Reconfigurability: A Key Property in Software Radio Systems

Apostolos A. Kountouris Ph.D., Christophe Moy Ph.D., Luc Rambaud

***Abstract*--This paper gives an overview of issues related to the reconfigurability property that software radio based equipment needs to satisfy for the benefits from the software radio concept to become commonplace. Even though current technology limits impose deviations from the ideal all-software- radio, the advent of high density reconfigurable hardware (FPGAs) as well as proper system engineering and design, make the software radio concept a viable reality. In this context reconfigurability needs to be designed in the system and so it becomes a system design issue. However, since system heterogeneity becomes predominant, more adapted design and system development methodologies have to be adopted. Hardware/Software Co-design offers interesting possibilities. Finally, as reconfigurability and reconfiguration open up SWR systems to the outside world in an unprecedented way, issues like security, safety, reliability, become of extreme importance.**

***Index Terms*--Software Radio, Reconfigurability, Reconfiguration, HW/SW Co-design.**

I. Introduction

The wireless communications landscape is very diverse and constantly evolving. Briefly, it is characterized by things such as: multiple and evolving standards, different types of equipment for subscribers and operator infrastructure, different and time varying transmission environments (in-building, urban, open space, high or low terminal mobility, etc.). Finally, various actors are involved (subscribers, network operators, service providers, equipment manufacturers) each with different motivations, objectives and expectations from new technologies.

Software Radio (SWR) is a concept [1] that has been gaining momentum in such a diverse set-up because it promises important benefits for every actor in this landscape. As far as the benefits from software radio are concerned, there is general consensus as the relevant literature and previous work indicate. The interested reader is referred to [1], [2], [3] for more information. However, turning the concept into commercial products is another story in which system level design issues are crucial. One of the goals of discussion groups like the SDR Forum [4] and the EU *Reconfigurable Radio Colloquium* [5] is to look into these matters.

This paper gives a general overview and analysis of issues related to a fundamental property in Software Radio: *Reconfigurability*. This property permits mobile communication devices (terminals, basestations and networks) to adapt to their constantly changing external environment. For each such device we can define its external environment in respect to the entities with which it communicates in one way or another. Taking this view one can understand that SWR devices such as mobile terminals and basestations have many things in common and a lot of differences as well since they are used in different ways. However the reconfigurability property, which is not to be confused with reconfiguration, is a common denominator. SWR equipment should be able to change (re-configure) its radio functionality part. Without this ability (reconfigurability) it makes little sense to talk about SWR.

The rest of the paper is organized as follows: first, some background information is given; second, the reconfigurability property and its implications in reconfiguration and software radio system design are described; next, it is examined how hardware/software co-design methodologies are suitable to address issues relating to the increased heterogeneity in software radio systems; finally, open issues are discussed and some conclusions are drawn.

II. Background Information

In this section some necessary background information permitting to make explicit the relation of software radio, reconfigurability and reconfiguration is given. From this discussion it becomes apparent that as we deviate from the ideal *all-software-radio*, reconfigurability is not given but it needs to be designed in the system.

A. Software Radio

The term *software radio* was first coined by J. Mitola in [1]. In its ideal realization *wideband* signal digitization occurs next to the antenna and the rest of the radio processing is implemented in software running on a very fast general purpose processor. This concept is depicted in FIG. 1.

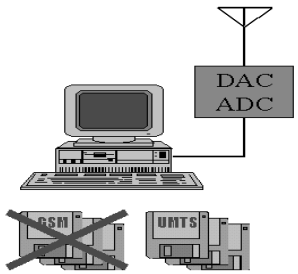

FIG. 1: Ideal SWR where all radio processing is in software

A. A. Kountouris, C. Moy and L. Rambaud are with the Mitsubishi Electric ITE, Telecommunication Laboratory (TCL), Rennes, FRANCE (tel.: +33-(0)2-99842110, e-mail: {kountouris, moy, rambaud}@tcl.ite.mee.com).

Software radio is all about tailoring the concept of reprogrammability to the domain of radio communications so as to enable the use of *generic* hardware/radio platforms for different types of radio applications just by changing the software.

However due to current technology constraints such as lack of wideband ideal A/D interface, limited processor speeds etc., alternative architectures need to be devised to describe the currently feasible precursors of the ideal software radio while waiting for the enabling technologies to catch up.

