraints simultaneously while realistic designs are multi-constrained. Moreover, low power consumption constraint is mandatory for embedded systems. However, it is not yet well handled by common synthesis tools.

– The parallelism is extracted from initial algorithm. To get more parallelism or to reduce the amout of required memory, the user must re-write it while there is techniques as polyedric transformations to increase the intrinsec parallelism.
– Despite they have the same input language (C/C++), they are sensitive to the style in which the algorithm is written. Consequently, engineering work is required to swap from a tool to another.
– The HLS tools are not integrated into an architecture and system exploration tool. Thus, a designer who needs to accelerate a software part of the system, must adapt it manually to the HLS input dialect and performs engineering work to exploit the synthesis result at the system level.

Regarding these limitations, it is necessary to create a new tool generation reducing the gap between the specification of an heterogenous system and its hardware implementation.

```
[HLS1] SPARK universite de californie San Diego
[HLS2] GAUT UBS/Lab-STIC
[HLS3] UGH
[HLS4] catapultC Mentor
[HLS5] PICO synfora
[HLS6] Cynthesizer Forte design system
```

2.1.4 Application Specific Instruction Processors

ASIP (Application-Specific Instruction-Set Processor) are programmable processors in which both the instruction and the micro architecture have been tailored to a given application domain (eg. video processing), or in some extreme cases to a specific application (eg H264 specific ASIP). This processor specialization usually offers a good compromise between performance (compared to a pure software implementation on a COTS embeded processor) and flexibility (compared to an application specific hardware co-processor).
As a consequence, this type of architecture is a very attractive choice as a System on chip building block. In spite of their obvious advantages, using/designing ASIPs remains a difficult task, since it involves designing both an efficient micro-architecture and implementing an efficient compiler for this specific micro-architecture.
Recently, the use of instruction set extensions has received a lot of interest from the embedded systems design community [NIOS2,FSL,ST70], since it allows to rely on a template micro-architecture in which only a small fraction of the architecture has to be specialized. Even if such an approach offers less flexiblity and forbids very tight coupling between the extensions and the template micro-architecture, it makes the design of the micro-architecture

more tractable and amenable to a fully automated flow.
However, to our knowledge, there is still no available open-source design flow addressing those two design challenges together, either because the target architecture is proprietary, or because the compiler technology is closed/commercial.
In the context of the COACH project, we propose to add to the infrastructure a design flow targeted to automatic instruction set extension for the MIPS-based CPU, which will come as a complement or an alternative to the other proposed approaches (hardware accelerator, multi processors).

2.1.5 Automatic Parallelization

`-- A COMPLETER LIP`

2.1.6 Interfaces

`-- A COMPLETER INSA Etat de l'art`

2.2 Objectives and innovation aspects

```
% 2.2. OBJECTIFS ET CARACTERE AMBITIEUX/NOVATEUR DU PROJET
% (2 pages maximum)
% Décrire les objectifs scientifiques/techniques du projet.
% Présenter l'avancée scientifique attendue. Préciser l'originalité et le caractère
% ambitieux du projet.
% Détailler les verrous scientifiques et techniques à lever par la réalisation du projet.
% Décrire éventuellement le ou les produits finaux développés à l'issue du projet
% montrant le caractère innovant du projet.
% Présenter les résultats escomptés en proposant si possible des critères de réussite
% et d'évaluation adaptés au type de projet, permettant d'évaluer les résultats en
% fin de projet.
% Le cas échéant (programmes exigeant la pluridisciplinarité), démontrer l'articulation
% entre les disciplines scientifiques.
```

The objectives of COACH project are to develop a complete framework to HPC (accelerating solutions for existing software applications) and embedded applications (implementing an application on a low power standalone device). The design steps are presented figure 1.

HPC setup Here the user splits the application into 2 parts : the host application which remains on PC and the SoC application which migrates on SoC. The framework provides a simulation model allowing to evaluate the partitioning.

