Wireless World Research Forum (WWRF)

Reconfigurable Physical Layer Architecture supporting End to End Reconfiguration (E^2R)

(a) Title of the research item:
Hardware Abstraction in an End-to-End Reconfigurable Device

Experts:

Jörg Brakensiek[1]	**jorg.brakensiek@nokia.com**
Dominik Lenz[1]	**dominik.lenz@nokia.com**
Bernd Steinke[1]	**bernd.steinke@nokia.com**
Mirsad Halimic[2]	**Mirsad.Halimic@panasonic-pmdc.co.uk**
Craig Dolwin[3]	**craig.dolwin@toshiba-trel.com**
Syed Naveen[4]	**naveen@i2r.a-star.edu.sg**
Alexis Bisiaux[5]	**bisiaux@tcl.ite.mee.com**

[1]Nokia GmbH, Nokia Research Center, Meesmannstr. 103, D- 44807, Bochum, Germany

[2]Panasonic MDL, Furzeground Way 3, UB11 1DD Uxbridge, Middlesex, Great Britain

[3]Toshiba Research Europe Limited, 32, Queen Square, Bristol, BS1 4ND, Great Britain

[4]Institute for Infocomm Research (I2R), Singapore

[5]Mitsubishi Electric ITE-TCL, Rennes, France

Subject Area: WG 6: Reconfigurability

(b) Objectives of the required research

End-to-end re-configurability has been identified as a major research topic in the European 6th framework program (FP6). Re-configurable physical layer is one of the focus areas, which is targeted in the upcoming FP6 project End-to-End Reconfigurability (E2R), illustrating the vision of End-to-End Reconfigurability [E2R03].
Inside one work package of the E2R project, the focus will be on the investigation and development of re-configurable physical layer resources, enabling true end-to-end re-configurable radios. Therefore it tackles the main problems and challenges in the configuration control technologies for mobile terminals as well as base stations. Such equipment will play a critical role in wireless applications beyond-3G, where ubiquity, flexibility and adaptability are the key concepts.
To achieve a true software-defined physical layer, which is capable of being reconfigured from the network/operator as well as by the user/application, a well-defined abstraction of the hardware and software resources is required.
Hardware abstraction is a research topic, which is currently widely discussed in the reconfigurability community (e.g. specific RFI in the SDR Forum) as physical implementation will be proprietary to manufacturer.
Within this research item, we will present different layers of abstraction suitable for an end-to-end reconfigurable approach, as targeted in the End-to-End Reconfigurability (E2R) project. [BL04]

This work has been performed in the framework of the EU funded project E2R. The authors would like to acknowledge the contributions of their colleagues from E2R consortium.

(c) State of the art in the area

In the past the configuration control for the physical layer in a wireless terminal has been limited

to changes requested by the Radio Resource Controller (RRC) or similar entity defined in the supported standard. These changes would be limited to a subset of the standard, as defined by the terminals class mark, and would include services such as switching between data services and speech calls or a change of speech codec. Each of these different configurations would be known at design time and would be created and tested in DSP/CPU software (sometimes in assembler code but more recently in C) and then implemented in an embedded ROM. A ROM solution was used rather than RAM or FLASH to save silicon area and allow operation at the high clock speeds required in a data processing application. Typically the only mechanism available for modifying the physical layer to implement functionality above and beyond that conceived at design time was a small amount of patch RAM in the program space of the DSP. This was combined with some form of patch vectoring and used to fix bugs in the ROM code.
More recently research projects have been looking at methods for reconfiguring the physical layer to implement functionality not conceived off at design time. At least TRUST [TRUST99], SCOUT [SCOUT01] and CAST [CAST99] that have directly addressed the problem of Configuration Control. A common theme exists i.e. all projects have used object-modelling techniques to encapsulate functionality and all have three primary components, as shown in the following table

TRUST	SCOUT	CAST	E2R
BPC (Baseband Processing Cell)	Proxy	Java Class	[CEM] (Configurable Execution Module)
TMM (Terminal Management Module)	TMM	RSC (Reconfigurable Resource Controller)	[CMM] (Configuration Management Module)
RMM (Reconfigurable Baseband Management Module)	RMM	PLC (Physical Layer Controller)	CCM (Configuration Control Module)

Table 1: Comparison of Configuration Components in FP5 projects with E2R

Both SCOUT and CAST address heterogeneous architectures and recognise the requirement to support communication between the processing elements. Only SCOUT appears to have addressed the issue of timing deadlines. ADRIATIC [ADRIATIC00] addresses this problem indirectly by working on the re-configuration times of the processing elements.
None of these projects directly address the problem of verifying that a new configuration will always operate correctly. MuMoR [MuMoR01] reduce this problem by constraining the configuration space to a number of RAT's.

