DYNAMIC MAPPING OF WAVEFORMS WITHIN THE PHAL EXECUTION ENVIRONMENT

Vuk Marojevic (Dept. Signal Theory and Communications, Universitat Politècnica de Catalunya, Barcelona, Spain; marojevic@tsc.upc.edu); Ismael Gomez (Dept. Signal Theory and Communications, Universitat Politècnica de Catalunya, Barcelona, Spain; ismagom@gmail.com); José Salazar (Dept. Signal Theory and Communications, Universitat Politècnica de Catalunya, Barcelona, Spain; jose.salazar@tsc.upc.edu); and Antoni Gelonch (Dept. Signal Theory and Communications, Universitat Politècnica de Catalunya, Barcelona, Spain; antoni@tsc.upc.edu)

ABSTRACT

This paper presents a computing resource management approach within the SDR execution environment PHAL. PHAL provides several features, including timing management and resource awareness, for an efficient computing resource management. Simulations show its capability for mapping a modular SDR application or waveform to the distributed computing resources in real-time, thus, facilitating the dynamic reconfiguration of SDR platforms.

1. INTRODUCTION

A heterogeneous radio environment characterizes the emerging 4th generation of wireless communications. SDR technology facilitates taking full advantage of a composite radio environment, where a wireless user can receive personalized services over any suitable air interface [1]. SDR is not limited to the radio access but rather applicable to the entire radio system. This system consists of user or mobile terminals, base stations and core networks.

Future radio systems will be flexibly (re)usable and mostly consist of programmable or software-reconfigurable processors, such as general-purpose processors (GPP's), digital-signal processors (DSP's), and field-programmable gate arrays (FPGA's). An execution environment that features some suitable middleware is necessary to be able to efficiently use these heterogeneous processing platforms for running software-defined radio applications.

An SDR execution environment needs to be aware of the computing resources' states of the underlying hardware at any time. This facilitates an efficient and real-time reconfiguration of the SDR platform and, thus, a dynamic switch from GPRS to UMTS or to any other radio standard of today or tomorrow without changing the radio equipment (Fig. 1).

The rest of the paper is organized as follows. Section 2 briefly examines some related work. Section 3 presents the most salient features of the PHAL execution environment, before describing our approach to computing resource management in software-defined radio. Sections 5 and 6 discuss the simulations and conclusions.

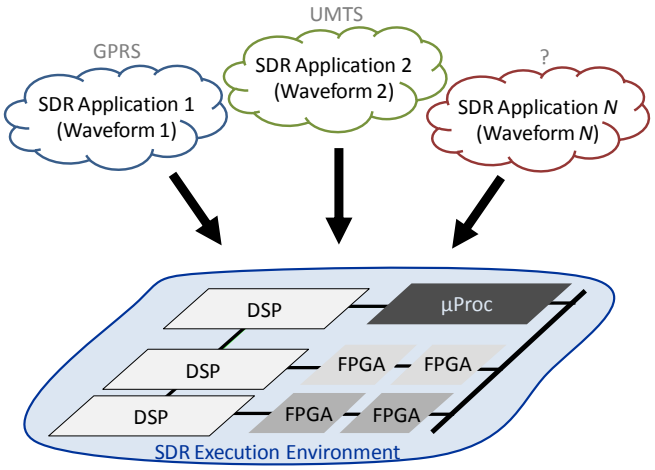

Fig. 1. Dynamic mapping of waveforms.

2. RELATED WORK

Execution environments as well as computing resource management are important topics in SDR research [2]. These two issues have mostly been separately addressed.

2.1. SDR Execution Environments

SDR execution environments include the JTRS's software communications architecture (SCA) [3] and Virginia Tech's OSSIE (open source SDR implementation embedded) [4], which is an SCA implementation targeting embedded systems. SCA concisely defines an architecture for the communication between software and hardware objects

based on CORBA. SCA currently assumes GPP devices; other processor types, such as DSPs and FPGAs, will be included in future releases. It relies on the underlying (POSIX-compliant) operating system that guarantees real-time execution of waveforms on best-effort basis.