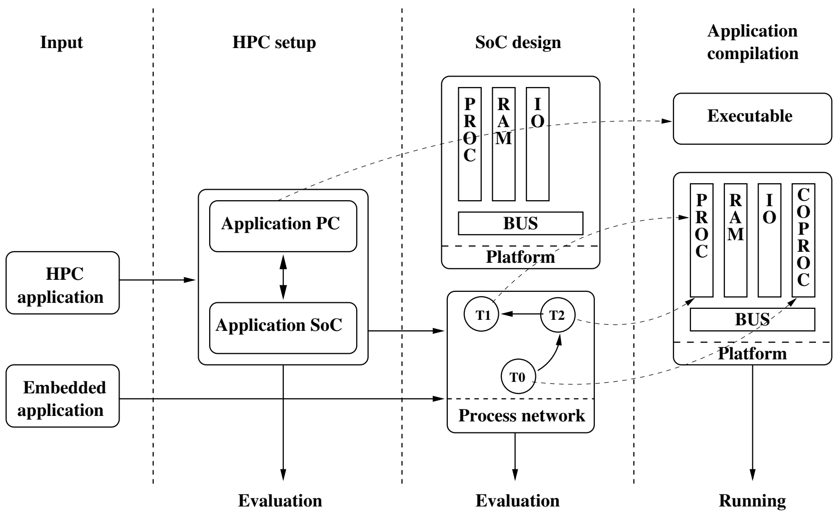

FIG. 1 – COACH flow.

SoC design In this phase, The user can obtain simulators at different abstraction levels of the SoC by giving to COACH framework a SoC description. This description consists of a process network corresponding to the SoC application, an OS, an instance of a generic hardware platform and a mapping of processes on the platform components. The supported mapping are software (the process runs on a SoC processor), XXXpeci (the process runs on a SoC processor enhanced with dedicated instructions), and hardware (the process runs into a coprocessor generated by HLS and plugged on the SoC bus).

Application compilation Once SoC description is validated, COACH generates automatically an FPGA bitstream containing the hardware platform with SoC application software and an executable containing the host application. The user can launch the application by loading the bitstream on FPGA and running the executable on PC.

The main scientific contribution of the project is to unify various synthesis techniques (same input and output formats) allowing the user to swap without engineering effort from one to an other and even to chain them, for example, to run polyedric transformation before synthesis. Another advantage of this framework is to provide different abstraction levels from a single description. Finally, this description is device family independent and its hardware implementation is automatically generated.

System design is a very complicated task and in this project we try to simplify it as much as possible. For this purpose we have to deal with the following scientific and technological barriers.

- The main problem in HPC is the communication between the PC and the SoC. This problem has 2 aspects. The first one is the efficiency. The second is to eliminate enginnering effort to implement it at different abstract levels.
- COACH design flow has a top-down approach. In the such case, the required performance of a coprocessor (run frequency, maximum cycles for a given computation, power consumption, etc) are imposed by the other system components. The challenge is to allow user to control accurately the synthesis process. For instance, the run frequency must not be a result of the RTL synthesis but a strict synthesis constraint.
- HLS tools are sensitive to the style in which the algorithm is written. In addition, they are are not integrated into an architecture and system exploration tool. Consequently, engineering work is required to swap from a tool to another, to integrate the resulting simulation model to an architectural exploration tool and to synthesize the generated RTL description.
- Most HLS tools translate a sequential algorithm into a coprocessor containing a single data-path and finite state machine (FSM). In this way, only the fine grained parallelism is exploited (ILP parallelism). The challenge is to identify the coarse grained parallelism and to generate, from a sequential algorithm, coprocessor containing multiple communicating tasks (data-paths and FSMs).

The main result is the framework. It is composed concretely of : 2 HPC communication shemes with their implementation, 5 HLS tools (control dominated HLS, data dominated HLS, Coarse grained HLS, Memory optimisation HLS and ASIP), 3 systemC based virtual prototyping environment extended with synthesizable RTL IP cores (generic, ALTERA/NIOS/AVALON, XILINX/MICROBLAZE/OPB), one design space exploration tool, one operating system (OS).
The framework fonctionality will be demonstrated with XXX-EXAMPLE1, XXX-EXAMPLE2 and XXX-EXAMPLE3 on 4 archictures (generic/XILINX, generic/ALTERA, proprietary/XILINX, proprietary/ALTERA).