B. Reconfigurability

The concept of reconfigurability has been existing for a long time. For instance, the use of microprogrammed control in CISC processor designs allowed for flexibility in both the design process as well as product maintenance (bug fixes, upgrades etc.) and evolution. The use of microcontrollers or other programmable components in embedded systems provides a high degree of in-system programmability that makes the system more flexible. Another very popular example is the MARS Pathfinder incident where failure of an entire space mission was avoided thanks to the ability to remotely *reconfigure* the system to fix a bug of repeated systems resets due to a priority inversion problem [6]. In radio applications reconfiguration capabilities can be seen in digital TV decoders, upgradable modems over wireline, as well as cell phones based on Java that run downloaded Java applets [3].

So, one can say that software download and reconfiguration of reconfigurable platforms is quite commonplace. What is new with software radio is the application of such capabilities in the domain of radio communications to include the radio (*air-interface*) functionality as well. In the case of the ideal all-software-radio reconfigurability is already built-in the system since microprocessors can be reprogrammed just by changing the contents of their program memory. As it will become apparent later on, the reconfigurability property is not readily available in currently realistic software radio systems but it has to be explicitly designed in the system during the early stages of its design.

C. Reconfiguration

Reconfigurability and reconfiguration are intimately related. Their relation is graphically shown in FIG. 2.

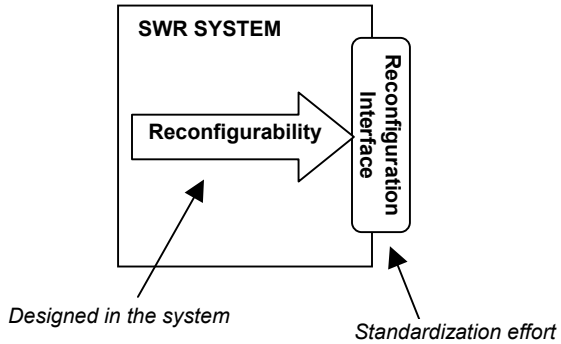

FIG. 2: Reconfiguration builds on reconfigurability.

Reconfiguration builds on reconfigurability. It is the process of changing the configuration of a given hardware to make it operate in some other way. Since reconfigurability is a system property, how to embed it in the system is left up to the equipment designers. On the other hand, reconfiguration of the air-interface targets a variety of equipment possibly from different manufacturers and is visible to the outside world. Being so, common interfaces have to be provided and naturally this is the subject of standardization activities.

The approach taken by SDRF [4] on reconfiguration is to propose extensions to existing standards like WAP [7] and MExE [8] that currently cover software download and reconfiguration of the application layer, to include the reconfiguration of the air-interface. Given that such devices handle not only radio transmission but interaction with their user entities as well, software radio devices based on WAP/MExE including SDRF extensions will provide for the reconfiguration of both radio and non-radio functionality in a *unified* framework.

In the elaboration of the reconfiguration methods several parameters that define various reconfiguration scenarios have to be considered. One, is the reason for reconfiguration; for instance, the requirements for a bug-fix are not the same as the requirements for switching to another standard. Another pertains to what is reconfigured, e.g. radio or other functionality. The reconfiguration data source and destination and the physical communication medium for the reconfiguration data transfer influence the process as well. For instance, over-the-air reconfiguration (OTA), has different implications than using a local link or the network. Finally, another factor is whether the reconfiguration has to happen *on-line* during equipment's normal operation, or *off-line* when the equipment is in stand-by; both ways correspond to realistic situations.

III. Reconfigurability and System Design

In this section issues concerning system design for reconfigurability will be discussed. A system is a collection of interconnected components plus the logic required in order to make components operate correctly individually and as a whole. We can view a system as a *hardware engine* and the harware/software *logic* driving it.

A. System Architecture for SWR Devices

As it was previously mentioned current technological constraints make the ideal SWR impractical. Nevertheless, alternative architectures exist on which reconfigurable radios can be implemented. A high-level block diagram of such an architecture is shown in FIG. 3.

