High-Level Design Methodology for Ultra-Fast Software Defined Radio Prototyping on Heterogeneous Platforms

Christophe Moy and Mickaël Raulet

***Abstract*—The design of Software Defined Radio (SDR) equipments (terminals, base stations, etc.) is still very challenging. We propose here a design methodology for ultra-fast prototyping on heterogeneous platforms made of GPPs (General Purpose Processors), DSPs (Digital Signal Processors) and FPGAs (Field Programmable Gate Array). Lying on a component-based approach, the methodology mainly aims at automating as much as possible the design from an algorithmic validation to a multi-processing heterogeneous implementation. The proposed methodology is based on the SynDEx CAD design approach, which was originally dedicated to multi-GPPs networks. We show how this was changed so that it is made appropriate with an embedded context of DSP. The implication of FPGAs is then addressed and integrated in the design approach with very little restrictions. Apart from a manual HW/SW partitioning, all other operations may be kept automatic in a heterogeneous processing context. The targeted granularity of the components, which are to be assembled in the design flow, is roughly the same size as that of a FFT, a filter or a Viterbi decoder for instance. The re-use of third party or pre-developed IPs is a basis for this design approach. Thanks to the proposed design methodology it is possible to port "ultra" fast a radio application over several platforms. In addition, the proposed design methodology is not restricted to SDR equipment design, and can be useful for any real-time embedded heterogeneous design in a prototyping context.**

***Index Terms*—Software-defined radio, design methodology, heterogeneous platform, cross-layer design**

I. INTRODUCTION

THIS paper proposes a method that meets most of the requirements associated with the design of Software-Defined Radio (SDR) equipments. SDR related research aims at investigating all the topics that can help improving future radio systems technologies [1], [2], [3]. In this paper, we address the particular issue of SDR equipments design, which is still a very open subject. This topic is also very similar to any real-time embedded heterogeneous design issue. SDR design is indeed a co-design issue, which is not restricted to SDR, and concerns most of the embedded real-time equipments. If we could discriminate SDR from other equipment categories, we could say that SDR brings the flexibility paradigm to its height. That is why SDR equipments are expected to be made of various flexible processing components, such as DSPs, GPPs, FPGAs and ASICs. Moreover, the flexibility is not only considered at design time in SDR systems, but also at running time. An SDR system consequently should be capable of taking benefit from the potential reconfigurability of the flexible components it is made of. This implies an appropriate design methodology, which transforms this potential reconfigurability into an effective flexibility at run-time [4].

The existence of a reconfiguration management architecture at equipment level, and at infrastructure system level [5] must not be forgotten. However reconfiguration management is not directly included in the design methodology in this paper, but we explain how this approach is compatible with its insertion, in a future step. This assertion relies on our deep experience in reconfiguration management issues [6] and [7]. Reves *et al.* in [8] are also investigating ways to resolve these problems in a similar spirit.

There are several ways to consider SDR design. The approaches can be divided into two distinct categories:

- fast prototyping for lab demos,
- commercial equipments design.

Time has not come yet to consider the second category as a realistic option. Only sub-parts of the design flow are achieved and they are mainly those used in a prototyping flow. We particularly address in this paper the fast prototyping issue from algorithmic simulation to multi-processing heterogeneous implementation. Usual heterogeneous embedded design approaches mix several independent tools dedicated to hardware or software, which require long efforts and can cause errors. In order to tackle the issue, we act at a higher level so that automation and acceleration are implied in the design process.

The paper is organized as follows. The requirements of heterogeneous co-design, as addressed in the SDR area, are explained in section II. But there are many ways to address the concerned co-design issue. This is explained in section III of this paper and section IV extracts the main features of a realistic flow. Section V suggests a possible instantiation of this flow and illustrates how the suggested heterogeneous approach meets section III requirements. Finally, SDR design examples based on the proposed approach are given in VI, before drawing some conclusions.

II. SDR DESIGN ISSUES

SDR design is very challenging and there is not any single solution which covers all radio engineers' expectations. This section intends to list the main issues to be resolved. It

C. Moy is with SUPELEC/IETR, Rennes, France. (phone: +33 2 99 84 45 84; fax: +33 2 99 84 45 99; e-mail: christophe.moy@supelec.fr)

M. Raulet is with INSA/IETR, Image Group, Rennes, France. (e-mail: mickael.raulet@insa-rennes.fr).

can be noted that other embedded systems share the same requirements: this does not restrict our approach to SDR. Advanced image processing, for example, is completely within the scope of the suggested methodology.

The ideal SDR design methodology should offer a set of characteristics facilitating the design of systems, which in turn features the following main characteristics:

- flexibility,
- multi-processing,
- heterogeneous processing,
- object-oriented design facilities,
- HW-SW co-design,
- embedded constraints,
- signal-processing simulation,
- hardware abstraction.

Moreover, portability and re-usability are constant concerns, as well as the elevation of abstraction, in order to simplify the global design process. This list is not exhaustive, but it is sufficiently challenging to be considered a very ambitious target.

A. Flexibility

The salient feature to be considered first regarding SDR is *flexibility*. All the previously listed elements have to be considered in light of this keyword. Radio design has been based for more than a hundred years on analog electronic components, which imposes a pre-defined transmission scheme. SDR is the answer to this limitation thanks to the recent technology progresses.

Flexibility is achieved through a digital approach. Radio applications become digital (ideally, software) and can be played in any hardware platform. The radio application (software here) consequently can be changed, just like a piece of software that can be changed on a computer. This is the end of a century of fixed-behavior radio. This is the beginning of the software radio era.

B. Multi-processing

SDR is defined as an underdeveloped version of the ideal Software Radio of Fig. 1 [1]. Software Radio comprises very demanding digital signal processing capabilities, since it brings both digital-to-analog (DA) conversion at the transmission side, and analog to digital (AD) conversion at the reception side, closer to the antenna. Consequently, multi-processing is an SDR intrinsic requirement, and its purpose is to speed up the required computations.

Software radio is currently not realistic except for low carrier frequency transmission systems. More often, at least a radio frequency (RF) translation to intermediate frequency (IF) or baseband (BB) is done in the analog mode, and this between the antenna and the AD or DA conversion. This pragmatic approach is illustrated in Fig. 2 and named Software-Defined Radio or SDR.

Even if this approach reduces the efforts supported by digital processing components, this situation remains very challenging. The constraints are so strong that the direct

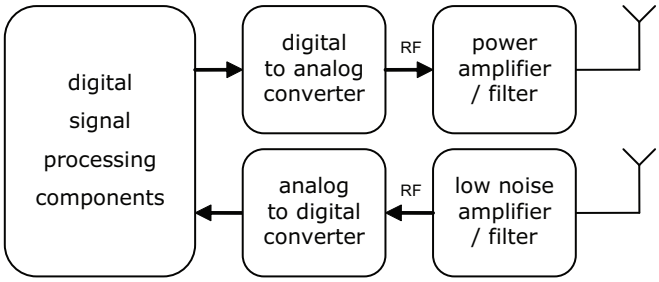

Fig. 1. Ideal software radio equipment architecture.

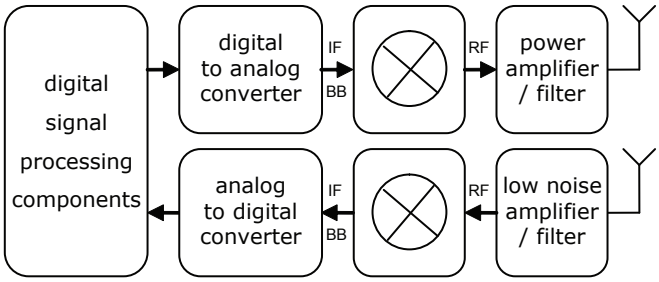

Fig. 2. A realistic SDR equipment architecture.

approach shown in Fig. 2 may fail and imposes sometimes to investigate signal processing alternative solutions in the most demanding contexts, as for UWB for instance in [9] and [10]. Nevertheless, in the most relaxed situation of a baseband digitization, digital operations are timed at a multiple of the signal bandwidth, which can then reach tens of MHz in a single band terminal for UMTS, and hundreds of MHz for multi-band base-stations. A multiple of GHz will soon be required for multi-standard SDR base-stations or UWB transmissions. Most demanding operations are those, which are over-sampled compared to the symbol rate: filtering at the transmitter exit or at the receiver entrance, as well as the very complex algorithms used for reception, such as turbo decoding.

Consequently, digital processing architectures are very complex and combine several similar devices or several kinds of processing devices. This last consideration is the topic of the next paragraph. But multi-processing, without heterogeneity, is a topic in itself. An SDR design methodology must provide some facilities, in order to map and program a distributed system. Designing an SDR equipment means that the software elements, which form the radio application, need to be allocated on the equipment hardware devices.

It has to manage communications as well as processing. It has to schedule communication and processing periods for each processing unit. This includes scheduling prediction and optimization means, in order to perform an automatic mapping and scheduling of the SW application for the HW platform. Moreover, the result of this is the demand for automatic generation of the communication glue associated with the processing operations.

C. Heterogeneous computing

The heterogeneity of devices indeed allows benefiting from the different advantages associated with each category of processing components. Let us keep in mind that flexibility is of major interest for SDR. Consequently, the most popular processing devices in SDR design are the following:

- GPPs,
- DSPs,
- FPGAs,
- digital ASICs,
- analog ASICs.

As explained earlier, analog ASICs are still necessary. SDR cannot get away from antennas, amplifiers and analog filters; this is a condition associated with the transition from the electric to the electro-magnetic world. Digital ASICs and FPGAs are able to process heavily parallel computing. ASICs are hardware solutions particularly suited for low power, very high speed computing. But they are dedicated to high quantity markets in order to compensate for the chips design cost. FPGAs provide some of the requested hardware (HW) capabilities at a lower buying and developing cost. But FPGAs offer HW flexibility, which is very important when considering SDR. Two levels of flexibility may be considered for FPGAs. Firstly, flexibility at design time, enabling to create several designs in a FPGA and run them at different periods of time. FPGAs have been used this way for years, and it is of great benefit. But FPGAs offer a new opportunity now, which consists in dynamically reconfiguring a sub-part of the FPGA gates while the rest of the gates can carry on the process. Delahaye *et al.* have defined and validated a design methodology, as well as flexible processing elements design approaches which are particularly interesting for SDR and involve a partial reconfiguration of FPGAs [11]. Other domains are also making investigations, as for crypto [12], but this is out of the scope of this paper.

DSPs offer the best trade-off between processing power and power consumption [13]. They are particularly appropriate in the SDR context. Manufacturers now integrate more and more co-processing capabilities dedicated to radio signal processing, such as Viterbi decoders in the TI C6416 DSP for instance. Their C language programming ability makes them really relevant for SDR. They enable portability from one DSP device to another one (with the hypothesis of good compilers).

GPPs have become a potential alternative in recent years with a tremendous increase in their computation power. GPPs have the advantage of supporting high-level programming languages, as well as highly advanced operating systems. This later feature may be helpful for portability but also for reconfiguration management purposes.

D. HW-SW co-design

The distribution of the processing elements between the different categories of processing devices needs to be done on the basis of the capabilities of each category of components, processing power, but also power consumption, heat dissipation, cost and other relevant factors must be taken into account. This topic is very complex and no automatic solution exists today. The solution relies mostly on engineers' experience. The main challenge is to separate software (SW) from hardware (HW). Let us clarify this statement: we consider SW as being an object running a processor, and HW the object executed on a FPGA. We will retain this distinction in the rest of the article.

By HW-SW co-design, we have to restrict here to the design flow ability to integrate at a high-level, processing elements of different nature in code, such as C or VHDL, so that those are run in different processing components categories, such as GPPs, DSPs, FPGAs or ASICs. We also take into consideration the specific case of parameterizable ASIC since, if the processing is fixed, the configuration of their parameters may be done on a DSP within a C function. In this sense, an ASIC is considered as a processing unit running a dedicated functionality.

E. Object-oriented design facilities

SDR is an evolution of the radio design. Radio design comes from the electronics field, which is very much linked to the notion of hardware component. This is mainly due to technology reasons. Electronics components have been for a century the only bricks available to build a radio. On the other side, computer engineering integrated the electronics paradigm of "components" in the SW domain. This makes a component-based design approach [14] definitely suitable for SDR design demand [6]. Advantages are numerous, but let us highlight the following at least: modularity, portability, replacement by reconfiguration, re-usability, *etc.* The reason is that the component-based approach leads to a natural separation between signal processing and execution architecture.