Reference [5], on the other hand, presents a non-CORBA based software framework and several protocols for software and hardware configurations. PHAL (Platform and Hardware Abstraction Layer) [6] is another SDR execution environment that does not rely on CORBA but rather on a proprietary *communications manager*, which we describe in section 3. Other SDR execution environments exist, but their discussion is out of the scope of this paper.

2.2. Computing Resource Management

Graph partitioning and scheduling techniques for SDR applications have been studied in [7]. It examines the problem of optimally scheduling waveforms to platforms consisting of 2 DSPs that are connected via bus of certain speed. The optimization objective is the application's speedup.

We have argued for a more general view of computing resource management in software-defined radio and have presented the corresponding framework in [8]. This framework features a resource modeling, which contemplates for different platform architectures and (number and types of) processing devices, and a computing resource management approach that allows for different objective or cost functions.

2.3. Q-SCA

Q-SCA (QoS SCA) is an SDR execution environment with quality of service (QoS) capabilities. It adds an admission controller and resource allocator to the SCA framework and introduces a resource modeling, which specifies the processing, dataflow, and latency requirements of waveforms [9]. Our approach is alternative to Q-SCA; [10] provides a recent comparative study between PHAL and SCA.

3. PHAL EXECUTION ENVIRONMENT

PHAL is an execution environment or middleware for SDR systems [6]. It accounts for:

- Heterogeneous SDR platforms and applications;
- Synchronization and time management;
- Execution control and resource monitoring;

We discuss these principal features of PHAL in continuation; [6] provides further details.

3.1. Heterogeneous SDR Platforms and Applications

PHAL, in principal, can handle any processing platform and application. As regards the platforms, PHAL works on GPP's, DSP's, or FPGA's. Fig. 2 indicates 3 platforms or, equivalently, one platform with 3 processors, where each platform or processor contains a processor-specific HW API.

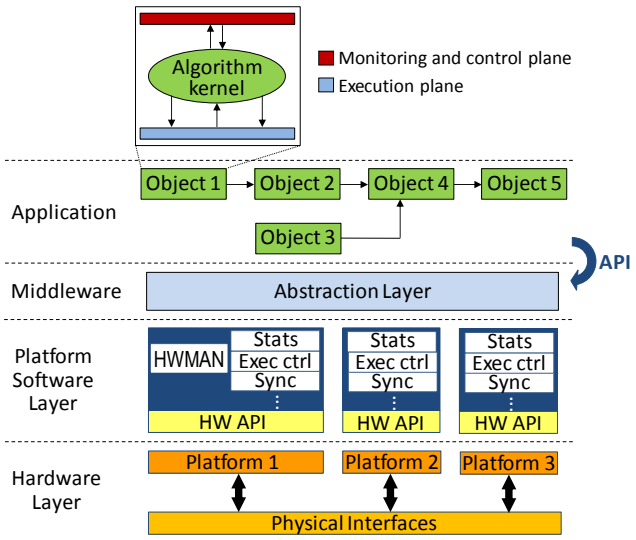

Fig. 2. PHAL's building blocks.

SDR applications or waveforms need to be programmed following a few simple programming rules: Software modules represent objects that contain an *init* phase, a *run* phase, and a *stop* phase. These objects communicate with each other using the PHAL software API (Fig. 2). This way any two modules that interchange data do so via well-defined interfaces.

3.2. Synchronization and Time Management

PHAL synchronizes the clocks on all devices at a certain and modifiable time granularity, where all devices follow a master clock of a chosen master device. PHAL considers processing time as a computing resource and manages it on time slot basis. Therefore, the processing time is divided into time slots. Each time slot executes part of the waveform in a pipelined fashion (Fig. 3).

This time management facilitates guaranteeing the real-time execution of waveforms and greatly simplifies the mapping and scheduling process. More precisely, since time is an implicitly resource a mapping that uses 100% or less of any available resource for waveform processing can meet the timing requirements. These timing requirements are radio service specific and are given as minimum data-rate and maximum latency demands [8].