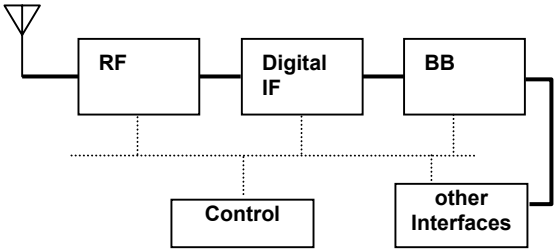

FIG. 3: Block diagram of a reconfigurable radio architecture

At the receiver side we see an analog RF part that translates a wide input band of interest down to IF where the input analog signal is digitized using a wideband A/D interface. At

this stage digital hardware takes us to the baseband where additional digital hardware is in charge of the baseband processing. The transmitter side takes the opposite direction. Additional external interfaces as well as the system control part are also represented.

At the RF part there are analog circuits consisting of filters, amplifiers, mixers, etc. Since this part is specific for each standard, it potentially limits the reconfigurability of the whole system. A certain degree of reconfigurability can be achieved by replication and switch as shown in FIG. 4. In this case the reconfiguration consists in a switch command that changes the hardware connections.

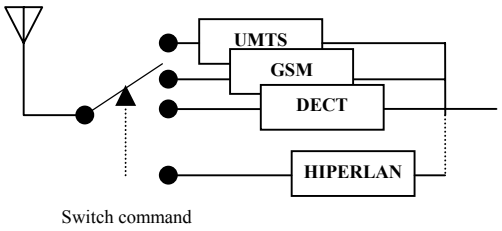

FIG. 4: Limited reconfigurability of the RF part

The digital processing part is taken care by a mix of ASIC and DSP components. DSPs being reprogrammable they offer maximum reconfigurability. On the other hand the reconfigurability of ASICs depends on the number of operational parameters that can be set by software and usually their number is limited. For instance a digital modulator/demodulator can be configured for a limited number of modulation schemes. DDC decimation and DUC interpolation factors can take certain values. Filter components may further inhibit reconfigurability since different radio application may have quite different filtering requirements. Finally, connections between the various components cannot be rearranged at will since usually a limited number of by-passes is provided between individual system components.

So, even though this architecture has a large digital part its reconfigurability has important limitations. However, the availability of high-density SRAM based FPGAs makes reconfigurability of the hardware functionality as high as that of software's. Features like fast reconfiguration times, partial reconfiguration and remote access to the FPGA resources, are also desirable [9], [10] for SWR equipment. Even though today, their use in terminal equipment seems impractical mainly due to power consumption and size considerations, in BTS equipment they can easily replace many of the ASICs that limit reconfigurability. FPGAs have an important role to play in SWR in infrastructure basestation equipment.

B. Reconfigurability and the System Lifecycle

The lifecycle of a SWR system can be broken down into the stages listed below:

- requirements analysis and specification
- design and implementation
- deployment
- change

In this context we have to distinguish the first time development, from successive modifications/upgrades of the system during its lifetime. Reconfigurability is all about the capability of a system to change while reconfiguration is the process of changing the system's way of working. It is important to understand that in order to effectively and seamlessly accommodate change, reconfigurability has to be designed in the system.

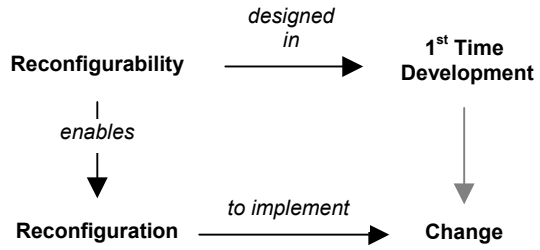

FIG. 5: Reconfiguration exploits reconfigurability in order to change

These ideas are graphically depicted in FIG. 5. With *reconfigurability* built-in during its *1st time development*, a system can be considered under constant development in order to implement *changes* in the way it operates. These changes are deployed via *reconfiguration* which is *enabled* by the property of reconfigurability.

C. Design for Reconfigurability

The first time design usually is guided by the desire to offer backwards compatibility (e.g. 3G basestations, supporting 2G as well) and the possibility to anticipate the future needs. Anticipating the future is very difficult to handle because of the uncertainty factor. Sometimes even if future trends are correctly forecasted minor technical details may compromise a design. To cope with these problems several design practices are of help.