The object-oriented approach indeed gives the opportunity to increase the design abstraction level. This has two main consequences. Firstly, SDR design may benefit from the latest progresses in high-level design. The use of UML for embedded systems design in general [15] and SDR design in particular [16] are some example. Secondly, we argue that the temptation of inventing a completely brand new design flow for SDR must be avoided. SDR design must be based on other design technologies in order to benefit from their advances. In that sense, we defend the following approach for SDR design. An SDR design flow has to give the opportunity to integrate processing elements (let us also call them IPs here, for Intellectual Property) made by other specialized designers or tools. Then SDR design may be seen as a kind of IPs integration process. This reduces the SDR design flow constraints supported by the processing elements or IPs design and provides the designer a higher level position, thanks to a component-based approach. This permits to consider signal processing elements as black boxes with very general characteristics, such as for instance execution time, mean power consumption, memory use, *etc*., and not as a succession of atomic level operations, or a set of gates. Another advantage, if not the most important, is that it allows to keep each IP design optimality, thanks to dedicated tools specific to each domain; For instance as a synthesizer for a FPGA, or as a compiler for a DSP. It is unfair to pretend being able to make a better tool than the each of these domains

specialists. Moreover, this gives the opportunity to benefit from IPs third party developers. This announces the proliferation of IPs integrators on generic platforms in the future. The re-use of IP, moreover, is a guarantee of reliability. It also gives the opportunity to benefit from the use of IP, the content of which is protected.

F. Embedded constraints

Embedded constraints can be the following:

- hard real-time,
- memory limitation constraints,
- power consumption,
- cost,
- size.

In the SDR context, the first step to overpass is often the resolution of hard real-time data flow constraints. This often implies the consideration of memory optimization as well. These two points are state-of-the art rapid prototyping approaches, and are of course also to be considered in commercial equipment design. A real-time guarantee is mandatory. Consequently, It may not be preferable to rely on real-time operating systems (RTOS) in that context, and when compared to a static operations scheduling.

Power consumption, as well as cost and size are not considered in a prototyping approach, but are when considering commercial systems. As a consequence, those are not in the scope of the approach proposed in this paper. A long term goal is to integrate these considerations from the very beginning of the design phase, which is currently done manually. Research studies on power consumption are numerous [2], with regards to the topic importance. Some are based on a power prediction [17] or measure [18]. Attempts to help taking into consideration several of these parameters at the very early steps of a design flow have been proposed [19], but without providing an integrated resolution method yet.

G. Signal-processing integrated simulation

Another required feature of an SDR design flow is the capability to simulate the system in order to check its functional and non functional behavior. A major point to be stressed is that there is almost no interest for such a simulation if the conformity between the simulated version and the final version installed in the real system is not guaranteed. If not, the checking will have to be done again in the platform.

An alternative option is to give the possibility to implement the system very quickly on the hardware platform, and directly run the real system in order to check it *in-situ*.

H. HW abstraction

Abstraction layers may have several roles in an SDR system. At design time, it may be seen as a technique that enables to abstract the signal processing IPs from the hardware target, as a Java code planned to run on a Java virtual machine. This is not very good for SDR, since it may decrease the processing performance too much. Another approach is to take benefit from a higher level design approach, in order to hide some details of the hardware implementation. This may be achieved through the use of only a few features for the characterization of IPs execution on this hardware. This permits to speed-up both design phase and direct implementation on the platform, where very precise measurements may be done with exact results.

Another abstraction is acting at run-time. This may be done thanks to a middleware layer. The most common proposal related to this, is the Software Communication Architecture (SCA) selected by the US Department of Defense (DoD) in the JTRS (Joint Tactical Radio System) program [20]. A CORBA (Common Object Request Broker Architecture) software bus is proposed in the SCA, in order to support the abstraction of the hardware for the software. This is a major source of concern for the radio design community. Solutions trying to go round the full-CORBA approach while supporting some SCA compatibility are numerous [21]. They mostly consist, for real-time signal processing, in bypassing CORBA which should definitely be restricted to reconfiguration management. As explained earlier, reconfiguration management is not within the scope of the current paper but has to be taken into account, as attempted in [6], [7], [8].

I. A design approach with a wider scope than the sole SDR domain

The list of properties for SDR design approach has many commonalities with other application fields. This means that solutions targeting SDR design may also be applicable in other domains. In the context of image processing in particular, all listed characteristics are valid, even the flexibility. This is particularly relevant regarding the very latest versions of video coding schemes from MPEG standardization groups. MPEG-4 for instance targets a large scale of coding/decoding schemes which implies flexibility by nature. This requirement is also stressed at its maximum in recent RVC (Reconfigurable Video Coding) where reconfigurability is a goal itself [22]. Another example is the need to solve massively heterogeneous multi-processing issues related to the networking infrastructure, which implement video and audio formats transcoders, in order to broadcast multimedia contents on different media types (from high definition HD TV to handheld DVB-H).

III. Current and innovative SDR design approaches

A. Today: no existing solution

Regarding the design flow requirements listed in the previous section, no integrated solution for the design of SDR systems currently exists and it may be expected that none will ever be found. Consequently, a set of solutions have to be addressed separately, and then combined. We would rather expect that a new tool shall be used, in combination with already existing tools, dedicated themselves to design subparts. We suggest here to present a summary of the potentially expected solutions, on the basis of the already existing state-of-the-art languages and commercial proposals.

B. Co-design languages

If we consider the SDR design issue as a co-design issue, let us consider first the languages that have been created in the last 10 years for that purpose. We may refer to SystemC [23], HandelC from Celoxica, and CatapultC from Mentor Graphics, to discuss the popular ones. The initial goal was to provide designers with a completely integrated flow for the SW joint design, SW being the processors coding, and HW, FPGAs coding. When using the term *integrated*, we refer to the use of a unique language that would be compiled for the SW part of the system and synthesized for the HW part after a joint edition, simulation, debug and validation process. However, the SW and HW worlds are intrinsically different if not antagonist, which makes it very challenging. Another idea was to facilitate the HW design, while getting rid off the VHDL coding and turning it to C-like. The previously mentioned languages proposed a C-like programming method for HW, while creating C++ libraries depicting HW design requirements, such as binary and vector data types, or the possibility to express parallelism. But all C++ code can not be converted into HW, which leads to a lot of synthesis impossibilities. The restrictions are so important that we almost not exaggerate if we say that it finally consists in writing VHDL in C++, which brings no help finally. Moreover, this approach is far from providing the most efficient FPGA design in terms of processing speed, surface occupation and power consumption. This finally becomes accurate with regards co-simulation, but as the HW part has to be manually reprogrammed, this completely breaks the design flow, since the validation at a high level is not guaranteed. We must stress here on the following point: a 95% synthesis of the HW target SystemC code is not important enough, if a huge amount of energy is needed to solve the 5% remaining issues, which is often the case. Finally, these co-design languages are often restricted to transfer level simulations at their best capability.

C. MathWorks

A very popular method for fast prototyping is proposed by Mathworks, based on Simulink. The first solution was proposed in the early 2000 in association with LYRtech, an SDR platform provider [24] and has been generalized to many other providers since. It consists in providing a direct bridge between a Simulink environment for signal processing simulation and the execution on a heterogeneous hardware platform made of DSPs and FPGAs. The link from Simulink to DSPs is obtained through RealTime Workshop from MathWorks. With regards the HW side, Xilinx is providing a set of FPGA IPs, which are guaranteed to be cycle accurate equivalent to the IPs provided in Simulink. It provides indeed an artificial translation from Simulink to the FPGA. However, the following restrictions have to be taken into account: It is firstly dependent on the existence of appropriate APIs (Application Programming Interface) for each platform. If this solution was generalized, all platform providers would have to make the effort to provide those APIs. Secondly and most importantly, the HW domain is restricted to Xilinx provider and technology. Even worse, only a sub-set of Xilinx IPs is valid in this approach. The constraint on the IPs is that an equivalent cycle accurate version of the IPs should exist in both Simulink and VHDL. Note that this does not prevent designers to make their own IP with this technique. The guarantee of the equivalence is then under the responsibility of each company.

Despite the above mentioned concern, MathWorks offers the most effective solution to SDR designer in the short-term. Just a lack of a higher modeling introductory work at specification level in the design flow may be missed. However, it really permits to implement a heterogeneous design in a Simulink environment from functional simulation to proper implementation. It is, in this sense, a signal processing based approach.

D. High-level design

Moreover with longer term objectives, we may refer to computer-science oriented approaches based on high-level modeling, such as UML (Unified Modeling Language) of the OMG (Object Management group). UML is another way to interpret co-design. As UML indeed is dedicated to help modeling any kind of system, even outside the engineering domain, then why not for mixed SW and HW? In addition, systems in general become more and more complex. Hardware and software designers (we refer to this term usual meaning here, not to the “differentiating processors from FPGA designers” meaning) have to cooperate in larger and larger projects. This induces a need for a global system design methodology. The engineering answer has been MDE for Model Driven Engineering, and the methodology, MDA for Model Driven Architecture [25]. This approach is mainly based on UML concepts. Another feature is the two steps PIM and PSM. PIM stands for Platform Independent Model and corresponds to a modeling of the application, which is done independently from any implementation consideration. Then PSM (Platform Specific Model) adds new characteristics to the model, in order to take into account the execution HW platform. MDA, which was introduced in 2000 by the OMG, is now a quite successful concept. An attempt to integrate SDR design in a UML methodology has been done in A3S project [19], [26] for the modeling and prediction of non functional characteristics of such systems. A metamodel for SDR has been established, in order to generate a UML profile named “A3S profile” [19]. A profile is a way to extend UML concepts for a specific domain. In practice, this consists in customizing a UML environment for a specific application domain (and getting rid off all the unnecessary concepts). The OMG has already standardized several metamodels for embedded real-time systems. The latest is MARTE [15]. New researches on SDR have been inspired by MARTE, such as Mopcom [27] whose aim is to define a design flow generating automatically VHDL code from high-level UML-based modeling [28].

Note that if we push SDR design towards cognitive radio, this increases once step further complexity. High-level design, through meta-modelling, is a key to deal with system complexity, such as proposed in [29].

E. Other approaches

Other design approaches for SDR have mainly oriented their solution towards a specific hardware implementation target, while offering the inherent flexibility of SDR design. One solution is to suggest an SDR customized processor. QuickSilver Inc. has designed a hardware chip and associated software development tools. The Adapt2000 ACM System Platform claims to integrate in a single IC, the Adapt2400 ACM (Adaptive Computing Machine), ASICs, DSPs, FPGAs capabilities and micro-processors [30]. The Adapt2400 comprises four distinct heterogeneous node types and a node wrapper. The InSpire Node Software Tool Set complements the Adadpt2400 architecture, and abstracts away the complexity of the heterogeneous, multi-nodal, multi-tasking ACM architecture under a single unified programming model.

Another choice is to privilege brut performance (computation speed, power consumption). A promising approach in this field is the use of systems on chip (SoC) that gathers several (more or less dedicated) processing units inside a single chip. The generalization to a high number of units, as required in SDR systems, leads to networks on chip (NoC). A NoC can be differentiated from a SoC, as the number of units become so high, that communication between units themselves becomes an issue, and they then need their own processing units. A very advanced work in the SDR domain is the FAUST (Flexible Architecture of Unified System for Telecommunication) chip designed by CEA [31]. It has been originally designed to support potential 4G candidates based on MC-CDMA modulation schemes [32]. Flexibility was a necessity in order to support several ranges of parameters and to evaluate several possibilities, and to incorporate flexible schemes for 4G. This included real-time reconfiguration capabilities in a certain set of configurations, which were limited by each unit parameters possibilities. Then it turned in a second step towards a real platform for the support of multi-standards SDR systems. CEA designed a new version of the FAUST chip named MAGALI (Multi-Applications Globally Asynchronous Low-power Integrated circuit), which incorporates new and more diverse processing units. This allows MAGALI to support a large scale of current and future standards, such as WiFi, WiMax, *etc*. with MIMO support. Please note that, even if the set of possibilities is very high, it is limited by specific manufacturing constraints. It is specifically dedicated radio processing and could not address other domains. It also includes advanced capabilities in terms of power consumption savings, which is a definitely advanced feature in the SDR field.

Even if this seems to be in contradiction with choosing a HW target, Vanu Inc. made the choice to remain generic a hundred percent, i.e. the hardware target they selected is "the" generic processor. It is based on the MIT researches, which were carried out in the middle of the 90's [33] for GSM base stations [34]. The main idea is to benefit from the GPPs technological improvement, according to Moore's Law, and keep the advantage of existing high-level programming tools, which are supported by a GPP environment. The portability is then implicit, as it is an exact application of the PC concept to the radio domain. Vanu Inc. was the first company in the USA to be certified for GSM SDR base stations, and began manufacturing in 2004. But we may wonder why this technique is not so popular and why Vanu Inc. does not have a stronger market position. This indicates that this technique is probably not yet mature for the infrastructure. It will consequently be even worse for terminals as GPPs main drawback is power consumption, a key point of terminal design.

