Functional Elements in E2E Reconfigurable Equipment

M. Bronzel, H. Seidel, TU Dresden, Dresden, Germany
J. Brakensiek, D. Lenz, Nokia Research Center, Bochum, Germany
A. Bisiaux, C. Moy, A. Kountouris, Mitsubishi Electric ITE-TCL, Rennes, France
M. Halimic, PMDL, UK, C. Dolwin, Toshiba Research Europe Ltd, Bristol, UK
S. Walter, Alcatel SEL, Stuttgart, Germany, S.K. Pilakkat, I2R, Singapoore
L. Maurer, DICE, Linz, Austria, T. Burger, ACP, Zollikon, Switzerland

email: bronzel@ifn.et.tu-dresden.de

ABSTRACT

Reconfigurable equipment within a Software Defined Radio approach will be addressed on different levels: the control level, the resource level, and the operational level. A reconfiguration architecture and functionality will be introduced that offers interfaces to management entities and controls the internal components to select, negotiate and perform the appropriate reconfiguration actions. The reconfigurable equipment will make it possible to optimize resources for dedicated air interfaces and to update and upgrade existing equipment with new features, functions and procedures.

I. INTRODUCTION

Equipment for efficient and reliable operation in an End-to-End Reconfigurability context as considered in the E2R [1] project, needs a well-defined framework for the local equipment reconfiguration management. Two complementary aspects of the configuration management issue are considered here: functionality and architecture. Both depend on the requirements placed upon the reconfiguration process, which are dictated by the different usage scenarios.

II. RECONFIGURABLE EQUIPMENT

A. Management and Control Architecture

Reconfiguration inside a terminal, base station or access point in an end-to-end manner is affecting all layers from the physical layer up to the network and application layer. The overall picture of reconfigurable equipment (i.e. terminal, basestation or access point) architecture is shown in Figure 1. A distributed hierarchical configuration management approach is adopted. It is distributed in the sense that in an end-to-end reconfigurability context many physically or logically distributed entities are required to collaborate. It is hierarchical since a configuration can have a different representation to different layers of abstraction within the reconfigurable equipment or to entities outside the equipment (e.g. a configuration supporting GSM or UMTS). This representation becomes more and more detailed inside as we approach the real hardware that needs to be configured [2].

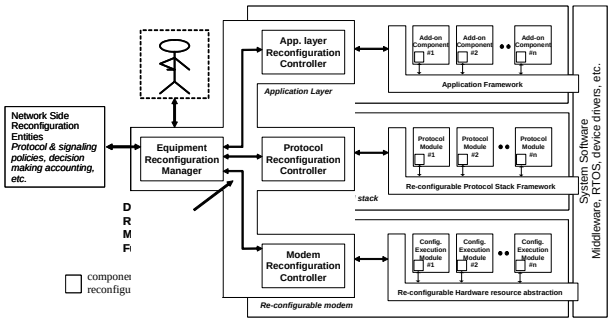

Figure 1: Overall reconfigurable equipment architecture.

The network reconfiguration manager, located somewhere in the network, is interacting with the equipment reconfiguration manager (EqRM), which supervises all equipment related reconfigurations. The Configuration Management Module (CMM) within EqRM takes care of physical layer related reconfiguration issues. It is independent from the underlying physical layer platform or its implementation and interacts with a Configuration Control Module (CCM). Hiding the implementation details of the reconfiguration capabilities of the hardware resources [3], the CCM controls the physical layer (PHY) reconfiguration. Reconfiguration of the equipment's system functionality inside the physical layer will comprise a set of reconfigurable, programmable or parameterizable hardware/software resources available for data and control processing. In particular, the CCM is responsible for controlling and supervising the reconfiguration process as well as for defining fallback modes, in case of mal function. Additionally, it shall provide download capability for complete or partial update of configuration data.

B. Physical Layer Architecture

A physical layer architecture has to be developed, which meets the functional requirements derived from different scenarios and standards. It has to provide the environment which is needed to implement the functionality required by the CCM.
The physical layer architecture of the reconfigurable modem is shown in more detail in Figure 2: A set of functional elements (RF-frontend, communication, digital processing) forms the upper layer of a system of different levels of hardware abstraction.