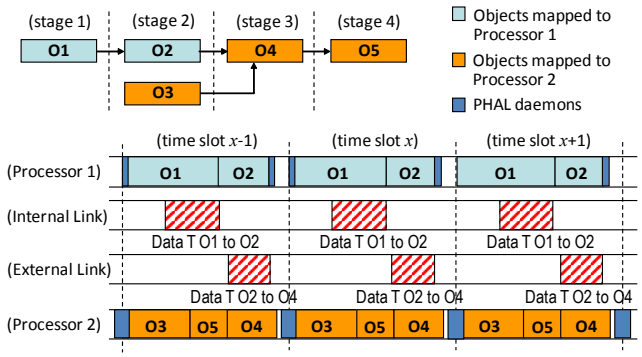

Fig. 3. Time slot division and pipelining.

This rigorous time management during the execution phase of an application also applies to other (pre- and post-execution) phases, where the application and its initialization parameters (e.g. filter coefficients) are loaded, variables are retrieved or updated, and so forth. This permits a deterministic characterization and control of the information latency at any time.

Fig. 3 indicates that PHAL requires a certain amount of computing resources for the time management, among others. This resource overhead is measured as less than 10% for a time-slot duration d_{ts} of 1 ms [6]. In other words, PHAL needs less than 100 µs per processor and 1 ms time-slot, leaving 900 µs of each time-slot for actual waveform professing.

3.3. Execution Control and Resource Monitoring

The software objects' structure (the *init*, *run*, and *stop* phases) permits PHAL to start, pause, step, and stop the execution of applications. It also provides the means for monitoring variables and environmental statistics in real-time. This is suitable for debugging but also for a cognitive resource management: PHAL is aware of the available and occupied computing resources of the underlying platform at any time instant.

Since PHAL may run on heterogeneous platforms, which generally contain different types of processing devices, additional abstraction layers are necessary for providing useful information on the platform's computing capacities and the application's computing requirements. We assume that these abstraction layers exist and that they translate computing resources and requirements to suitable metrics. We have found that MOPS (million operations per second) is suitable for characterizing the processing resources and requirements, whereas Mbps (mega-bits per second) effectively characterizes the bandwidth resources and demands [8]. On the basis of this information, PHAL can monitor and efficiently manage the given computing resources.

4. SDR COMPUTING RESOURCE MANAGEMENT

4.1. Context, Motivation, and Objectives

SDR computing resource management is important for mobile and fixed radio equipment. It manages the limited computing resources of mobile terminals and of radio infrastructure (base stations, core network, etc.). Single-user mobile terminals are very limited in processing and energy resources. Radio infrastructure is not so much constrained on these resources. On the other hand, an efficient management could minimize its operating cost. Furthermore, the computing resource management on one side (mobile terminals) may have implications on the resources of the other side (radio infrastructure) and vice versa. This section does not discuss all these issues but rather presents a simple computing resource management approach, motivating future research.

SDR computing resource management needs platform support, that is, an SDR execution environment that is aware of the platform's occupied and available computing resources at any time. Timing synchronization is necessary for the distributed real-time execution of waveforms. PHAL provides these features (see section 3). The reconfiguration process should, finally, be transparent to the user and not interrupt the current service provisioning.

4.2. Resource Modeling

In [8] we have proposed a resource modeling on time-slot basis. There we have suggested two matrices to model the processing powers (in million operations per time slot – MOPTS) and the platform's inter-processor bandwidths (in mega-bits per time slot – MBPTS) and two for modeling the corresponding resource requirements. The units MOPTS and MBPTS are obtained from multiplying the, more general, units MOPS and Mbps by the time slot duration d_{ts} [in seconds per time slot]. PHAL's function *HWMAN* (Fig. 2) keeps track of these resources in real-time [10].

We assume that the software modules (SDR functions) and their resource requirements in the above formats are available for each processor type. Without loss of generality, this paper considers a single SDR function's implementation—as received over the air (OTA) form an application server—that can run on any processor.

4.3. Mapping Algorithm and Cost Function

Any algorithm that is able to correctly manage the computing resources, as provided by PHAL, is applicable. We distinguish between the algorithm and the cost function, which actually manages the computing resources. In order to allow for different computing resource management policies, we suggest a general-purpose algorithm

(subsection 4.3.1) and policy-specific cost functions (subsection 4.3.2).