Let us mention also the tools which are very accurate for the optimized design of FPGA IPs and may be complementary to the previous ones. For example: LISAtek [35], and GAUT [36]. LISAtek aims at providing integrated tools for the design of ASIPs (Application Specific Instruction Set Processor), in order to obtain an optimized hardware processor architecture for a dedicated set of processing operations and the associated programming tools (debugger, compiler). This could be used for the design of an SDR SoC for instance, as experimented in [37]. GAUT enables to reuse an IP algorithmic specification written in C/C++, and to synthesize it into an equivalent VHDL RTL specification. The potentially pipelined architecture which is generated, allows exploring the design space for FPGAs or ASICs, through a trade-off between processing speed, power consumption and area. This can be a complementary tool for the integration of the processing units into the previous solutions.

From the system point of view, another tool worth to be considered for the design of embedded equipments is CoFluent Studio, made by CoFluent Design [38]. The tool allows to manually exploring a heterogeneous design space with an extreme intimacy, in order to anticipate and orient design possibilities. This is a product-oriented approach which requires quite some time and expertise. Moreover, CoFluent does not provide automatic code generation facilities. The exploration stays at the level of a virtual platform. We mostly privilege in this paper rapid prototyping oriented solutions, which help a signal processing engineer to implement quickly an SDR system with automated steps.

Future will tell if one of those approaches more or less dedicated to SDR, is an appropriate answer or not. Maybe the SDR market will be shared between several of them. The design methodology suggested in the next section is less dedicated to SDR than most of those described here, and could also be worth for other embedded real-time prototyping design perspectives.

IV. A REALISTIC DESIGN FLOW FOR SDR OPEN ARCHITECTURES

The term "realistic" is used here to explain that, with current state-of-the-art technologies, a perfect solution for fully integrated co-design is not available, and even not foreseen. The term "open architectures" means that we consider the solutions based on COTS programmable and reconfigurable components. We exclude from the study any approaches based on pre-oriented hardware, which reduces its range to only a subset of SDR capabilities in terms of flexibility. The solutions suggested here can be used for any heterogeneous real-time data-flow oriented embedded design.

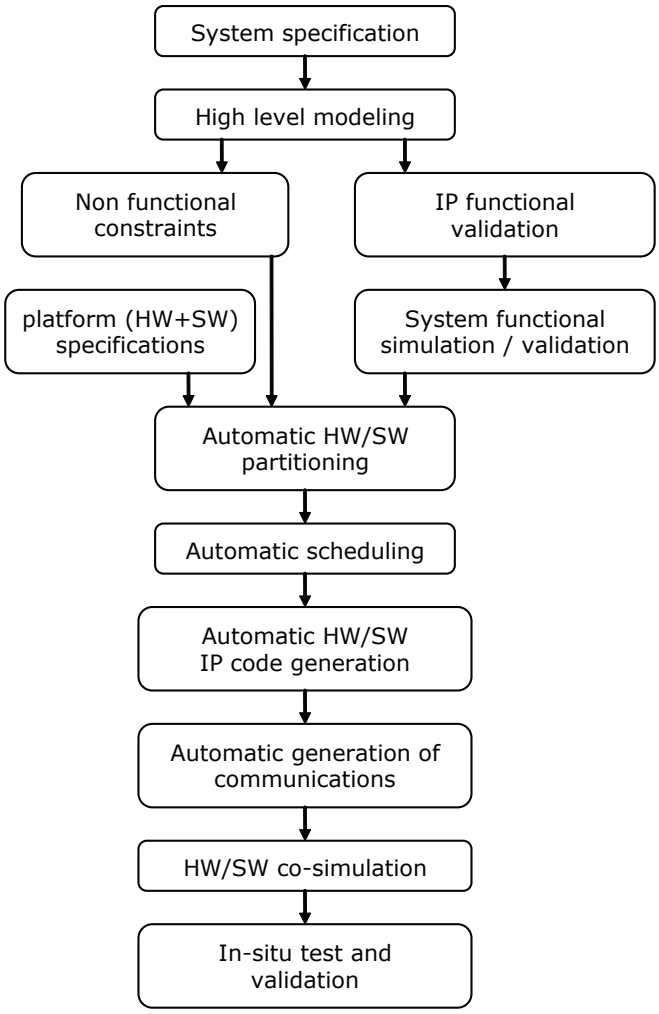

Fig. 3. Ideal SDR design flow.

However, the concern of this paper is SDR, so we must keep in mind that SDR is not only data-flow oriented. The approach has to be compatible with the adding of a reconfiguration management architecture at a later stage of the design, more control-oriented by definition. Moreover, SDR design must not be restricted to radio design. It should be seen from a larger point of view or scale, including other layers of the OSI (Open Systems Interconnection) model. SDR design is also a question of cross-layer design. This typically concerns, at terminal level, a joint approach for both radio and image processing for instance [39].

A. SDR Ideal design flow

Fig. 3 schematically represents what could be the ideal SDR design flow. It would, first of all, consist in a high level modelling phase. This would allow taking advantage of mature modelling techniques, which allow in turn the various design protagonists, respectively HW, SW and signal processing designers, to refine the very early design choices in a common environment issued from the system specifications.

The first step consists in making a simulation and functional validation of each IP. The hardware target that is planned to run the IPs, would be here completely transparent. Ideally, the same language would be used to program a SW (dedicated to run in a processor) or a HW (dedicated to run on a FPGA) IP.

The system functional validation is directly obtained by combining the previous step validated IPs. In conjunction with this step, non functional requirements, as well as platform features are derived. The platform has to be considered here as the combination of devices with their associated low level software as a board support package, or as RTOS, *etc.*.

Then all the information is merged to allow an automatic HW/SW partitioning for heterogeneous multi-processing. Predictions figures are achievable, such as the application execution time, so that other HW/SW matching could be tried if the constraints are not respected. This allows dimensioning the HW platform at the early stage of the design flow. Let us just note that this is far from current reality. The partitioning guarantees the functional accuracy of the application multi-processing version, when compared to the previously simulated mono-processing version. Of course, in that ideal world, the automatic scheduling is also done. The communication code generation is derived directly from the scheduling. IPs automatic code generation is obtained, for either HW or SW. The transformation guarantees an equivalence with the previous model: this permits to avoid the repetition of the simulation and verification procedures, already done earlier.

A co-simulation tool allows to validating definitively the heterogeneous system. Finally, the implementation on the platform is straightforward as the ideal design flow takes every aspects of the implementation, whatever their level.

B. A realistic SDR design flow

A realistic SDR design flow is suggested in Fig. 4. It intends using a set of already existing commercial and/or academic technologies at their maximum capabilities.

Two main goals are achieved here. The first goal is to avoid the reprogramming of the same function several times. This is what happens usually in heterogeneous design. No less than 6 re-coding steps may be done for certain sub-parts of a heterogeneous system, as for instance a HW IP:

- Matlab functional validation,
- floating point C functional validation,
- fixed-point C validation,
- SystemC,
- cycle-accurate SystemC,
- VHDL.

In addition to the extra time necessary to code again the same functionality, this approach is very error-prone and each step requires a repetition of the debug and validation process.

The second goal consists in automating all the implementation dependent actions, and keeping the multi-processing functional accuracy from separately developed and validated IPs. The IPs typical abstraction level is C language or VHDL. This could even be an executable code or a bitstream coming from a third party, which would keep its code secret. This means that the IPs code for the selected HW target is available, or can be obtained with dedicated tools (compiler for DSP and synthesizer, place and route for FPGA), so that non functional

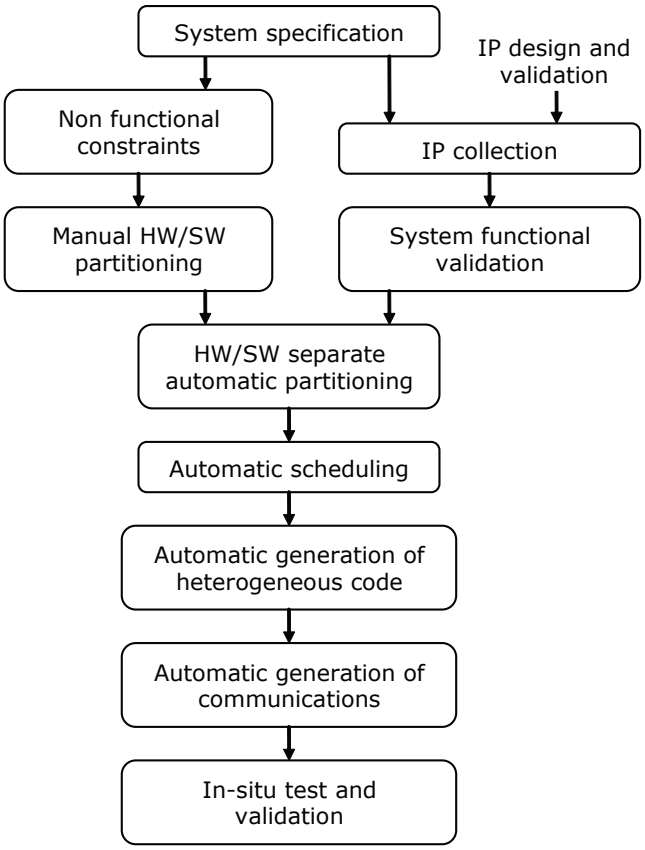

Fig. 4. Realistic SDR design flow today.

characteristics may be extracted, such as for instance: the programming code and data code size in a DSP, the number of gates for a FPGA, the execution speed, *etc.*

As previously stated, the IPs design is voluntary set out of this flow. Please note that we also definitely consider here, that there is no efficient solution for an automatic HW/SW partitioning. This must be left to the designer's discretion, his/her choice being made on the basis of his/her experience.

Then HW (respectively FPGAs) and SW (respectively processors) worlds have to be treated separately, each of them having their own ways of managing multi-processing for automatic partitioning and scheduling. But strong equivalent principles must be applied for both; this enables to keep a global cohesion for an SDR design approach. The most important thing is the asynchronism between operations. This is an obvious thing when it concerns processors, since processing speed is correlated to the device clock and not data rhythm. It can be used in FPGAs using a GALS (Globally Asynchronous, Locally Synchronous) design methodology for HW IPs [40]. This gives many advantages which are also SDR requirements:

- re-usability: an IP may be used in different designs at different clock rates,
- portability: from one HW device to another,
- managing reconfiguration in a future step,
- power consumption considerations, since GALS originate from those.

Nevertheless automatic code generation for communications between both worlds has to be possible at least.

The methodology should also bring some automatic partitioning and scheduling concerning the SW side only. It should be able to provide a multi-processing version, which is a functionally certified equivalent of the application used for the mono-processing version validation.

Section V proposes a way to implement the design flow proposed in Fig. 4.

C. How can reconfiguration be supported at a later stage?

Reconfigurability is intrinsically provided within this methodology by two complementary means:

- the choice of generic hardware components for the platform, such as GPPs, DSPs, FPGAs,
- through the software application building, which is done *via* a component-based approach for both SW (processors) and HW (FPGAs) processing elements.

This is the necessary base that will enable to integrate a reconfiguration management architecture at a later stage.

We have derived a reconfiguration management architecture that is particularly suited to this design methodology [6], [7]. It is out of the scope of this paper to describe it, but we precise that it has been successfully developed and experimented through several prototyping showcases. One main feature of an appropriate reconfiguration management architecture, is to perform reconfiguration operations in a very short time, in order to limit as much as possible the overhead, when compared to the signal processing duration.

It is important to point out that we deliberately disjoined the reconfiguration management design on one hand, and the SDR signal processing on the other hand, while keeping all the necessary interconnections between them.

We do not think that this technology is mature enough yet, so that both approaches can be merged. That is the reason why the design methodology of this paper does not include, but supports the coherence with a reconfiguration management architecture. However we are attempting to pursue this final goal, such as for the FPGA sub-part in the Mopcom collaborative project [27], [28].

Adding to this, it has been highlighted that a particular effort has been done to completely master each part of the design. We insist that this is a key point of an efficient reconfiguration process, i.e. integrate reconfiguration in the processing elements design itself. From a formal point of view, we have been investigating parameterization techniques for several years. This consists in designing processing elements in such a way, that they may be reconfigurable quickly enough through the use of parameters [41]. If we extend the scope of this approach, we can plan to use it for multi-standard equipments design in combination with the use of *common operators*, as explained in [42], [43], [44].