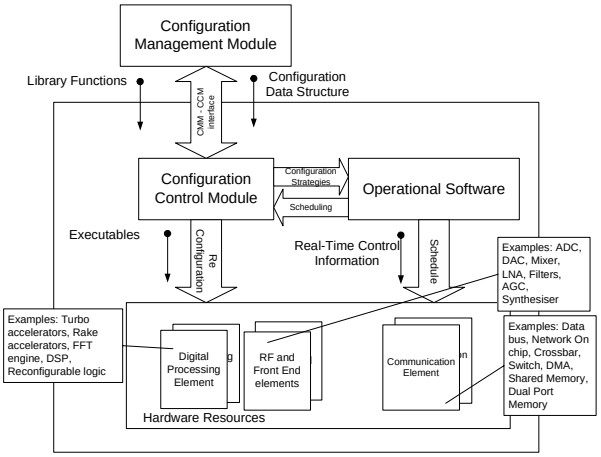

Figure 2: Physical Layer Architecture.

At this level, functional elements appear as reconfigurable modules implementing a specific functionality (e.g. Down-Conversion, Modulation, Decoding). The CCM can configure such modules and also manage the communication resources between the processing elements to guarantee the required functionality and maintain the required data throughput.

Downloading of object code to programmable resources and bit-streams to reconfigurable logic as well as setting parameters of hardware accelerators has to be performed at a lower level of abstraction supported by low level device drivers in order to comply with the principle of hardware abstraction and thus platform independence.

Handling these levels of abstraction and temporal scheduling of hardware and software resources requires appropriate operational software. The challenge of such a software system is to manage the processes which are required to maintain the reconfigurability of the system and at the same time to guarantee the functionality of the underlying data processing system.

The operational software module (OSM) provides a logical view of the execution environment for the higher layers. The execution platform will most likely consist of a mix of general purpose processors (GPPs), digital signal processors (DSPs), field programmable gate arrays (FPGAs), and parameterizable hardware accelerators which are connected by a variety of interconnects. While the CCM manages these resources at a logical level, the OSM provides the primitives to perform these tasks. Operational software incorporates real-time operating systems (RTOS) for various programmable processing elements (PPEs) to allocate and manage the processing resources (e.g. execution time, priority and scheduling policy) for each task that needs to be scheduled on them. Likewise methods for managing the associated memory resources will be provided.

Communication between various processing elements is another critical resource (connectivity, bandwidth, etc.) that the CCM will manage. The OSM will provide the logical interfaces to configure and manage these resources. Processing elements will be allocated to various tasks in a flexible manner by means of a distributed middleware. Supporting the run-time reconfiguration is a critical function of the OSM. Loading and unloading of software modules on the PPEs will also be possible. The OSM will incorporate appropriate loaders for the target PPEs, while providing a common logical interface to the CCM. Similarly the OSM will also include drivers and loaders to load and configure various reconfigurable and parameterizable hardware resources. Other services for support of development and execution such as non-volatile storage, file system, logging service, diagnostics etc. will additionally be provided.

III. CONFIGURATION CONTROL

The CCM is responsible for configuring and scheduling resources in the physical layer to fulfill system configuration requests from the CMM.

A. CCM Requirements

The key requirements for the CCM can be defined as:

1. Platform independent: The functionality of the CCM will be independent of the hardware resources it controls. This ensures design reuse across different nodes (i.e. Terminal, Basestation or Access Point) and scalability as the underlying hardware evolves.

2. Dynamic downloading of Data Processing Modules: The CCM directly manages the configuration at the module level. But by also supporting the dynamic creation of data processing modules the configuration granularity is reduced to a much lower level.

3. Power conscious: Reconfigurability implies increased power consumption. One of the tasks of the CCM is to reduce this overhead to a minimum.

4. Standard interface to the CMM: By supporting a platform independent interface the CCM hides the underlying hardware from higher layers in the configuration plane.