(d) Possible approach

The physical layer architecture of a reconfigurable device consists of a set of reconfigurable functional elements (RF-frontend, communication, digital processing). They appear as (re)configurable execution modules (CEMs) implementing a specific functionality (e.g. Down-conversion, Modulation, Decoding, Rake). A specific entity (called Configuration Control Module, CCM) reconfigures such modules and also manages the communication resources between to guarantee the required functionality and maintain the required data throughput. [SDR04]

Configuration Control Module (CCM)
The role of the CCM is to supply an abstract configuration interface to the signal processing system in a wireless terminal, basestation or access point.
The CCM is located in the System Abstraction Layer (SAL) or high level Hardware Abstraction Layer (HAL). This layer implements a set of interfaces that allow high level entities to configure resources, i.e. create and link signal processing functions (e.g. modulation, demodulation, channel coding, source coding, RF transceiver, AFE, etc.). By supplying a platform independent interface, called Service API, the E2R system can reconfigure either ends of the wireless link

without a detailed understanding of the underlying implementation.
The configuration of the wireless terminal includes downloading of software (ISA objects, FPGA code, etc.) and parameters into resources to implement different functions as well as the configuration of the communication fabric to implement the required data flow and control flow structures. [DWB04]
Control and supervision of the re-configuration process takes place in co-operation with the Configuration Management Module (CMM) and/or by autonomous processes implemented by dedicated CCM functions and optional modules like the adaptation module.
The CCM will be implemented by means of software modules, build on top of the underlying operational software, i.e. a certain operating system (OS), using its extensions like device driver to gain access to the real hardware interfaces or additional OS extensions to implement diverse CCM functionalities.

Configurable Execution Modules (CEMs)
The Configurable Execution Modules may support different levels of reconfiguration (configuration capabilities):
Dedicated signal processing entities, programmable IPs blocks

- Algorithm specific accelerators
- Programmable logic (e.g. FPGA, CPLD)
- Application tailored DSPs, AIPS
- General purpose processors, DSPs

A manufacturer will exercise the choice and selection of suitable CEMs, implementing certain physical layer functionality. This manufacturer dependency requires an implementation independent view of the physical layer hardware in order to support certain reconfigurations and use cases.

Hardware Abstraction Layer Concept
Hardware abstraction will be used on various levels to support the Configuration Control Module and the Reconfiguration Management plane on top. Especially the exchange of reconfiguration capability parameters between certain entities (e.g. network entities, operator) has to be addressed.

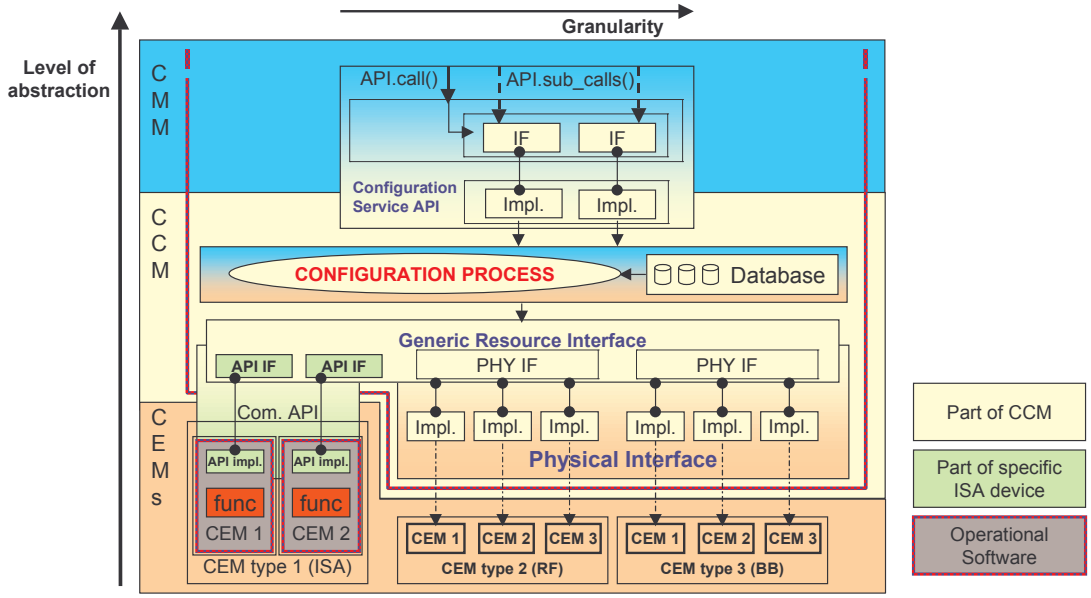

Figure 1: Abstract functional View

Therefore a hardware abstraction layer concept has been developed in the WWI project E2R (End-to-End Reconfigurability), which introduces hardware abstraction on certain levels as shown

in Figure 1:

- The Configuration Control Module can be seen as the high-level abstraction of the re-configuration capabilities of the physical layer hardware. Abstraction has to be done on function rather than implementation level. This level implements a kind of Service API.
- The Generic Resource Interface, abstracts the different classes of Configurable Execution Modules (CEMs) for the CCM.
- The Physical Interface abstracts the real implementation of a CEM
- Intelligent CEMs have at least a basic operating system running on top. Their physical interface will be replaced by a low-level communication API, which enables communication (i.e. exchange of configuration data, program code).

Configuration Service API

The *Configuration Service API* is the mentioned interface towards the upper layer between the Configuration Control Module (CCM) and the platform independent Configuration Management Module (CMM). The Configuration Service API is implemented in the CCM, and is used by CMM. This is a platform independent entity and provides services for the configuration. Different levels of configuration granularity can be envisaged:

- Coarse grained: "configure WLAN chain with param1 = x & param2 = y"
- Mid grained: "configure channel estimation = algo1"
- Fine grained: "configure FIR coefficient1 = 0.654"

Typically the upper level management entity (i.e. the CMM) is using the coarse-grained layer, but optimisation of the radio link or certain adaptation functions, may require the direct access to the mid grain or even the fine grain configuration layer, which could be done either by the CMM or additional adaption or radio link modules.

Inside the Configuration Service API, the coarse grain level will use functions of the mid grain level and the mid grain level those of the fine-grained. Thus, the architecture of underlying hardware is hidden, but assessable in an abstracted common way.

Generic Resource Interface

The intention for the definition of the Generic Resource Interface is to have a common single interface to the lower layer. Abstract HW models will encapsulate front-end, digital baseband resources respective functions and communication resources. The CCM has to interact with the resources available, which are executing the physical layer processing.

The physical implementation will be proprietary and might be organized in several different ways. Functional partitioning will lead into several possible allocations of functionality into multifarious CEMs. In the described approach for the way of reconfiguration, the CEMs can be grouped into two different interface classes:

- **ACTIVE**: ISA devices which have their own firmware respective Operating System (OS), and therefore an API for handling accesses. This is named the Low-level Communication API.
- **PASSIVE**: All other devices without their own firmware respective OS. They do not provide an API, and therefore need an active handling. This is referred to as low-level Physical Interface..

The Generic Resource Interface (i.e. the upper-level interface) is therefore subdivided in the different properties of generally different access-types of CEMs as shown in Figure 1.

Physical Interface

The Physical Interface is part of the low-level HAL for devices without own firmware or OS. They need an active handling of each configuration, which is demanded. Some examples are:

- Analogue circuits: e.g. AD-converter (configure required number of bits)
- Accelerator e.g. upload Microcode
- Programmable ASIC: e.g. Set certain register values & read status registers

The Physical Interface provides mapping of Generic Resource function calls to the proprietary configuration procedures of dedicated CEMs, i.e. the real physical addresses contained in the database are utilized for the re-configuration.

Low-level Communication API

The low-level Communication API provides common services for configuration of ISA devices running under certain OS. They are implemented on certain ISA device. ISA devices have their own firmware respective Operating System (OS), and therefore an API for handling the reconfiguration requests. The configuration through the Communication API includes downloading of software modules and parameters.

A UML description of the described hardware abstraction is shown as a summary in the following figure.