From the experimentation point of view, we have also shown that the reconfiguration management may be merged, at least partially, with the signal processing in order to offer more flexibility. That is why we propose hierarchical and distributed management architecture in [45]. It is particularly obvious

in the particular case of FPGAs partial reconfiguration, as investigated in [7], [11].

The transition from one configuration to another can be extracted from the difference existing between two designs, which are generated rapidly by the methodology proposed in this paper. Some overhead has to be planned for the transition from one design to another in terms of processing time.

D. Which already existing solutions fit or do not?

The combination of Simulink with platforms such as LYRTech seems to us the best solution. But there is an eliminating limit: designing one's own IP is a necessity in order to control the system in such a sensible implementation domain. With the Simulink approach, the designer depends on Xilinx IPs to fully use the design flow pertinence. It is also possible to add your own IP, but it then requires a second coding: both for Simulink and VHDL. Then you miss one of the goals, which is to avoid re-coding.

High level design approaches based on UML offer promising perspectives to formalize the top of the design flow. A bridge is needed between specification and functional validation. A3S approach and A3S metamodel [16] partly answered this need, but without generating, as a result of the architectural study, the code for both validation and implementation. The Mopcom project is trying to fill this gap for the HW side, while using automatic transformations between three layers of metamodels [27]. Each of these layers takes into account, with an increasing level of accuracy, the implementation details. The aim is to model a system at several levels of abstractions, while keeping advantage at a lower level of the validation and results already obtained at a higher level.

We do not pretend that we have explored all the existing solutions, but we mentioned the most interesting ones. In a nutshell, we can draw the following analysis: there are two ways of considering the extension of the design flow towards a fully integrated and automatic design flow for SDR. Either starting from the electronics point of view and follow a bottom-up way. This was chosen by designers who want to keep real-time a priority, for instance. Or it is also possible to look at it in a top-down perspective and privilege a software engineering approach, in order to take full benefit from the latest computer science advances and tools.

V. A Design Methodology for Ultra-Fast SDR Prototyping Based on SynDEx

We suggest here in detail a design methodology for SDR prototyping, respecting 4 and fulfilling the requirements of part II. The set of tools we are using for that purpose is probably not the only possibility. That is why we firstly presented this methodology from a general perspective in section IV-B, and from a possible instantiation perspective in the present section. Designers are then free to use any other implementation of this methodology, according to their particular application domain or habits. We remind also that the granularity level of the constitutive elements considered in the methodology is the size of a signal processing IP, such as a filter, a Viterbi decoder or a FFT for instance.

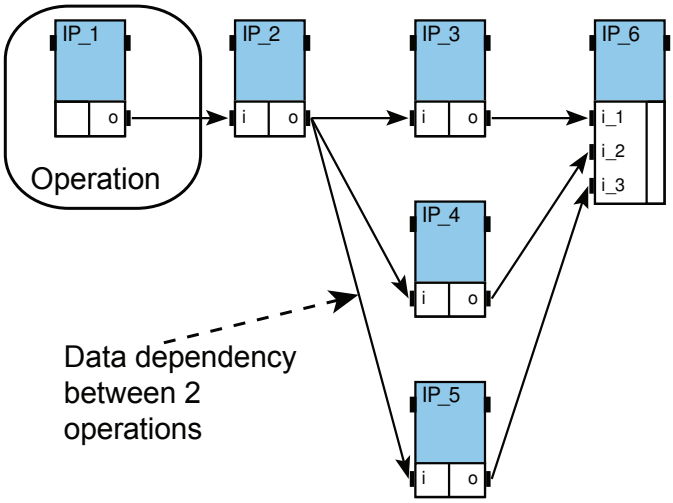

Fig. 5. Example of a SynDEx software application graph.

A. SynDEx approach

The tool selected for the design flow has been SynDEx [46], created by INRIA. SynDEx stands for Synchronized Distributed Executives. The role of this tool in the proposed methodology is to proceed the coarse grain design steps until the code generation [47]. This mainly concerns the communication code glue between IPs.

Then specialized tools dedicated to each processing domain (respectively HW/FPGAs and SW/processors) take over to the implementation. A major requirement of the design flow is that no re-writing of the code is necessary between these two steps in order to keep the former step verifications valid.

a) Major features: Entries for the design flow are two graphs:

- a hardware platform diagram,
- a software application diagram,

In order to speed-up the design phase, it is deliberately chosen to have only a restricted set of parameters and to take into account the architectural exploration:

- execution time of each IP,
- atomic communication time for each data type,
- communication media features

The goal is to obtain an implementation on the HW platform as quickly as possible, in order to deal with realistic constraints instead of having approximated simulations of those constraints in the design flow itself.

Partitioning optimization of SynDEx is performed using graph theory. The SW application graph is modeled by an extended Data Flow Graph (DFG), which is an oriented hypergraph. Each vertex corresponds to an algorithm operation or a processing element, which is activated by the fullness of its input buffer and each edge represents a data transfer between operations. An example of a SynDEx software application graph is given in Fig. 5.

Moreover, the model includes hierarchy, delay, conditioning (if / then / else) and factorization (for loop), in order to express the potential parallelism. Factorization, which is associated

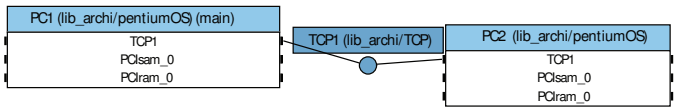

Fig. 6. Example of SynDEx hardware platform graph.

with a repetition factor, is used to repeat operations and requires additional specific vertexes in the DFG:

1) Fork/Join vertexes: the "Fork" vertex explodes each element of the data it receives for each parallel repetition of the consumer operation, whereas the "Joint" vertex builds the data it sends, via the concatenation of each separated element produced by each producer operation repetition.

2) Iterate vertex: this vertex aims at sequentially duplicating a producer/consumer operation, where the outputs of the current operation can be the input of the next one, if the data names are similar. More details can be found in [48].

The hardware architecture is modeled by a non-oriented hyper-graph, where each vertex is a processor (hardware component) and each hyper-edge represents a communication medium, as shown in Fig. 6. In this model, a processor consists of one operator and as many communicators as there are connected media. An operator executes the operations, which are a part of the algorithm, and a communicator executes a communication operation, when a data transfer is required. By doing this, a multi-component architecture can be represented by a network of Finite State Machines (FSM) interconnected with communication media (FIFO, shared memories etc.).

The aim of this tool is to find the best combination of an algorithm which specifies the running of the application, and a multi-component architecture. In addition to this, real-time and embedding constraints must be satisfied. This methodology is based on a graph theory, in order to model the software application and the hardware architecture. The software and hardware are described by two distinct graphs. SynDEx transforms those two graphs with graph transformations in order to generate an optimized code implementation.

An efficient implementation graph is obtained thanks to optimizations and simultaneous distribution and scheduling operations, while trying to minimize the execution time.

There are a large number of possible implementations. The optimization problem aims at selecting the most efficient one between all of them (best latency). The latency is the total DFG execution time on a given HW architecture, between the first scheduled operation and the last one. Moreover, the distribution and scheduling problem in the case of HW multi-component is known to be NP-hard (an exhaustive research on all the possible fulfillments is inconceivable). This is why heuristics are used to find the the optimal solution best approximation (greedy and genetic algorithm). The current heuristic approach attempts to minimize the total running algorithm execution time on the multi-component architecture. Moreover, since the Synchronized Distributed Executives (SynDEx) are automatically generated and safe, this consequently eliminates some of the tests and low-level hand-coding, and decrease the lifecycle development as well.

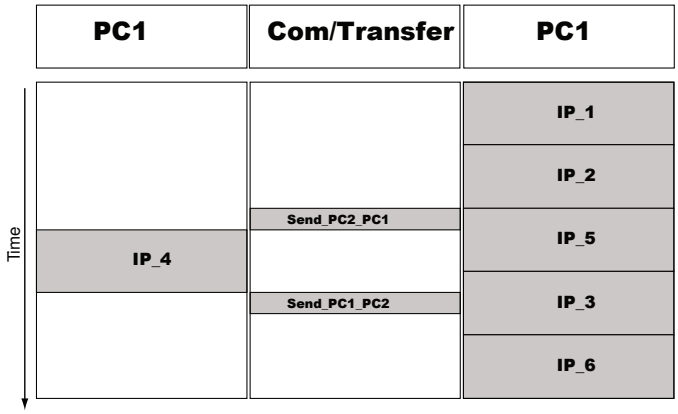

Fig. 7. Example of SynDEx partitioning and scheduling.

SynDEx provides a timing graph, as in Fig. 7, which includes simulation results of the distributed application and thus enables SynDEx to be used as a virtual prototyping tool.

SynDEx is also providing a static scheduling, which is of major importance for hard real-time requirements. It gives the guarantee of an execution within a restricted given latency period. SDR equipment will have very strong real-time constraints to respect. The introduction of SW in radio design must not cause customer's suspicion, or even worse, them rejecting it. This has already been witnessed in the mobile phone area with the collapse of WAP (Wireless Application Protocol), illustrating the fact that a badly introduced technological advance can turn into an economic disappointment.

b) SynDEx tool: SynDEx is a CAD tool whose original aim is to parallelize the application processing for a multi-processor network [49]. It provides a multi-processing version of an application that is functionally accurate, when compared to the mono-processor initial version. SynDEx performs an optimized partitioning and scheduling of an application used with a GPPs multi-processors architecture. Typically, the communication media is a TCP/IP in this context. This can be easily implemented in a platform, instead of a network perspective, by varying communication means between the processors, such as FIFO, dual-port memory, PCI bus, *etc.*. Moreover, SynDEx has been extended towards the embedded reality and in particular in terms of memory optimization [48], [50]. In this sense, SynDEx answers particularly well SDR issues in the restricted SW domain, in other words if the SDR equipment is only made of processors. SynDEx indeed is close to the ideal software radio design. Nevertheless, other SynDEx intrinsic features are worth to be used, beyond the purely SW application domain. After the multi-processor matching, SynDEx generates a scheduled version of the application for each of the processors involved in the platform. This code is generic and not dedicated to any implementation. It is written in M4 macro-code. Therefore a M4 macro-code file is obtained for each processing node of the graph describing both scheduling and communication synchronisms. Those M4 files can be translated in any programming language (assembly, C, VHDL, *etc.*). The translation is operated with a GNU M4 macro-translator, thanks to the use of translation

TABLE I
LIBRARIES DEVELOPPED FOR THE SUPPORT OF MULTI-TARGETS, MULTI-PLATFORMS DEVELOPPED BY IETR/INSA

Platforms	Processors	Communication media	Media Type
Sundance SMT310q	TMS320C64x	SDB (Sundance)	FIFO/SAM
Sundance SMT320	TMS320C62x	SHB (Sundance)	FIFO/SAM
Sundance SMT361	DM642	Comport (Sundance)	FIFO/SAM
Sundance SMT395	x86 (windows)	PCI_RAM (Sundance)	FIFO/RAM
Sundance SMT365	FPGA	PCI_SAM (Sundance)	FIFO/SAM
Sundance SMT 319 (Framegrabber)		TCP (windows, C62x, C64x)	FIFO/SAM
Pentek p4292		Converter ADC (Pentek)	FIFO/SAM
Vitec VP3-PMC		Converter DAC (Pentek)	FIFO/SAM
Sundance SMT348		VP3_RAM (Vitec)	FIFO/RAM
		Bi-FIFO (Pentek)	FIFO/SAM

TABLE II
ALREADY EXISTING LIBRARIES

Processors	Communication media	Media Type
Intel x86 (linux)	TCP (linux)	FIFO/SAM
ADSP21060	RS232	FIFO/SAM
TMS320C40	CAN	FIFO/SAM
MC68332		
MPC555		

libraries, considered as dictionaries. We have been developing translation libraries for many different embedded targets, as shown in Table I, added to the already existing ones shown in II.

Libraries for platforms, processors and communication media are listed in both tables. Some libraries depend only upon the language (C, VHDL), others depend on the processor nature, and finally on other devices, typically those used for communications (FIFO, memory, bus, *etc.*).

Since SynDEx is dedicated to the processor world (called SW in this paper), it extracts parallel operations from the application graph, in order to map them on different processors (hardware components). However, it performs IP sequential operations on each of the HW graph processors. This consequently does not match with the HW (FGPA) reality, which benefits from parallel execution. However all the code generation and scheduling carried out in the SW world on the one hand, and between the HW and SW world on the other hand can be kept in totality. The HW domain content has only to be considered separately, bearing in mind that common principles are shared with the SW side.