Briefly, the use of programmable components increases the system flexibility; independently of whether we talk about microprocessors or FPGA's, their functionality can be changed after deployment by simply changing the contents of some memory. Finally, if ASIC's are a necessary choice, circuits that offer a certain degree of flexibility by parametrization are to be preferred over ones that may function in only one way. In addition, overdesigning the systems gives extra resource margin by underutilizing resources in terms of what is needed at the time the system is initially designed. This margin permits to fit the same hardware engine to future needs.

What a hardware engine can do is only limited by the hardware characteristics of its constituent components. These define a sort of operational range or in other words what the hardware is capable of doing in terms of maximum resource utilization. A hardware engine with flexibility and reconfigurability designed in, may satisfy future needs that do not demand more resources than available, by simply changing the software. Future needs outside this range prompt for hardware changes and so hardware modularity would allow to localize the impact of such hardware engine upgrades. This process is graphically depicted in FIG. 6 (next page).

Finally, system flexibility is further increased by the adoption of modular hardware and software architectures. Hardware modularity was used in the SPEAKeasy project [11]. The concept of *"plug & play"* should be applied in both sys

tem dimensions: hardware and software. Thus, system design simply becomes the composition of a set of modules both at the hardware and software levels. Such a compositional view strongly depends on clear and well defined interfaces and permits for instance to isolate the modifications in a small part of the system, make extensive reconfiguration in an incremental manner.

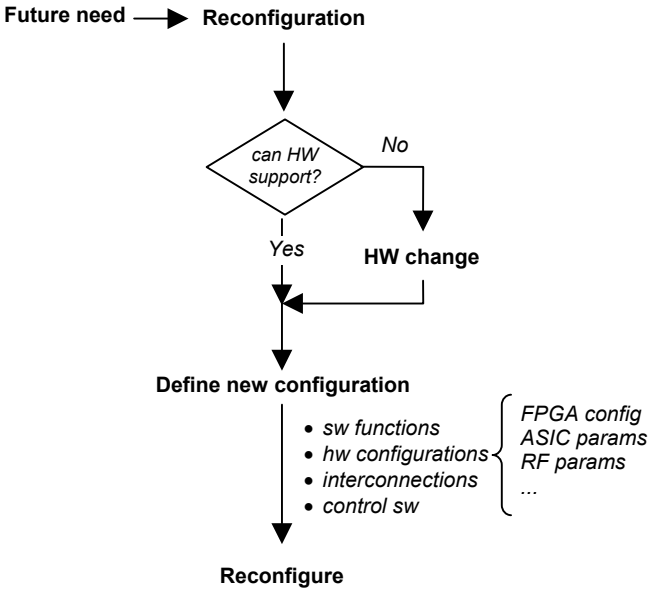

FIG. 6: Future needs prompt for reconfiguration

To summarize, reconfigurability depends on the programmability of the hardware components or sub-systems, programmability of their interconnections and modular hardware and software architectures. Modularity permits equipment to adapt to future needs which are mostly unknown at design time and facilitates also the reconfiguration deployment process since thanks to modularity changes can be localized into specific system parts.

IV. HW/SW Co-Design for Software Radio

As it was already mentioned, given the technology shortcomings in DSP speed, moving the digital processing closer to the antenna necessitates the introduction of specialized hardware processing in the form of ASICs and/or FPGAs which offer higher reconfigurability. The introduction of high density FPGAs puts at the disposal of designers a certain amount of silicon surface that can be freely reused when future needs prompt for system reconfiguration. This is one of the reasons for considering hardware/software co-design methodologies.

In general what we have is a collection of processing nodes of specific or general nature that may be interconnected in many ways depending on the communication needs between system processing nodes. Furthermore, the coexistence of reconfigurable hardware (FPGAs) and microprocessing elements opens up the possibility to implement part of the functionality in software and another part in hardware. Thus, designing reconfigurable radio systems becomes simply an instance of the hardware/software co-design problem. Co-design is important due to the need to be able to rapidly develop the desired configuration for our flexible hardware engine and thus materialize the benefits from reconfigurability.

Such methodologies permit to cope with the increased *heterogeneity* present in wireless communication systems. Heterogeneity is an important parameter and for this reason in the next section a brief overview is given.