5. Seamless and partial re-configuration: The transition from one configuration to another should be done in a precise and time accurate fashion to avoid loss of functionality to sections unrelated to the change.

6. Reliable, predictable and secure configuration: The CCM must guarantee the correct functionality under hard real-time constraints and has to verify the consistency of configuration.

7. Configuration and control of all stages in the receive and transmit path: RF and AFE components can be configured or switched on or off as required. Sample clocks and time bases will also be configured. In addition to the baseband digital signal processing the CCM will also manage signal processing applications such as vocoders and video codec's.

B. CCM Functionality

As noted above the CCM is responsible for reconfiguring the underlying physical layer in terms of:

- Reconfiguration of the hardware resources
- Reconfiguration of the interconnection scheme
- Temporal & spatial scheduling of a configuration

In order to maintain these functions, several tasks have to be implemented in the CCM which can be logically partitioned into the following classes, which are also depicted in Figure 3.

- Authentication & integrity check of configuration
- Negotiation with CMM
- Memory management
- HW resource management & spatial scheduling
- Monitoring of environment
  - Statistics (used up resources, status information)
  - Metrics (SNR, multi path, interference)
- Maintenance of the databases containing
  - Execution metrics
  - Standards and algorithms
  - Configuration parameters
  - Configuration data
  - SW modules (ISA code)
  - Device driver
  - Gathered statistics & metrics

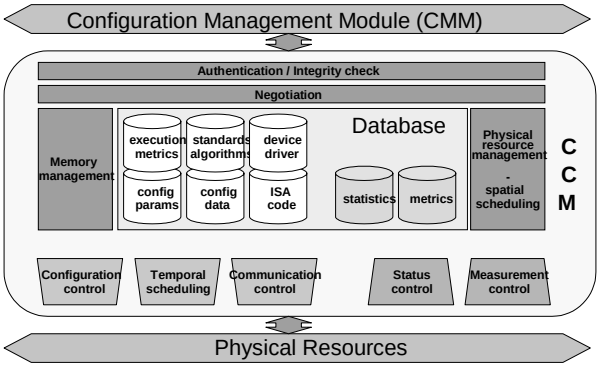

Figure 3: CCM Functional View.

By integrating these processes the CCM should be able to fulfill the requirements and to ensure proper operation of the underlying hardware resources.

IV. RECONFIGURABLE HARDWARE RESOURCES

Reconfiguration for E2R equipment comprises partitioning and configuration of functional and hardware layer. Commonalities and similarities of functional building blocks (FBB) will be identified and then used for partitioning FBBs into common classes. Reconfiguration parameters will be extracted for these classes in order to reduce time and resource requirements for the re-configuration process. The identified classes will be mapped to corresponding classes of processing elements on the hardware layer.

A. HW Resource Configuration

The reconfigurable HW architecture of the physical layer consists of a set of hardware resources (analog and digital) which are configurable for supporting all required digital baseband and RF front-end functions. In the process of reconfiguration, these hardware resources are connected, monitored and controlled by the CCM. For each hardware component the CCM needs information about data, control and reconfiguration interfaces. Among these are:

- Reconfiguration time
- Reconfiguration parameters (range)
- Amount of reconfiguration data
- Partial vs. full reconfiguration
- Reference for execution time
- Reference for power consumption

The processing elements can be constructed from a set of programmable or parameterizable hardware building blocks. Configurable processing elements considered for this platform are micro-controllers, GPPs, DSPs, HW accelerators, applications specific DSPs (ASDSPs), FPGAs, digital and analog parameterizable ASICs. Since type and level of reconfigurability of hardware resources varies from the truly general-purpose components such as Digital Signal Processors or Field Programmable Gate Arrays to dedicated components such as programmable oscillators there is a need for a generic view of hardware resources. Therefore, hardware resources are modeled in a uniform abstract way providing required information to the CCM and hiding the implementation specific details. In addition, this approach enables a seamless integration of a new type of hardware into the architecture.