B. Heterogeneous processing support

This methodology is originally dedicated to multi-processor (of any kind: DSP, GPP, μC, *etc.*) architectures programming. It also efficiently supports devices which are controlled by processors, such as analog or digital ASICs. And, in order to adapt it to SDR design, it is mandatory to extend it to reconfigurable hardware computing, namely FPGA.

HW implication has indeed to be considered outside the SynDEx partitioning optimization, if we want to keep HW efficiency (of parallelism), since the processor world is sequential. That is why we mentioned earlier that HW/SW partitioning was not supported by this methodology. The risk related to the HW and SW approaches in a single automatic methodology, is to decrease one side performance for the benefit of the other side. Optimization purposes may be indeed at opposite ends, with regards HW and SW. This can be considered as a granularity issue, between respectively the description of an efficient algorithm written in SW (coarse grain), C language, or HW (fine grain) VHDL.

Therefore, we suggest a methodology which integrates the HW design with the SW and system design methodology presented till now, while the HW/SW partitioning has been manually done. The characteristics that should be kept in the HW side and that have to be fully compatible with the design at the SW side, are the following:

- based on a component-based approach,
- HW portability from one design to another, whatever the device, the clock, *etc.*
- HW to SW IPs migration,
- the support of IPs made of gates as well as IPs running on embedded processor cores inside FPGAs,
- the support of ASIC.

c) FPGA support: A reformulation of the design approach has to be made here. The way to solve the interaction issue between an intrinsically sequential and a highly parallel approach lies in the response to the following question: How should we reformulate the data-flow approach related to the processors sequential world within the reconfigurable hardware parallel world, without loosing parallel HW highly efficient processing speed? Let us just recall that one major goal in SDR is to add, delete, replace a processing element, without causing any disturbance to the other processing elements of the radio design being inside or outside the FPGA.

The first part of the solution is solved thanks to the IP approach, which guarantees the parallelism inside the IPs, since it has been designed with usual HW tools. In other words, SynDEx allows keeping the execution parallelism inside an HW IP with this approach, whereas SynDEx is designed so that it executes sequential operations inside a single processor, and only extracts parallelism between several IPs on a multiprocessing architecture.

The second part of the solution uses GALS, as already addressed in section IV-B. That permits to provide also parallelism through pipeline. GALS permits to extend the IPs processing asynchronism to the HW world, independently from the data flow rhythm (just has to be faster). Input buffers, associated with adequate hand shaking signals of Fig. 8 permit to launch and stop the IPs operation, depending on the presence (or not) of data at the input of the IP, or at the place available in the buffers at its output.

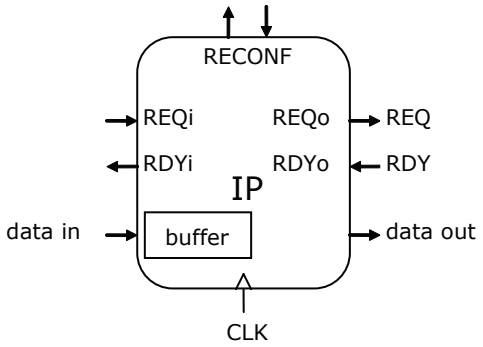

Fig. 8. GALS schematic encapsulation.

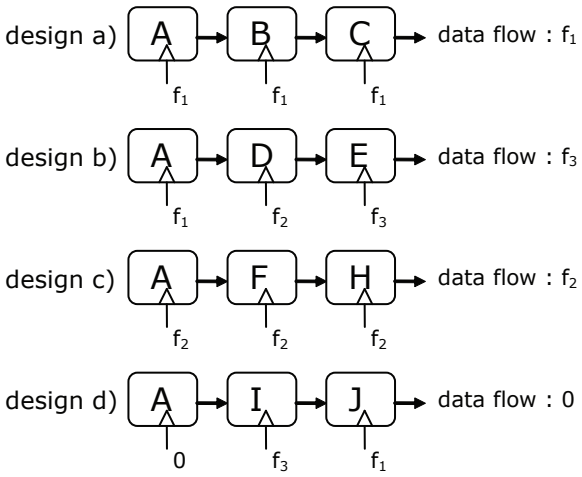

Fig. 9. Data flow speed adaptation in a GALS if $f_1 > f_2 > f_3$.

GALS has three main interesting features in our methodology, as illustrated in Fig. 9:

- It adapts the IP operation average frequency to the other IPs frequencies of the chain (9b for IP A),
- This allows to change an IP of clock domain (9c),
- This allows blocking an IP from time to time, as for a reconfiguration operation for instance (9d).

d) Specific design use cases with FPGA: In the specific case of a unique IP running on a FPGA, the methodology is one hundred percent useful and automatic, as there is no sequential issue for one IP. The FPGA is then a HW accelerator. In this particular context, an automatic HW/SW partitioning is even possible as illustrated in Fig. 11. Let us consider the example of the software application in Fig. 10. If one has to decide whether IP4 should be inside a FPGA or a DSP, then the condition is to have both code versions, and their corresponding execution time for both targets. Then, the methodology is able to deal with such a context and may optimize and generate the full code for both DSP and FPGA with the scheduling obtained on Fig. 11.

The methodology takes the decision to instantiate IP_4 on the FPGA instead of the DSP, since executing IP_4 in parallel of IP_3 and IP_5 executions, saves time: IP_4 execution time and communication overheads are completely masked during IP_5 and IP_3 executions. The FPGA acts here as a co-processor.

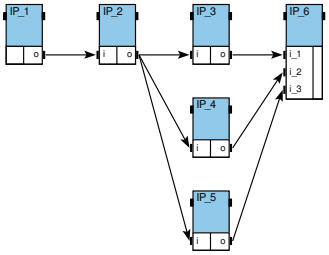

Fig. 10. Software application graph example.

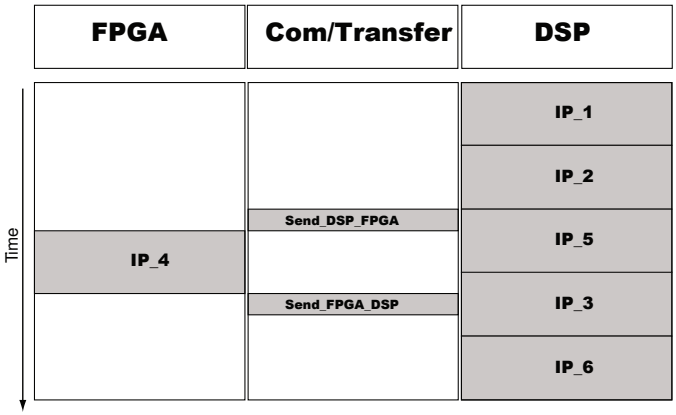

Fig. 11. Fully automatic scheduling obtained with one IP on the FPGA.

A sub-optimal but entirely automatic way of using the suggested methodology is also possible with several IPs on a FPGA. As shown in Fig. 12, it consists in separating a FPGA in as many HW components as there are IPs. If IP3 and IP4 of Fig. 10 are planned to be implemented on a single FPGA, we may define two *Virtual FPGAs*, assigning IP3 on virtual FPGA1 and IP4 on virtual FPGA2. Fig. 12 shows how the methodology is able to parallelize IP3 and IP4 and take it into account in the global scheduling of the application. The merging of all generated VHDL code for each HW components, only needs to be done hereafter.

e) Digital and analog ASIC support: SDR design may not only involve fully programmable or reconfigurable devices, it is necessary to support also specific devices such as ASIC for instance. For example, in an SDR design, typical ASIC devices are: digital down and up converters (DDC and DUC), analog to digital and digital to analog converters (ADC and DAC), amplifiers, analog filters, antenna, *etc.* such as illustrated in Fig. 13. These circuit boards have to be inserted in the design flow for two reasons: firstly, they may be the destination or the source of the data to be processed. Secondly, they may be parameterizable and need to be configured through primitive functions, which are activated by a processor for instance. Therefore, the processor needs to have the necessary code necessary, so that it can perform both initialization and adjustment of these parameters. Please note that in some contexts, a DSP may change registers inside a FPGA, from which they configure the ASIC.

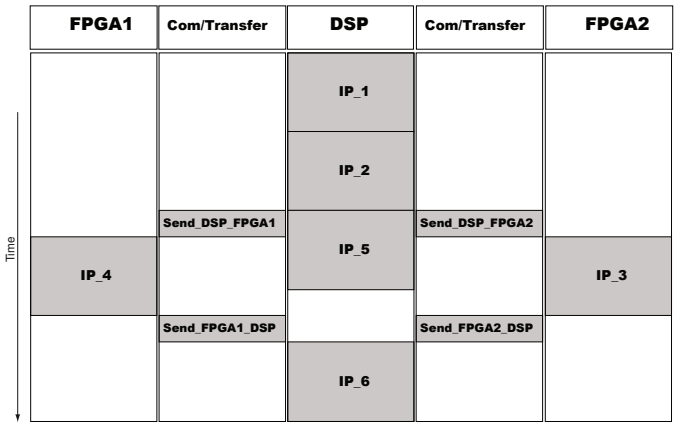

Fig. 12. Scheduling graph with one DSP and one FPGA for each IP.

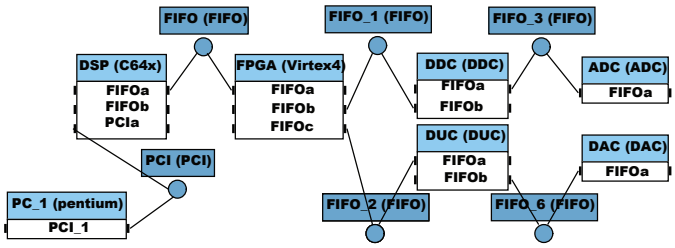

Fig. 13. SynDEx hardware graph with ASIC (DDC, DUC, ADC, DAC).

C. A non functional time-based approach

The set of non functional characteristics to be taken into account would ideally be as many as possible, but in order to keep it feasible, it is restricted here to *time*. This means that each IP has only to be characterized in time in order to be integrated in the design flow. Time includes both processing and communication time. Either it is extracted from measurement, or it can be estimated. Then SynDEx processes the global application timing optimization by mapping the IPs on the hardware architecture. The precision of the global design prediction depends on the accuracy on each IP. The partitioning algorithm used in SynDEx is a greedy algorithm, which provides a good compromise between partitioning execution time and quality result. Another approach has also been developed on the basis of a genetic algorithm, in order to refine the SynDEx greedy algorithm solution [47]. This approach provides the same functional results. If the genetic algorithm is used, the cost function decreases the latency in some cases by 50%. The price is an extended computation time of many thousands compared to a solution obtained by a SynDEx greedy solution.

Moreover, on top of time considerations, some memory considerations may be taken into account and considerably diminish the default memory allocation done by SynDEx [50].

D. Platform abstraction

This methodology implicitly provides, on one hand an hardware platform abstraction related to the software processing elements, by using already existing IPs. With regards to communications, on the other hand, SynDEx generates a generic code which allows the designer to translate it in whatever language he/she wants, as explained in V-A.

The automatic partitioning is maximum in the case of multi-processing on the SW side (processors) using SynDEx, and on the HW side (FPGAs) using dedicated hardware tools such as GAUT for instance. Only the border between HW and SW has to be manually decided: this is the current limitation of the HS/SW co-design issue.

This abstraction permits ease in terms of portability for new processing units or new platforms. It relies on the use of libraries using the appropriate drivers for each platform

An IP can be moved from SW to HW if its code is available for both targets, in C and VHDL for example. This allows design adaptation by mean of a simple click:

- migrate a processing element from a processor to a FPGA and vice-versa,
- add a processing element in a data-flow graph whatever is on the processor: a FPGA or an ASIC
- change of platform,
- migrate a processing element from a gate-oriented implementation to a processor-oriented implementation on a processor core inside a FPGA,
- etc.

E. Methodology principles summary

The proposed methodology is based on the SynDEx tool in association with several concepts, such as the component-based approach. This paragraph aims at recalling its advantages and show how it meets the requirements described in part II as for SDR design, and as for the realistic design flow, described in part IV-B.