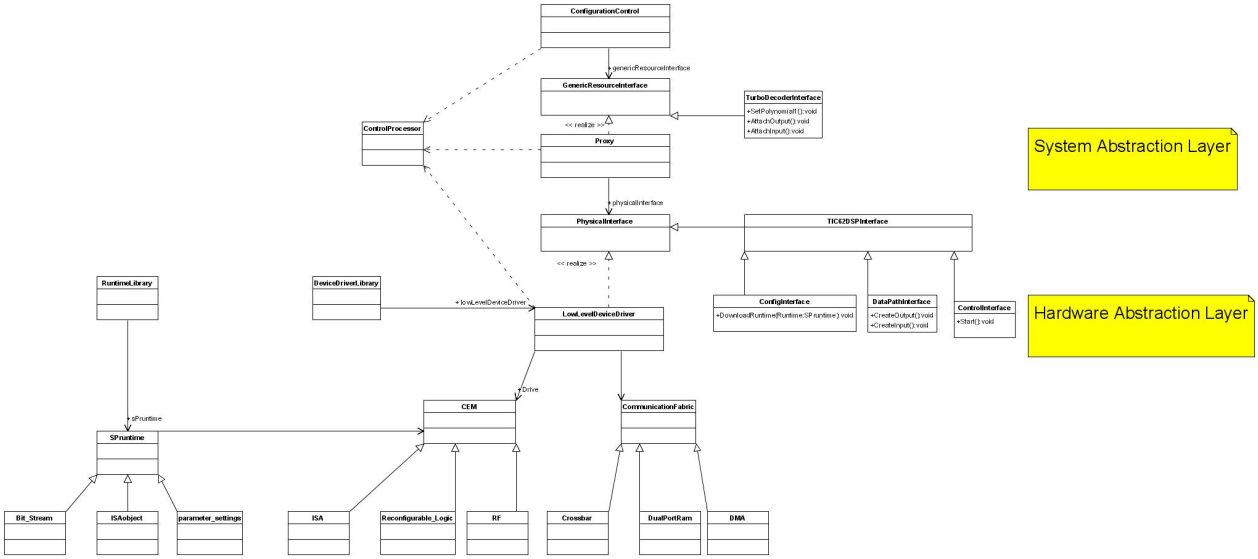

Figure 2: UML Description of the Hardware Abstraction

(e) Expected results

This research work will lead to a clear understanding of the requirements for each relevant abstraction layer within the reconfigurable physical layer and combine different classes of reconfigurable devices. This includes the exchange of capability parameters for the physical layer. The capability parameters are used by the user, service supplier and operator to decide on the most suitable terminal/basestation configuration for the given context.
The abstraction layer will translate an implementation specific configuration, proprietary to a certain manufacturer and specific to a certain implementation, into a functional description. These translations and descriptions have to be generic and flexible enough, in order to serve different requirements. At certain levels, harmonization between manufacturers is required.
Additional certain CEMs will be evaluated and modelled in an abstract way, in order to be integrated into the system architecture.

(f) Time frame to get the expected results

The E2R project is planed in 3 phases, where as in the first phase (2004/2005) the basic concepts and architectures will be developed. Therefore it can be expected that first definitions are available soon and will be verified within certain use cases.
Final results are expected after the final phase in 2007/08.

List of References

[E2R03] End-to-End Reconfigurability (E^2R), IST-2003-507995 E^2R, http://www.e2r.motlabs.com

[BL04] J. Brakensiek, D. Lenz, T. Wiebke, S. Gultchev, R. Tafazolli, A. Bisiaux, C. Moy, A. Kountouris, M. Halimic, C. Dolwin: Management and Controlling Architecture in E2E Reconfigurable Terminals. 3rd Karlsruhe Workshop on Software Radio, Karlsruhe, March 2004

[SDR04] SDR Forum, Hardware abstraction working group: Request for Information – Hardware Abstraction Layer. February 2004.

[DWB04] C. Dolwin, S. Walter, J. Brakensiek, B. Steinke, K. Moessner, *Response to Hardware Abstraction Layer Request For Information*, SDRF-04-I-0007-V0.05, 2004

[CAST99] Configurable radio with Advanced Software Technology, IST-1999-10287, http://www.cast5.freeserve.co.uk

[MuMoR01] Multi-Mode Radio Architecture Platform for enhanced 3G, IST-2001-34561, http://www.mumor.org

[TRUST99] Transparently Reconfigurable Ubiquitous Terminal, IST-1999-12070, http://www4.in.tum.de/~scout/trust_webpage_src/trust_frameset.html

[SCOUT01] Smart user-Centric cOmmUnication environmenT, IST-2001-34091, http://www.ist-scout.org

[ADRIATIC00] Advanced Methodology for **D**esigning **R**econf**I**gurable SoC and **A**pplication-**T**argeted **I**P-entities in wireless **C**ommunications, IST-2000-30049**,** http://www.imec.be/adriatic/