A. Manifestations of Heterogeneity in Wireless Equipment

In the context of wireless communications equipment heterogeneity manifests itself in various distinct but closely interrelated levels: the *signal* level, the *algorithmic* level and the *system* level.

At the signal level radio architectures are inherently heterogeneous in the sense that analog processing (components) co-exists with digital processing. As technology evolves, digitizing the analog signals will move closer to the antenna and this will eventually make the analog part smaller. At the algorithmic level, in wireless equipment, heterogeneity manifests itself in yet another way, control processing is mixed with digital signal processing. At the system level and specifically in the digital processing part, heterogeneity manifests itself by the co-existence of ASICs, FPGAs and microprocessors (DSPs, microcontrollers etc.). Heterogeneity further increases by the introduction of multi-processing where different types of processing elements are mixed to best accommodate the peculiarities for different types of processing like for instance control or DSP, fixed or floating point etc.

B. Brief introduction to hardware/software co-design

HW/SW co-design methodologies, as the name indicates, address the problems related to the simultaneous development of both the hardware and software parts of a system. Why this is desirable in respect to traditional system development lifecycles, is out of the scope of this paper but the interested reader is referred to [12], [13]. Having specified the desired system behavior co-design helps in finding an adequate architecture to implement the specified functionality. An adequate architecture is one that represents the best performance-cost trade-off out of a number of possible solutions, the design space.

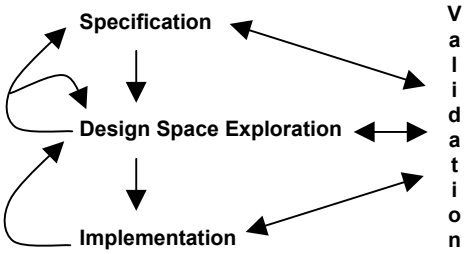

FIG. 7: HW/SW Co-design is an iterative process

As shown in FIG. 7, co-design methodologies take the designer from specification down to implementation through an iterative process in which *design space exploration* occupies a central place. After having specified the desired system functionality/behavior in an implementation independent manner, the design space of possible solutions is searched to find the solution that respects the non-functional constraints.

Given a target architecture a partitioning of the functionality into software and hardware is found. Finally, in the implementation phase the hardware, software and communications between processing elements are synthesized. The latter includes the hardware/software synthesis.

Throughout the co-design process at each phase validation activities are employed in order to validate if the design objectives during a design phase have been met and decide whether to iterate or continue to the next phase. *Validation* techniques include simulation, formal verification, estimation, prototyping etc.

C. Open issues

Though research in co-design has gone a long way several issues still remain open and some of them are exacerbated by the very nature of radio equipment and the many ways heterogeneity manifests itself in such systems. There is a need to take a global approach that includes under the same framework analog and digital design, the control and DSP (dataflow) dimensions. Previous work in co-design has addressed separately DSP systems [14] and embedded control systems [15], [16], with more or less flexibility in choosing a target architecture, etc. Other co-design systems especially for DSP telecom applications offer the ability to mix analog and digital design aspects [17].

Finally the mix of hardware and software functionality in the digital processing part still lacks a unified approach. Regarding this last point there is a weak point in the unification of software and hardware specification. Several initiatives try to address this issue by proposing a common high-level description languages. Others propose common internal representations, co-simulation etc. to make various formalisms coexist. The SystemC initiative takes the former approach [18]. Even though the use of a single formalism is a step to the right direction, the heart of the problem lies in the fact that traditionally hardware and software are described in different abstraction levels. Software is described at the behavioral level and hardware at the register transfer level (RTL). This has to do with the efficiency of existing implementation (synthesis, compilation) tools. So even if the same language is used nothing is really achieved if the abstraction levels remain different. The following may effectively contribute in closing the semantic gap:

- Advances in DSP high-level language (e.g. C) compilation with the advent of VLIW architectures in the DSP arena [19], [20]. VLIW processors will eventually close the gap between DSPs and general purpose processors.
- Advances in hardware High-Level Synthesis (HLS) or behavioral synthesis [21], [22]. HLS research has resulted in production tools like [23], [24].
- Automatic or computer assisted derivation of efficient and equivalent fixed point descriptions from floating ones [REF: Fridge].