SynDEx provides an overall application execution time prediction in the context of a multi-processing platform, featuring several possibilities towards heterogeneous design. At least some FPGA design contexts are fully integrated in the methodology, and ASICs are also included in the design. SynDEx has the advantage to propose an optimized and static scheduling, which is a guarantee for hard real-time SDR constraints. Moreover, the methodology allows the automatic code generation for each processing unit of the platform, including IPs encapsulation and communication glue. Since this code is generic (M4 macro-code), it may be translated in whatever language, thus enabling the support of a heterogeneity of processing devices. The re-usability of IPs is intrinsic to the methodology. All this permits to speed up designs at an amazing rate.

The reduced set of non-functional parameters is considered as a positive feature, with regards ultra-fast prototyping. SynDEx is a tool which can generate prototypes based on existing previous work in a couple of days. Students can use it for small projects. Only a few weeks are needed to become an expert. It is affordable to student trainees for a period of several months as soon as a supporting designer is in the lab. Section VI proves the approach efficiency for many design scenarios examples. If a frequent user of this methodology has a hardware platform at his disposal, then he/she gets used to perform the simulation, test and validation of the application on the platform itself.

Please note that, in the future, any other tools providing such extended features, concerning the presented goals compared to SynDEx, would be worth to be considered. Moreover, this methodology could be also expanded at the top of the design flow, towards higher level design tools based on UML. The input graphs from SynDEx could easily be generated from previous graphs defined in a UML environment. A bridge between A3S graphs and SynDEx graphs for instance would be straightforward. SynDEx would then only take the timing non-functional parameter from A3S. The other non functional parameters (power consumption, memory and surface considerations) taken into account in A3S would not be used in the mapping process itself but would at least have been considered at the very beginning of the design, which is better than nothing. Of course, the long term goal is to really integrate all non-functional parameters in the complete design flow.

VI. Design Examples (Implementation)

Whatever the methodology is, the necessary SDR development efforts are, on one hand, the coding of an application in order to simulate and validate the correct radio functionality, and on the other hand, the coding of a low level software associated with the hardware components in order to prototype the radio on the platform. The lack of methodology results in the repetition of this process from scratch for each new design. This is the current SDR domain situation.

The methodology suggested in this paper permits to share development efforts for both:

- radio signal processing IPs,
- low level software of hardware architecture.

This is the main cause of the acceleration of the methodology; together with a set of concepts particularly adequate for SDR design and explained earlier. We suggest in the following paragraphs to illustrate the methodology efficiency in various design *scenarii*. Please note that it will be illustrated that SDR design is tackled in its widest acceptation here. This involves also cross-layer design, as PHY (radio) layer, as well as application (video) layer may be merged with the proposed design approach. This proves also the relevance of the approach for general heterogeneous embedded design outside SDR.

A. Starting from scratch

Our first attempt to develop an SDR is as simple as a broadcast FM receiver development. This is a play activity for a hundred percent software radio implementation, since the antenna is directly connected to the analogue to digital converter. The selected platform is a Pentek P4292 made of four TMS320C6203 DSPs. All this processing power is much greater than necessary, and is only used to evaluate the methodology in a multi-processing context. First of all, the work consists in developing the FM receiver data-flow graph with a component based approach, and the fixed-point C language code for the content of each IP. This represents a time period ranging from a few days to a couple of weeks of work at the most, depending on the previous knowledge of the modulation, the receiver quality (stereo or mono, choice

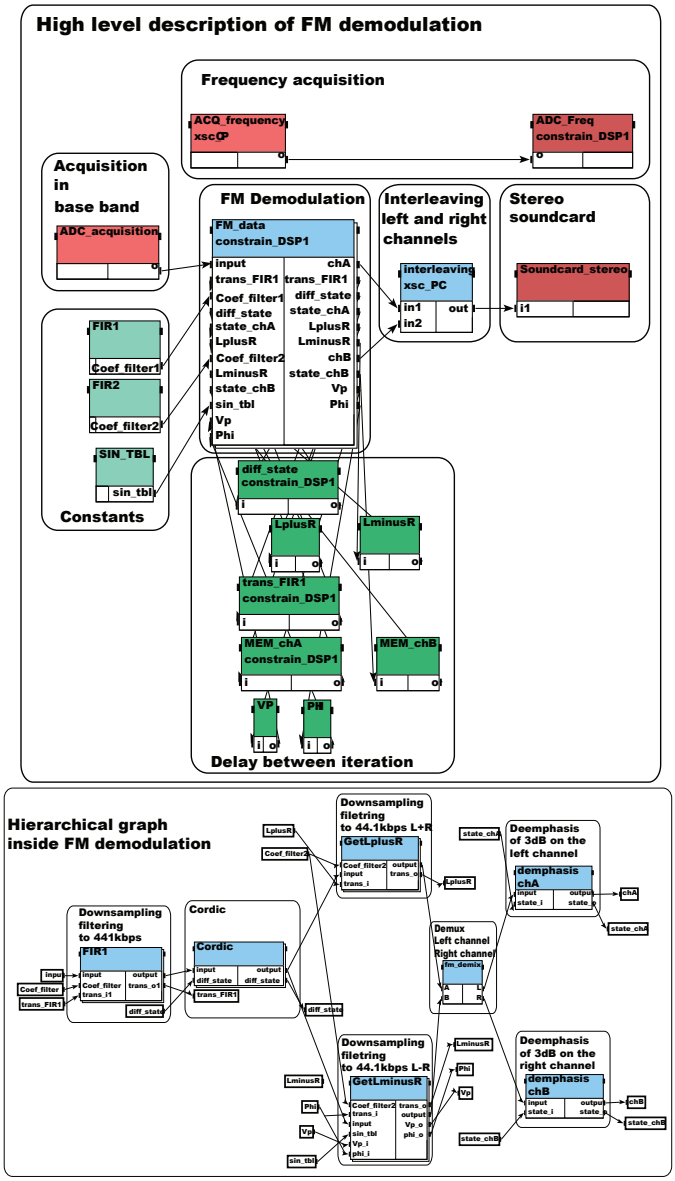

Fig. 14. SynDEx FM receiver application graph.

of demodulation algorithm, *etc.*). This can be done on a first hardware platform architecture made of one PC, which is available by default in SynDEx.

Secondly, low level software for the HW component support are generated for the Pentek platform and gathered in libraries. This concern all the specific primitives, which address synchronisations and threads for the C6203, as well as the communications dedicated to the platform FIFO . This may take several weeks, but less than 2 months and includes M4 learning.

It is not at this stage that the methodology allows to gain anything, since this preliminary work is always necessary. However, as the design steps are very well identified and separated in the methodology, this is already helpful and enables to save time. The work can also be easily separated between two designers, one signal processing designer for the software application (FM) and one for the platform software support

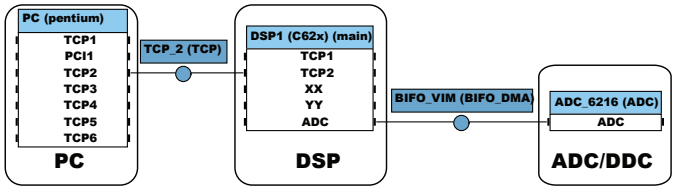

Fig. 15. SynDEx Pentek hardware platform graph.

(libraries). Fig. 14 gives a FM receiver application graph which enables to generate, with the use of this methodology, the full software radio FM receiver. Bottom Fig. 14 represents the hierarchical view of the top *FM Demodulation* box.

The FM receiver shown here is a complex one which operates stereo demodulation. It is the result of the improvement of much simpler previous designs. Only one DSP is sufficient to run the FM receiver in real-time. Nevertheless the methodology enables to run it on several DSPs with no additional effort.

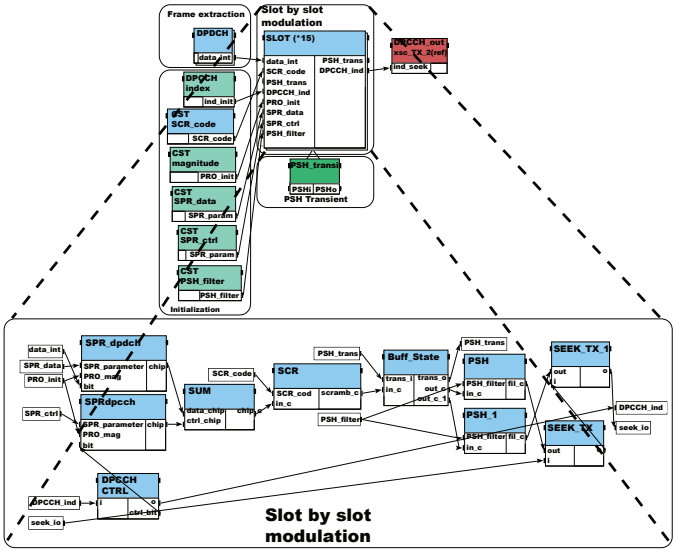

Fig. 16. SynDEx UMTS transmitter application graph.

B. Digital ASIC implication

Under-sampling techniques are used with a 65 MHz ADC sampling frequency. The tuning is managed by a software, which changes the DDC (Digital Down Converter). These considerations may appear or not in the HW platform graph. The digital ASICs present in such architectures, and in the Pentek platform for example, are usually managed by DSPs. DSPs contain the adequate initialization code, necessary to configure ASICS parameters , such as ADC and DDC here, but also DAC (Digital to Analog Converter) and DUC (Digital Up-Converter) in a transmitter context. The DSP may also change these parameters at run-time. ADC and DDC are represented in a unique bloc in Fig. 15, since the only information the DSP needs to know is which media (BIFO_VIM) it has to send data to, and which control information was sent to the 2 ASICs. The connection between the 2 ASICs is described thereafter.

C. A new SW application installed on the previous platform

The Pentek platform, featuring four C6203, is of course supposed to support much more demanding radio applications, such as 3G radio waveforms. We then develop a UMTS FDD baseband chain, implementing DPDCH (data) and DPCCH (control) channels until the intermediate frequency (IF).

After a fixed point functional validation on a PC environment (Visual C++ or Borland), the application is then ported in the embedded multi-DSP platform environment at no cost, since libraries were already existing from FM first experiments. From a practical point of view, this means that once the SW application graph is done in SynDEx and validated on a mono-processor PC architecture graph, the designer just need to take the hardware platform graph of the Pentek platform previously developed for the FM application, and map in one click the UMTS application on the Pentek multi-DSP platform. The methodology automatically generates, thanks to SynDEx and the M4 macro-translator, the *main.c* executive file of each DSP, including IPs launching and synchronizations means. The designer may even choose how many DSPs are needed to run the application in real-time, thanks to the overall execution time prediction provided by SynDEx. Anyway, he/she can directly check it on the platform itself in real conditions. SynDEx helps indeed to make this prediction while offering the possibility to automatically insert timing measurement between all IPs. Then mono-processing runs are carried out for each potential device target, and measurements provide the execution time figures for each IP. Once these figures have been entered in SynDEx, as well as the communication timings, the tool is able to optimize the multi-processor implementation mapping. It provides a timing prediction and the associated code for each processing unit. The designer may explore different architectural possibilities until he/she is satisfied.

Fig. 16 shows the UMTS transmitter application graph within SynDEx environment. Several hierarchical blocks are present in this graph, as well as repetitions. The hardware architecture then comprises 130 instances and this for a single frame execution. One UMTS frame is made of 15 slots, the content of which has been detailed in the bottom half part of Fig. 16.

Fig. 17 shows the Pentek platform HW architecture with four TI C6203 DSPs and one ADC.

D. Very fast platform extension

We saw in the previous paragraphs that the UMTS application porting on the Pentek platform is done at no cost, since SynDEx libraries already exist from the FM case study. It may also be necessary sometimes to rapidly investigate which little changes could happen. Let us consider the context of the UMTS repartition processing between a GPP and several DSPs. In the Pentek platform environment, this means adding a TCP link between one of the platform DSP and the host PC processor. SynDEx already has the necessary TCP library on the PC side. So one only needs to develop the DSP side equivalent library, which only consists in encapsulating the

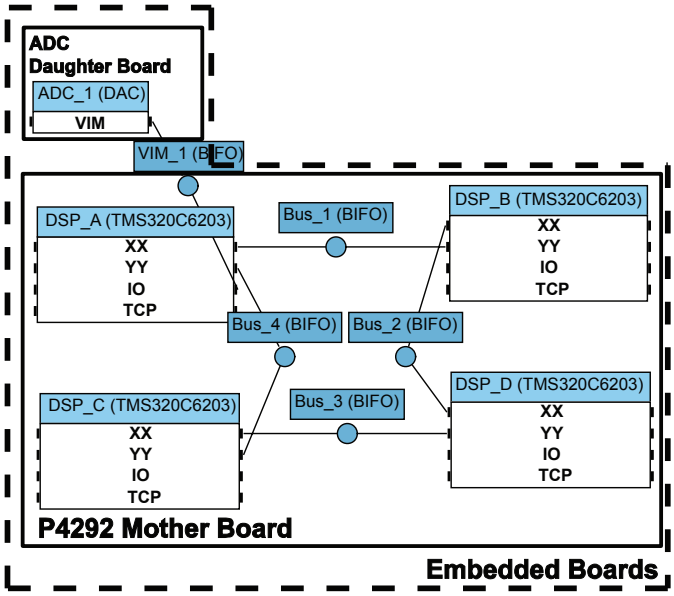

Fig. 17. SynDEx Pentek hardware platform graph.