B. Analog Front-End Considerations

The difficulties of an SDR compliant RF front-end are quite different to the challenges arising in the digital baseband processing. The latter is mainly limited by complexity, which makes an evolutionary path that gradually exploits future performance increases of CMOS based digital circuitry, possible. This is significantly different to the situation of RF front-ends, which are primarily bounded by physical limitations of RF devices. Those generally known RF degradations encompass device nonlinearities, dynamic range limitations, signal degradations due to device mismatches and the like. A possible way to overcome these analog impairments is based on digital signal processing functions locally implemented in the RF transceiver front-end. One example of such a digital based analog impairment correction could be a programmable FIR filter for the compensation of in-channel amplitude/phase ripples, which are mainly caused by of the analog channel selection filter. This approach will rely of course on the availability of an advanced CMOS process options for the semiconductor process under consideration.

To further maximize the flexibility of the foreseen RF concept, a digital control and data interface should be implemented between the front-end and the digital base band. Since excessive data rates are expected at the output/input of the ADCs/DACs, a digital front-end is needed to decimate/interpolate the signals to moderate (low multiples of the symbol/chip rate) levels. This will also necessitate the development of fractional sample rate converters, to connect the sample rate of the different standards (e.g. the UMTS chip rate of 3.84 Mcps) with a physical clock rate that is common for all standards.
For implementation of re-configurable RF front ends architectures with a high degree of flexibility are preferred. Thus, the primary architectures of choice are the direct-down conversion receiver and the direct-up conversion transmitter because the amount of external, fixed frequency RF components is minimized and the external filtering at one or even multiple IF frequencies that is inherent to architectures with multiple frequency conversions can be avoided.

C. Digital Baseband Hardware Architectures

In order to derive abstract models for reconfigurable hardware resources, different levels of abstraction with varying degree of granularity (e.g. Instruction Set Level, Register-Transfer Level, Gate Level, Circuit Level) have to be considered. Each model can be described in the behavioral, structural or physical domain. The design of reconfigurable hardware architectures has to take into account flexibility (i.e. software programmable), high computational performance (i.e. high data rate, low latency), as well as low power consumption for handheld devices. Figure 4 shows a reconfigurable digital baseband processing architecture controlled by the CCM.

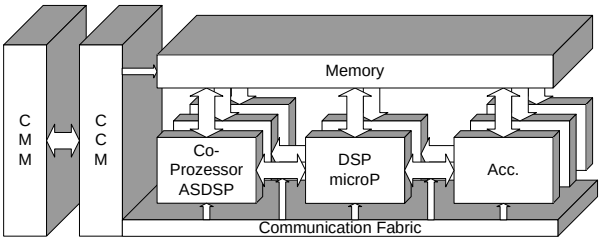

Figure 4: Reconfigurable Baseband HW Architecture.

C.1 Application Specific DSPs

Algorithm specific hardware accelerators can be used to enhance the baseband digital signal processing capabilities. But they offer only a limited degree of reconfigurability. Fully programmable DSPs while providing a high degree of flexibility often do not match the required signal processing requirements. Software programmable and thus reconfigurable application specific DSPs (ASDSPs) have a far better signal processing performance and power consumption than general purpose DSPs. They fill the gap between application specific HW accelerators and general purpose DSPs. The following design paradigms will be considered for developing ASDSPs which can be configured by the CCM:

Application Classes: A generalization of base-band processing algorithms for different air interfaces will lead to a set of basic algorithm classes for which a common software platform as well as a common hardware platform can be designed.
Parameterizable Basic Architectures (PBA): These basic architectures enable a fast adaptation of the baseband processing platform to a given application. One PBA must be found for each algorithm class. Among those are the signal processing machine (SPM) and the control processing machine (CPM).
Parameterizable Instruction Set Architecture (ISA): The main characteristic of this interface between programmer and hardware is to map the scalability of the hardware platform to the command complexity.
Parameterizable Design and Programming Tools: These are tools like assemblers, debuggers, simulators etc. Similar to the ISA, the tools must be easily adaptable to the different architecture alternatives.