These elements will permit a complete implementation independent specification of system functionality that will eventually be implemented in either hardware or software.

Even though SWR raises a variety of issues to be addressed by co-design methodologies these remain the best approach to design such systems rapidly and consistently. They also permit to facilitate development of new system configurations in order to answer future needs.

V. Issues to be Addressed

Reconfigurability and reconfiguration from one hand make possible to obtain the benefits from software radio implementations but from the other introduce a series of concerns relating to security, safety, reliability to name a few. Software radio systems become increasingly open to the outside world and thus increasingly vulnerable as well. As far as safety is concerned, think what may happen if a reconfigurable radio equipment is reconfigured to function in a way dangerous for its user. In terms of security questions like, -what if an unauthorized entity, uses reconfiguration in order to obtain confidential information about the user or the equipment itself, obtain unauthorized access to the network or break down the equipment-, are posed. Finally, reliability is summarized in the following question: what if an authorized reconfiguration makes the equipment unstable and its operation unreliable. In equipment with limited reconfigurability the ways that something can go wrong are also limited and users are accustomed to reliable operation. If such issues are not addressed adequately wireless equipment success will be compromised. As indicated in [25], tackling reliability and robustness issues is an important avenue in future systems research.

To answer these questions several things come to mind. Security has to be designed in the reconfiguration process. The introduction of Java in the internet provides interesting insight in many of these issues and also in ways to approach these problems.

Reliability may also be achieved if reconfigurable radio systems are considered mission critical with fault tolerance techniques built in. Experience from work in military as well as in space programs could be of great importance in designing secure, safe and reliable software radio systems. Formal techniques in specification and verification have also interesting contributions when developing new configurations and for validating them before deployment.

VI. A SWR Experimentation Platform

In our laboratory an experimentation platform was assembled to study SWR system design, reconfigurability, reconfiguration and eventually find some answers to the aforementioned problems, especially reliability. Experimentation will also give us insight on how reconfigurability needs to be customized for different types of equipment according to the needs and expectations of their respective users (subscribers for terminals and operators for infrastructure equipment). What seems reasonable for a particular type of equipment may be of no use for another.

This platform, shown in FIG. 8, has a modular hardware architecture making future hardware upgrades possible. The presence of multiple DSPs will enable the study of multiprocessing issues; at a second time the introduction of FPGA modules will permit to apply co-design techniques and address hardware and software reconfiguration in a unified manner. Finally, this platform will permit to study the inter

actions and behavior of the various entities in a cellular network when reconfiguration of the air-interface occurs.

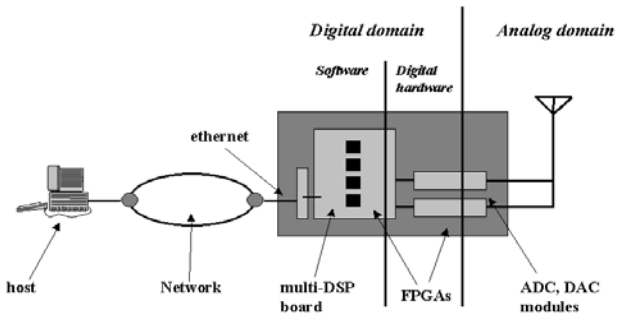

FIG. 8: Experimentation platform

VII. CONCLUSIONS

Software radio is not only about software implementations of DSP algorithms that will execute efficiently enough (in terms of time, memory and power consumption) to eliminate the need of using air-interface specific digital and analog front-ends. Software radio is also about the process of changing a device's radio operation; *reconfiguration*. For reconfiguration to be possible software radio systems have to be designed for reconfigurability. This in turn is a design challenge since all software radios are not currently feasible and reconfigurable hardware logic needs to be included. In addition modular software and hardware architectures should also be employed. For such heterogeneous designs hardware/software co-design methodologies can be of use throughout the product life, helping in initial system development as well as on the development of future system changes.

In other words, we need to have the algorithms but even more we need to be able to install them in a secure way, statically or dynamically, ensuring that communication and correct operation are not compromised. In wireless cellular communications things are even more complex since correct operation involves several actors and different type of equipment throughout the network and their interactions in respect to the reconfiguration of their radio functionality part needs to be studied. For re-configurable radio equipment to gain acceptance several requirements concerning reliability, security and safety need to be satisfied. In fact wireless communications equipment should be considered as mission critical.