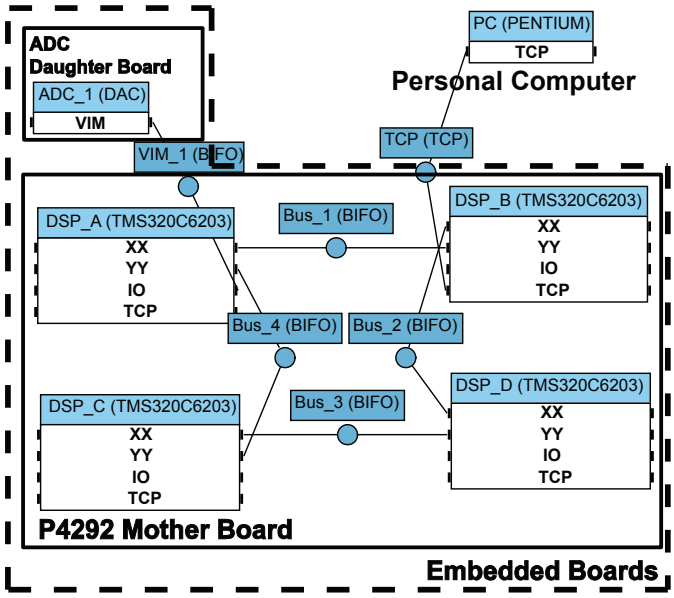

Fig. 18. SynDEx Pentek+PC hardware platform graph.

code provided by Pentek in a SynDEx library. This takes one day.

Fig. 18 shows the SynDEx graph of the new platform involving both PC and DSP, which allows now to automatically provide the code and run the UMTS application in a new environment made of one Pentium and four TI C6203 processors.

This gives the opportunity to rapidly evaluate if new design choices are relevant are not, and this at very low cost and in real conditions. This is exactly the opposite of a manual design methodology, where any architecture exploration is very expensive.

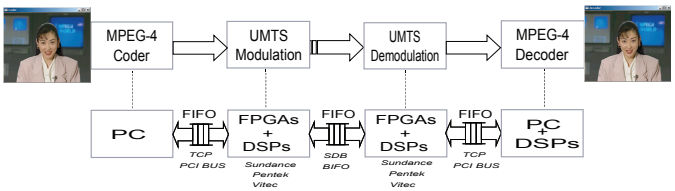

Fig. 19. MPEG-4 over UMTS FDD.

E. Very fast application extension

The platform extension regarding the general purpose processing side is experimented in a cross-layer design mixing video and radio processing. We consider the following application already discussed in [39]: a combined UMTS FDD / MPEG-4 implementation.

This cross-layer implementation benefits from two designs made in two different research contexts, but using the same methodology suggested here. On one hand, the SDR design approach we are illustrating in this paper, and on the other hand an image processing perspective to investigate MPEG-4 coding features. These two designs are being developed in 2 completely distinct manners in 2 distinct SynDEx projects. Each of the projects just has to connect to the same TCP socket, on the PC side for the MPEG-4 application, and on the DSP board side for the UMTS radio. Connection is obtained automatically between the two applications, and MPEG-4 data are transmitted through a UMTS link.

F. Rapid prototyping on a new platform

Now the porting context of this complex application gathering UMTS and MPEG-4 applications is considered. This is yet another issue, which needs to be addressed by SDR design. It has been illustrated in [39] in the case of a Pentek platform made of two C6203 DSPs, and a Sundance platform featuring two C6416 DSPs. We suggest referring to [39] for more information about hardware platform description model, and corresponding developed libraries. The schematic of Fig. [19] displays a porting scenario summary.

Another platform used in this work is a desktop computer associated with the VP3-pmC multi-DSP board. The PC consists of an Intel Core 2 Duo CPU at 2.2 GHz. The VP3-pmC is a parallel programmable processing platform dedicated to professional video applications, like MPEG4-AVC/H.264 real-time encoding and MPEG2 to H.264 trans-coding. This platform comprises 5 DSPs TMS320DM642 running at a 720 MHz clock rate, each of them comprising a 32 MBytes SDRAM and a FPGA hardware co-processor, in order to manage the communications between all platform DSPs and the communications with the PCI-Host. All transfers between DSPs are managed by 5 DMA controllers, which are inside the FPGA.

This platform has been previously described with the SynDEx tool in [51]. The UMTS application is automatically prototyped on this platform, and takes advantages of the methodology: no deadlocks, functional checking of the application portability, and automatic multi-processors implementation. Thanks to the small internal memory, the TI cache memory

TABLE III
TIMINGS OF THE UMTS TRANSMITTER FOR A NEW PLATFORM

1 DSP	2 DSPs	3 DSPs	4 DSPs	5 DSPs
14,3 ms	11,3 ms	11,3ms	11,3ms	11 ms

TABLE IV
TIMINGS OF THE UMTS TRANSMITTER WITHOUT AND WITH FPGA ON THE SUNDANCE PLATFORM

Target	Sundance	Pentek	
Configuration	1*C64x	1*C62x	1*XC2V + 1*C62x
Time/frame	15.9ms	20.2ms	9.9ms
MFL ratio	60%	84%	32%

is automatically used as presented in [52]. Timings, given in Table III, have proven that increasing the number of DSPs for this application was not accurate, since we obtain only a 1.35 acceleration factor from one to two DSPs, and then no more acceleration whatever the number of DSPs are, and this until there are five of them.

Results for this platform are slower than the compulsory standard real-time figure, which is 10ms. This shows that the application graph has to be redefined, in order to extract more potential parallelism so that the hardware platform can perform more parallel executions.

G. FPGA implication

Another extension of the design space exploration is to combine SW and HW processing, with regard to processors and FPGAs. This could be the solution for the real-time issue encountered in the previous paragraph scenario.

Thanks to the development of VHDL M4 libraries, the suggested methodology enables to generate VHDL as easily as C language code from SynDEx. The sole limitation has been exposed earlier: SynDEx is not appropriate to automatically optimize the scheduling and partitioning of multiple HW IPs, since it based on a sequential processing model between IPs. But all the other methodology features are valid, such as the automatic code generation of communications and synchronization means. And in the particular case of only one IP inside a FPGA, it means that the restriction is null and the FPGA is used as a hardware accelerator.

The UMTS FDD baseband transmitter most demanding IP in terms of processing power is the pulse shaping filter. We propose to map the UMTS receiver in the hardware platform of Fig. 20, in order to speed up the receiver execution.

Results of Table IV show the timings obtained after the implementation, without or with an FPGA. An FPGA is used as an accelerating device and permits to respect real-time constraints, since a UMTS FDD frame has to be generated every 10 ms and the combination of a Xilinx Virtex2 FPGA and a TI C62x DSP enables to decrease the frame execution down to 9.9 ms.

H. MPEG-4 Decoder rapid prototyping

An MPEG-4 decoder description has been built for intra pictures in [53], according to the video MPEG-4 texture decoding

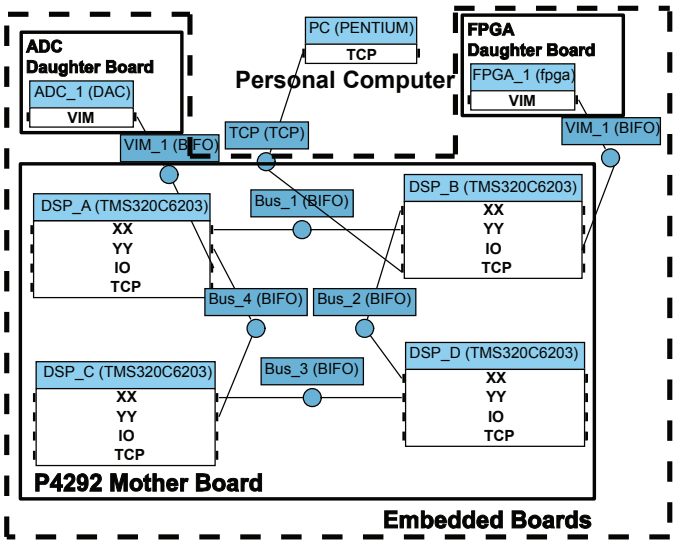

Fig. 20. SynDEx Sundance+FPGA hardware platform graph.

libraries. The operation granularity of those fits IDCT, VLC and dequantization levels within a block. Description granularity has a significant impact on the final implementation. The MPEG-4 decoder in [53] is extended in [52] to Simple Profile. MPEG-4 natural texture coding tools divide intra or predictive pictures into macroblocks, each made of four 8x8 blocks of luminance channel, and the associated 8x8 blocks of chromatic component. Each MB operation has been optimized for a DSP implementation: interleaving loops, conditional tests leading to a pipeline rupture, no dynamic allocations. Our implementation is an open source one, started with the xvid decoder (http://www.xvid.org), and which received the xvid team agreement to be the first porting over a DSP.

Thanks to the methodology developed here, we would like to emphasize how fast an application can be ported from one platform to another one. In this case, this MPEG-4 decoder has been quickly ported over several platforms presented earlier in this paper: Vitec, Pentek and Sundance. It can decode in real-time sequences up to 2048 Mbps with a VGA resolution at 60 frames per second on the TI C6416 DSP at 1 Ghz. The data-flow application development phase, in atomic operations enabling parallelism expression, requested approximately a full year working time for one person working on a Sundance platform. In comparison, the porting over the 2 other platforms is very fast (less than one day) for a first implementation shot as soon as libraries of Table I in V-A0b are available. This proves the efficiency of the design methodology suggested in this paper. For instance, the development of libraries for the Pentek and Vitec platforms takes around 2 weeks time. Library indeed consists in encapsulating the board support package provided by the platform manufacturer. Note that after the first implementation on a new platform obtained in a few mouse clicks with SynDEx, it takes a few more days to optimize execution time if real-time is not reached at first shot.

VII. Conclusion

We have discussed and tried to convince the reader of the advantages of the suggested SDR prototyping methodology. This methodology has been derived from years of experience in terms of prototyping, and enables to produce prototypes within ultra short time delays. It can also be used by students as well, thanks to its simplicity.

The main point to be stressed in this methodology is, that the gathering within a unique framework of several concepts and tools, can and do work indeed. Practical limits are defined and clear objectives are targeted. A component based approach is the basis of the methodology, and combined with efficient and realistic automatic transformations, it allows us to propose a heterogeneous design strategy for SDR prototyping, and in a wider perspective, the design of any embedded equipments. We are convinced that SDR design will not be solved by a brand new design methodology. Moreover, SDR design perspectives are as numerous as radio design perspectives currently are. Other tools may be chosen with regards the rules given in Fig. 4. Moreover, other tools will have to be added to the suggested flow, in order to enhance its current capabilities. In particular, we are investigating higher level design techniques, using UML modelling solutions. Benefits from automatic co-design advances in general will help as well.

A future step of the design solution we suggest, consists in merging the design methodology exposed in this paper with the reconfiguration management that we are investigating in other studies. This will be the SDR design ultimate level we aim to attain.

References

[1] J. Mitola, "The software Radio Architecture," *IEEE Commun. Mag.*, pp. 26–38, May 1995.

[2] J. Reed and B. D. Woerner, *Software Radio: A Modern Approach to Radio Engineering*. Prentice Hall, 2002.

[3] W. Tuttlebee, *Software Defined Radio: Origins, Drivers and International Perspectives*. John Wiley & Sons, 2002.

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

[5] P. Demestichas, G. Vivier, K. El-Khazem, and M. Theologou, "Evolution in Wireless Systems Management Concepts: from Composite Radio Environments to Reconfigurability," *IEEE Commun. Mag.*, May 2004.

[6] A. A. Kountouris and C. Moy, "Reconfiguration in Software Radio Systems," in *Proc. Second Karlsruhe Workshop on Software Radios*, Karlsruhe, Germany, Mar. 2002.