4.3.1. t_1-mapping

The t_1-mapping is a dynamic programming approach. It considers one SDR function at a time, starting with f_1 and finishing with f_M. It maintains N possible mapping options throughout the mapping process. Therefore, f_i is pre-mapped to each processor. For each of these mappings one of the N mapping paths, connecting the $(i–1)$ processor-function pairs associated with functions f_1 to $f_{i–1}$, is chosen. After having processed SDR function f_M, the algorithm chooses its mapping (processor) that is associated with the minimum cost. Backtracking the corresponding path form this processor-function combination gives the final mapping out of N (partially) different paths [8].

4.3.2. Cost Function

A simple cost function instance consists of two superimposed terms, the cost of computation and the cost of communication, to manage the most important computing resources *processing powers* and *inter-processor bandwidths*. The cost of computation is defined as the quotient between the processing requirement of a given function and the remaining processing capacity of a given processor. The cost of communication sums the quotients between the bandwidth requirements and the available capacities on the corresponding links. Resources are dynamically updated for a correct computing resource management. Assignments that would require more resources of any type than available are infeasible and are given infinite costs [8].

5. SIMULATIONS

5.1. SDR Application (Waveform)

Fig. 4a shows the functional diagram and the processing requirements of the chip- and bit-rate processing of a UMTS downlink receiver. This processing chain consists of 17 pipelining stages [8]. If we assume that the time slot duration is $d_{ts} = 10^{-3}$ s, the pipelining latency becomes 17 ms, which is reasonable.

The waveform's processing requirements have been obtained as the number of MAC's (multiply-accumulate operations) times the sampling frequency f_s [in Hz], whereas the bandwidth demands have been assumed as 16 bits per sample times f_s. This provides the computing requirements in MOPS and Mbps, which are translated to MOPTS and MBPTS as indicated in section 4.2.

This processing chain constitutes part of the waveform that should run on the SDR platform in its new configuration. We assume that the transmitter and the other processing layers (of the entire SDR-UMTS transceiver) are separately mapped and that sufficient resources are available for doing so.

5.2. SDR Platform

We consider one SDR platform of 3 processors. Fig. 4b shows its graphical and Fig. 4c its mathematical models. The platform's total processing capacity is $C = C_1 + C_2 + C_3$. The bandwidth capacities between each pair of processors, as defined in the bandwidth matrix $\boldsymbol{B}$, map to the same physical link (of bandwidth B) which can be scheduled in a round robin fashion. A bandwidth resource update, however, needs an update of all B-matrix entries (except for those on the main diagonal that model processor-internal communication capacities). This way PHAL models and manages shared communication resources.

5.3. Scenarios

We consider different scenarios to simulate the fact that the computing resources vary from platform to platform and may be a function of previous configurations, those that are not completely substituted through partial reconfigurations. Therefore we assume several platform states: C takes the values 2.5, 2.65, …, 4 MOPTS and B the values 0.5, 0.75, …, 2.5 MOPTS. Since a platform may remain in different (processing heterogeneity) states, we simulate the following 4 cases:

I. $C_1 = C_2 = C_3 = C/3$;
II. $C_1 = 1.25 \cdot C/3$, $C_1 = C/3$, $C_1 = 0.75 \cdot C/3$;
III. $C_1 = 1.50 \cdot C/3$, $C_1 = C/3$, $C_1 = 0.50 \cdot C/3$;
IV. $C_1 = 1.75 \cdot C/3$, $C_1 = C/3$, $C_1 = 0.25 \cdot C/3$.

5.4. Results and Discussion

Fig. 5 shows the simulation results. It indicates the feasible and infeasible mappings for the simulated platform conditions. An infeasible mapping means that the t-mapping algorithm is not able to find a mapping that satisfies the waveform's computing resource requirements with the available resources. In general, such a mapping could not guarantee the waveform or service-dependent real-time constraints and is, thus, useless for the wireless user. Whether this infeasibility is the algorithm's failure or due to the lack of resources is out of the scope here.

It is interesting to observe that the higher the heterogeneity in the processing capacities, the less bandwidth resources are necessary to obtain a feasible mapping. This is so because the higher the processing power of one processor, the more SDR functions can be finally executed on it, requiring less inter-processor data transfers and, thus, less bandwidth. (For the theoretical case that one processor could execute the entire SDR application, no bandwidth resources would be necessary.)