In software radio for mobile communications, the system-level issues are predominant. Software radio research is a multi-disciplinary task. Expertise in radio telecommunications needs to go hand in hand with expertise in algorithms, computer science/engineering and system design as well as expertise in information technology.

VIII. ACKNOWLEDGMENTS

The authors would like to thank Damien Castelain for his support and the colleagues at SDR Forum for the fruitful discussions that helped the authors form their views on certain aspects of SWR.

IX. REFERENCES

[1] J. Mitola, *"The Software Radio Architecture"*, IEEE Communications Magazine, vol. 33, pp. 26 - 38, May 1995.
[2] W. Tuttlebee, *"The impact of Software Radio"*, EU Software Radio Workshop, Brussels, 1997.
[3] W. Tuttlebee, *"Software Radio Technology: A European Perspective"*, IEEE Communications Magazine, Feb. 1999.
[4] Software Defined Radio Forum, SDRF, http://www.sdrforum.org/
[5] First European Colloquium on Re-configurable Radio Systems & Networks, 4 March 1999, Q.E. II Conference Center, London, UK.
[6] M. Jones, *"What really happened on Mars Rover Pathfinder"*, The Risks Digest, ACM, http://catless.ncl.ac.uk/Risks/19.49.html#subj1.
[7] WAP, http://www.wapforum.org/what/WAP_white_pages.pdf
[8] ETSI, SMG04 MExE, http://www.etsi.org/.
[9] D. Nicklin, Xilinx, Electronics Engineering Magazine, *"Utilizing FPGAs in Re-configurable Basestations And Software Radios"*.
[10] Xilinx whitepaper, *"Remote Field Updates Using FPGAs"*, http://www.xilinx.com/xilinxonline/remote_wp.htm.
[11] P. G. Cook, W. Bosner, *"Architectural Overview of the SPEAKeasy System"*, IEEE JSAC, vol. 17, no 4, pp. 650 - 661, Apr. 1999.
[12] Special issue on HW/SW Co-Design, IEEE Design and Test of Computers, vol. 10, no 3, Sep. 1993.
[13] W. Wolf, *"Hardware Software Co-Design of Embedded Systems",* Proceedings of the IEEE, 82(7), Jul. 1994, pp. 967-989.
[14] A. Kalavade, E. A. Lee, *"A Hardware/Software Codesign Methodology for DSP Applications",* IEEE Design and Test of Computers, vol. 10, no 3, pp. 16 - 28, Sep. 1993.
[15] G. De Michelli, R. Gupta, *"Hardware - Software Cosynthesis for Digital Systems",* IEEE Design and Test of Computers, Sep. 1993.
[16] F. Balarin, M. Chiodo, P. Giusto, H. Hsieh, A, Jurecska, L. Lavagno, C. Passerone, A. Sangiovanni-Vincentelli, E. Sentovich, K. Suzuki, B. Tabbara, *"Hardware-Software Co-Design of Embedded Systems: The Polis Approach"*, Kluwer Academic Press, June 1997.
[17] HP-ADS, http://wwwtmo.external.hp.com/tmo/hpeesof/products/ads/adsoview.html.
[18] SystemC, http://www.systemC.org/.
[19] Thomas J. Dillon, *"The VelociTI Architecture of the TMS320C6x"*, in Proceed. ICSPAT, 1997.
[20] Philips TriMedia, http://www.semiconductors.com/trimedia/.
[21] R. Camposano, *"Tutorial: Behavioral Synthesis"*, in Proc. of 33rd DAC, Las Vegas, Nevada, Jun. 1996.
[22] R. A. Bergamaschi, *"Behavioral Synthesis: An Overview"*, IBM Technical Report, RC20944, 1998.
[23] R.A. Bergamaschi, A. Kuehlmann, *"A System for Production Use of High-Level Synthesis"*, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol.1, no.3, pp. 233-243.
[24] Synopsys Behavioral Compiler, http://www.synopsys.com/products/beh_syn/beh_syn_br.html
[25] John Hennessy, *"The Future of Systems Research"*, IEEE Computer, vol.32, no 8, pp. 277 - 33, Aug. 1999.