[7] J. Delahaye, C. Moy, P. Leray, and J. Palicot, "Managing Dynamic Partial Reconfiguration on Heterogeneous SDR Platforms," in *Proc. SDR Forum Technical Conference*, Anaheim, USA, Nov. 2005.

[8] X. Revés, A. Gelonch, V. Marojevic, and R. Ferrús, "Software radios: Unifying the reconfiguration process over heterogeneous platforms," *EURASIP Journal on Applied Signal Processing*, vol. 2005, no. 16, Sep. 2005.

[9] S. Paquelet, C. Moy, and L-M.Aubert, "RF Front-End Considerations for SDR Ultra-Wideband Communications Systems," *RF Design*, Jul. 2004.

[10] C. R. Anderson, "A Software Defined Ultra Wideband Transceiver Testbed for Communications, Ranging, or Imaging," PhD dissertation, Virginia Tech, 2007.

[11] J.-P. Delahaye, C. Moy, P.Leray, and J. Palicot, "Partial Reconfiguration of FPGAs for Dynamical Reconfiguration of a Software Radio Platform," in *Proc. of IST Mobile and Wireless Communications Summit*, Budapest, Hungary, Jun. 2007.

[12] G. Gogniat, T. Wolf, W. Burleson, J.-P. Diguet, L. Bossuet, and R. Vaslin, "Reconfigurable Hardware for High-Security/ High-Performance Embedded Systems: The SAFES Perspective," *IEEE Trans. VLSI Syst.*, vol. 16, no. 2, pp. 144–155, Feb. 2008.

[13] J. Bier, "Use a Microprocessor, a DSP or both?" in *Proc. Embedded Systems Conference*, Apr. 2007.

[14] C. Szyperski, *Component Software, Beyond Object-Oriented Programming*. Addison-Wesley, 1998.

[15] Www.omgmarte.org.

[16] C. Moy, M. Raulet, S. Rouxel, J. Diguet *et al.*, "UML Profiles for Waveform Signal Processing Systems Abstraction," in *Proc. of SDR Forum Technical Conference*, Phoenix, USA, Nov. 2004.

[17] J. Laurent, E. Senn, N. Julien, and E. Martin, "Functional Level Power Analysis: An Efficient Approach for Modeling the Power Consumption of Complex Processors," in *Proc. DATE04*, Paris, France, 2004.

[18] D. Elléouet, Y. Savary, and N. Julien, "An FPGA Power Aware Design Flow," *Lecture Notes in Computer Science, Springer*, vol. 4148, 2006.

[19] S. Rouxel, J. Diguet, N. Bulteau, J. Carre-Gourdin, J. Goubard, and C. Moy, "UML Framework for PIM and PSM Verification of SDR Systems," in *Proc. of SDR Forum Technical Conference*, Anaheim, USA, Nov. 2005.

[20] *Joint Tactical Radio System (JTRS) Standard Modem Hardware Abstraction Layer Application Program Interface*, May 2007, version 2.11.1.

[21] G. Gaillard, E. Nicollet, M. Sarlotte, and F. Verdier, "Transaction Level Modelling of SCA compliant Software Defined Radio Waveforms and Platforms PIM/PSM," in *Proc. of DATE*, 2007.

[22] C. Lucarz, M. Mattavelli, J. Thomas-Kerr, and J. Janneck, "Reconfigurable media coding: a new specification model for multimedia coders," in *Proc. of the IEEE Workshop on Signal Processing Systems*, 2007, pp. 481–486.

[23] "Open SystemC Initiative, SystemC v2.0.1," www.systemc.org.

[24] F. Charot, M. Nyamsi, P. Quinton, and C. Wagner, "Architecture Exploration for 3G Telephony Applications Using aHardwareSoftware Prototyping Platform," in *Proc. of Computer Systems: Architectures, Modeling and Simulation*, Amos, Greece, 2003.

[25] A. G. Kleppe, J. Warmer, and W. Bast, *MDA Explained - The Model Driven Architecture : Practice and Promise*. Addison-Weslay, 2003.

[26] S. Rouxel, G. Gogniat, J. Diguet, J. Philippe, and C. Moy, "Schedulability Analysis and MDD," in *From MDD Concepts To Experiments And Illustrations*, J. Babau, J. Champeau, and S. Gérard, Eds. Wiley, Sep. 2006, pp. 111–130.

[27] www.mopcom.fr.

[28] S. Lecomte, S. Guillouard, C. Moy, P. Leray, and P. Soulard, "A co-design methodology based on Model Driven Architecture for Real Time Embedded systems," *Mathematical and Computer Modelling Journal, ed. Elsevier*, 2010, to be published.

[29] C. Moy, "High-Level Design Approach for the Specification of Cognitive Radio Equipments Management APIs," *Journal of Network and System Management - Special Issue on Management Functionalities for Cognitive Wireless Networks and Systems*, Mar. 2010, to be published.

[30] B. Plunkett and J. Watson, *Adapt2400 ACM Architecture Overview*. QuickSilver Whitepaper, 2004.

[31] R. Rabineau, D. Lattard, Y. Durand, M. Lobeira, and J. Rossi, "Flexible Test-Bed for B3G Systems," in *Proc. of IST Mobile and Wireless Communications Summit*, Mykonos, Greece, Jun. 2006.

[32] Y. Durand, C. Bernard, and D. Lattard, "FAUST: On-chip distributed architecture for a 4G baseband modem SoC," in *Design & Reuse IP-SoC*, Grenoble, France, Dec. 2005.

[33] V. Bose, "Design and Implementation of Software Radio Using a General Purpose Processor," Ph.D. thesis, MIT, Jun. 1999.

[34] T. Turletti and D. Tennenhouse, "Complexity of a Software GSM Base Station," *IEEE Commun. Mag.*, Feb. 1999.

[35] A. Hoffmann, H. Meyr, and R. Leupers, *Architecture Exploration for Embedded Processors with LISA*. Kluwer Academic Publishers, Dec. 2002.

[36] P. Coussy, C. Chavet, P. Bomel, D. Heller, E. Senn, and E. Martin, "GAUT: A High-Level Synthesis Tool for DSP applications," in *High-Level Synthesis: From Algorithm to Digital Circuit*, P. Coussy and A. Morawiec, Eds. Springer, 2008.

[37] D. Kammler, L. Godard, and I. Gomez, "Second Report on ASIP Design Methodoloies," NEWCOM, Tech. Rep. DR4.4, Feb. 2007.

[38] J.-P. Calvez and V. Perrier, "SOC Architecting and Design with CoFluent Studio, Concepts and. Methodology -Part I," www.cofluent.com.

[39] M. Raulet, F. Urban, J. Nezan, C. Moy, O. Deforges, and Y. Sorel, "Rapid Prototyping for Heterogeneous Multicomponent Systems: an MPEG-4 Stream Over an UMTS Communication Link," *special issue*

on Design Methods for DSP Systems of Eurasip Journal on Applied Signal Processing, vol. 2006, pp. 1–13, 2006.
[40] J. Muttersbach, T. Villiger, H. Kaeslin, N. Felber, and W. Fichtner, "Globally-Asynchronous Locally-Synchronous Architectures to Simplify the Design of On-Chip Systems," in *Proc. of 12th IEEE International ASIC/SOC Conference*, Sep. 1999, pp. 317–321.
[41] A. Rhiemeier, "Benefits and Limits of Parameterized Channel Coding for Software Radio," in *Proc. of 2nd Karlsruhe Workshop on Software Radios*, Germany, Mar. 2002.
[42] J. Palicot and C. Roland, "FFT: a basic function for a reconfigurable receiver," in *Proc. of 10th International Conference on Telecommunications*, vol. 1, 2003, pp. 898–902.
[43] C. Moy, J. Palicot, V. Rodriguez, and D. Giri, "Optimal determination of common operators for multi-standards software-defined radio," in *Proc. of 4th Karlsruhe Workshop on Software Radios*, Karlsruhe, Germany, Mar. 2006.
[44] V. Rodriguez, C. Moy, and J. Palicot, "Install or invoke?: The optimal tradeoff between performance and cost in the design of multi-standard reconfigurable radios," *Wiley InterScience, Wireless Communications and Mobile Computing Journal, Special Issue on Cognitive Radio, Software Defined Radio And Adaptive Wireless Systems*, vol. 7, no. 9, pp. 1143–1156, 2007.
[45] J.-P. Delahaye, "Plate-Forme Hétérogène Reconfigurable : Application à la Radio Logicielle," Ph.D. thesis, Université de Rennes 1, Apr. 2007.
[46] www-rocq.inria.fr/syndex/.
[47] T. Grandpierre and Y. Sorel, "From algorithm and architecture specifications to automatic generation of distributed realtime executives: a seamless flow of graphs transformations," in *Proc. of 1st ACM and IEEE International Conference on Formal Methods and Models for Co-Design (MEMOCODE)*, Mont Saint-Michel, France, Jun. 2003, pp. 123–132.
[48] M. Raulet, M. Babel, J.-F. Nezan, O. Deforges, and Y.Sorel, "Automatic Coarse Grain Partitioning and Automatic Code Generation for Heterogeneous Architectures," in *Proc. SIPS*, Seoul, Korea, Aug. 2003.
[49] T. Grandpierre, C. Lavarenne, and Y. Sorel, "Optimized rapid prototyping for real-time embedded heterogeneous multiprocessors," in *Proc. of 7th International Workshop on Hardware/Software Codesign (CODES)*, Rome, Italy, May 1999, pp. 74–78.
[50] M. Raulet, "Optimisations Mémoire dans la méthodologie adéquation Algorithme Architecture pour Code Embarqué sur Architectures Parallèles," Ph.D. thesis, INSA of Rennes, France, 2006.
[51] A. Maccari, J. Nezan, F. Urban, and M. Raulet, "Interconnected distributed RAM in SynDEx," in *Workshop on Design and architectures for Signal and Image Processing, DASIP*, Grenoble, France, 2007.
[52] F. Urban, M. Raulet, J.-F. Nezan, and O. Déforges, "Automatic DSP cache memory management and fast prototyping for multiprocessor image applications," in *14th European Signal Processing Conference, Eusipco*, Florence, Italy, Sep. 2006.
[53] J.-F. Nezan, O. Déforges, and M. Raulet, "Fast prototyping methodology for distribued and heterogeneous architectures: application to Mpeg-4 video tools," *Design Automation for Embedded Systems*, 2005.

Christophe Moy was born in 1972 in La Roche sur Yon, France. He is an INSA engineer (INSA stands for: National Institute of Applied Sciences), Rennes, France, and graduated in 1995. He received his M.Sc. and Ph.D. degrees in Electronics in 1995 and 1999 at the INSA institute. He worked from 1995 to 1999 on spread spectrum and RAKE receivers for the IETR (Institute on Electronics and Telecommunications of Rennes). He then worked 6 years for the Mitsubishi Electric ITE-TCL research lab, where he was focusing on Software Radio systems and concepts, including digital signal processing, HW and SW architecture, co-design methodology and reconfiguration. He also addressed the design of SDR impulse UWB systems. He represented Mitsubishi Electric at the SDR Forum and worked on the French research program A3S (design of SDR through a Model Driven Architecture UML-based methodology), as well as the IST project E^2R phase 1. He is now a Professor in SUPELEC. He carries his research in the IETR based SCEE lab, which focuses on Software and Cognitive Radio. He addresses heterogeneous design techniques for SDR, as well as cross-layer optimization topics. He is involved at the European level in the IST Network of Excellence NEWCOM++, and SEC EULER project. He has also been participating to several French ANR project on SDR design, named Idromel and Mopcom.

Mickaël Raulet was born in 1977 in Saint-Brieuc, France. He received his postgraduate certificate in signal, telecommunications, images, and radar sciences from Rennes University in 2002, and his Engineering degree in electronic and computer engineering at the National Institute of Applied Sciences (INSA), Rennes Scientific and Technical University. Then, in 2006, he received a Ph.D. degree in electronic and signal processing, which was delivered by the INSA institute in collaboration with the software radio team of Mitsubishi Electric ITE (Rennes –France). He is currently working at the Institute of Electronics and Telecommunications of Rennes (IETR), as a research engineer focusing in rapid prototyping of video standard compression algorithms on embedded architectures (multi-DSP architecture) and he is also the Image Group project leader of French Media and Network projects, such as Scalim@ges (MPEG-4 SVC main topic) and Mobim@ges (DVB-H main topic). Since 2007, he has been involved in the ISO/IEC JTC1/SC29/WG11 standardization activities (better known as MPEG), such as a Reconfigurable Video Coding Expert.

His interests include video standard compression and telecommunication algorithms and rapid prototyping on multi-DSP architectures from Texas Instruments.