The execution time of the t_1-mapping algorithm is around 50 µs on a 2.4 GHz general-purpose processor. Hence, the mapping can be computed during a single time slot (of 1 ms), adding only negligible overhead to the less than 100 µs per time slot (see section 3.2). This verifies the suitability of the mapping approach for a dynamic mapping of waveforms within the PHAL execution environment.

The algorithm's short execution time makes it also possible to pre-compute a mapping on the basis of the real-time resource information to decide whether or not to download a certain waveform to a given SDR platform or, which waveform to download in case of multiple options. This could, on the other hand, be part of the learning process of a cognitive engine. Both, PHAL and our computing resource management framework are apt for cognitive radios, providing a cognitive execution environment [10] for a cognitive computing resource management [11].

6. CONCLUSIONS

This paper discusses the dynamic mapping of waveforms within the PHAL execution environment. PHAL is an SDR framework that is capable of performing computing resources management of heterogeneous computing resources. Therefore it facilitates the information of the momentary required and available computing resources of waveforms and platforms in real-time. We have presented the viability of one computing resource management approach; others are possible.

Finally we have indicated the suitability of PHAL's computing resource management for cognitive radios. Cognitive radio mainly focuses on spectrum management. Nevertheless, spectrum efficiency—in terms of bits per Hertz—is mainly achieved at the expense of additional computing cycles. These cycles are limited and, thus, need to be properly managed as well.

REFERENCES

[1] J. Mitola, "The software radio architecture," *IEEE Commun. Mag.*, vol. 33, no. 5, pp. 26–38, May 1995.

[2] SDR Forum Web Site, http://www.sdrforum.org/

[3] SCA Website, http://sca.jpeojtrs.mil/

[4] Open Source SCA Implementation Embedded (OSSIE) Web Site, http://ossie.wireless.vt.edu/trac/wiki

[5] S.-L. Tsao, C.-C. Lin, C.-L. Chiu, H.-L. Chou, M.-C. Wang, "Design and implementation of software framework for software defined radio system," *Proc. IEEE 56th Vehicular Technology Conference (VTC 2002-Fall)*, 24-28 Sept. 2002, pp 2395-2399 (vol. 4).

[6] X. Revés, A. Gelonch, V. Marojevic, R. Ferrús, "Software radios: unifying the reconfiguration process over heterogeneous platforms," *EURASIP Journal of Applied Signal Processing*, vol. 2005, no. 16, Sept. 2005, pp. 2626–2640.

[7] A.-R. Rhiemeier, "Modulares software defined radio," Ph.D. dissertation, Forschungsberichte aus dem Institut für Nachrichtentechnik der Universität Karlsruhe (TH), Band 9, Karlsruhe 2004.

[8] V. Marojevic, X. Revés, A. Gelonch, "A computing resource management framework for software-defined radios," *IEEE Trans. Comput.*, to be published.

[9] J. Lee, S. Kim, J. Park, "Q-SCA: Incorporating QoS support into software communications architecture for SDR waveform processing," *Real-Time Syst (2006)*, pp. 19-35, Springer Science + Business Media, LLC 2006.

[10] I. Gomez, V. Marojevic, J. Salazar, A. Gelonch, "A lightweight operating environment for next generation cognitive radios," *Proc. 11th Conf. digital Systems Design (Euromicro)*, 3-5 Sept. 2008, to be published.

[11] V. Marojevic, N. Vucevic, X. Revés, A. Gelonch, "Cognitive computing resource management for a ubiquitous wireless access", *Proc. 4th Int'l Conf. Ubiquitous Intelligence and Computing (UIC'07)*, Hong Kong, July 11-13, 2007, LNCS vol. 4611/2007, pp. 808-818, Springer Berlin / Heidelberg.

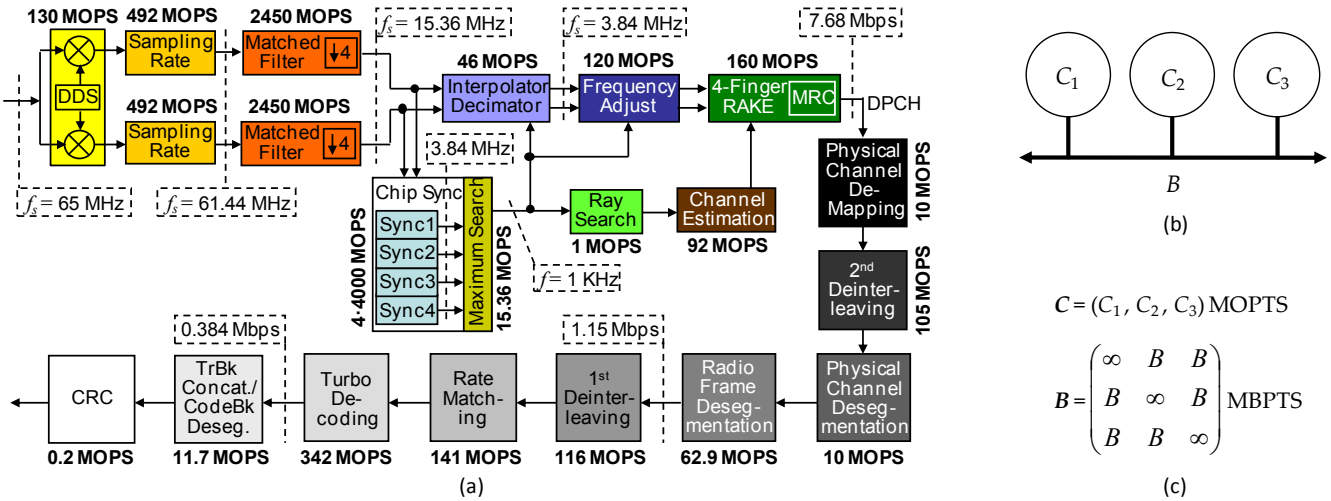

Fig. 4. Chip and bit-rate processing of SDR-UMTS Receiver [8] (a); SDR platform graphical (b) and mathematical (c) models.

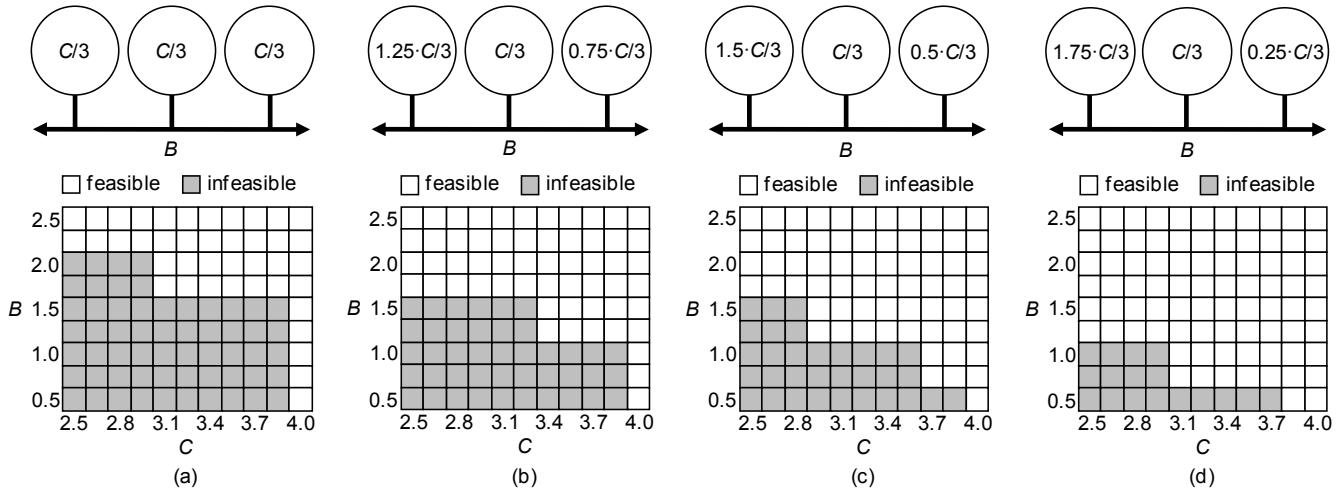

Fig. 5. Mapping results for processing heterogeneities I (a), II (b), III (c), and IV (d).