C.2 Algorithm Specific Accelerators

A hardware accelerator is a block which is added into the system architecture in order to speed up signal processing and add new functionality to the system. Co-Processors and to some extend ASDSPs are tightly coupled to the μP/DSP. They provide an extension to the instruction set and potentially affect the DSP/μP pipeline. Autonomous accelerators are only loosely coupled to the DSP/μP. They provide a separate instruction set and operate independent from the μP/DSP. Algorithm specific accelerators are considered for processing of an algorithm/application, where only a limited set of more or less complex operations is needed. In contrast to a DSP, this kind of accelerator is therefore optimized for a certain class of algorithm, implementing a limited instruction set. Similarities and differences between the certain standards/functions have to be identified and parameterized. The basic algorithms, underlying the systems, are evaluated and classes of algorithms with their corresponding parameter sets will be identified. They will be the basis for the accelerator design. Re-configuration capabilities will be provided only to a limited extent. The accelerator concept allows a modular and scalable system architecture with independent, decentralized processing components.

V. SCENARIOS

The roles of the entities involved in the reconfiguration process of the physical layer, as well as their interactions, are illustrated here through two examples. The first one relies on a basic reconfiguration scenario relating to intra-standard performance enhancement, which is considered as one of the first achievable case-studies in the E2R

roadmap. The second one tackles one of the most challenging goals of the project: reconfiguration for inter-standard handover.

A. Intra-standard performance enhancement

This scenario only involves the user equipment (UE) and, to a certain extent, part of the network. It consists in downloading a new piece of code (or patch) to either replace a bugged algorithm of the modem or improve its performance [4]. The reconfiguration process can be divided into the following consecutive steps:
1. A dialog must take place between the UE and a remote network reconfiguration manager (NRM) so that the UE can be informed that a new patch is available for download.
2. The NRM then selects the appropriate patch and sends it to the UE as a classic payload through a normal communication standard that supports reliable data transfers.
3. The UE recognizes the payload as configuration data and installs it in its local configuration database; it is now ready to use the updated version of the code.
4. Even though the patch is downloaded while on-line, no specific real-time constraints need to be met and actual reconfiguring may happen off-line.
The CCM must support the following operations to enable such a reconfiguration scenario: It aware of its own hardware and software constituents and keeps track of the various algorithm versions that it possesses in configuration memory. It provides the NRM with all this information in order to determine whether updates are available and which patch should be sent. It monitors the downloading of the patch in a secure and reliable manner.
It installs the patch into its configuration memory. The new version of the algorithm is now ready for use, and the previous version is kept for rollback capability.

B. On-line mode switch for inter-standard handover

This end-to-end reconfiguration scenario involves the whole communication chain, but is addressed here from the equipment reconfiguration point of view: Passing seamlessly from one standard to another at run-time, the new standard being instantiated by reconfiguring the modem and the upper layers while the former standard is still running. In this case, it is assumed that the UE already has all the required configuration data (no need to download it first). The reconfiguration process can be divided into the following consecutive steps:
1. A negotiation takes place between the NRM, which takes the decision to perform a handover, and the UE to determine if the reconfiguration is achievable, and under which conditions.
2. The UE is reconfigured to support the new standard (all layers are concerned). This must happen while the former standard is still running and without interfering with it. Particularly, the reconfiguration process must not in any case prevent the real-time constraints of the former standard to be met.
3. A new communication link is opened based on the new standard. The two links must co-exist for a certain time, long enough for the new standard to be ready to take over the communication. This phase requires a tight synchronization with the network.
4. Once the handover has been done, the former link is shut down without interfering with the new one.
Here, the CCM must support the following operations to enable such a reconfiguration scenario: Knowing which resources are allocated to the former standard and which are available for instantiating the new one, it is able to accept or decline the reconfiguration requested by the NRM. After dynamically allocating the required resources, it will instantiate the new standard by reconfiguring the modem. This must be done without interfering with the former standard. After the handover has been done, the CCM frees the resources that were allocated to the former standard and makes them available for further reuse.

V I. CONCLUSIONS

Functional elements and their interactions in an end-to-end reconfigurable equipment have been addressed at different levels of abstraction. Particular emphasis has been placed on the configuration control and hardware resources. Many open issues have been identified which will be subject to further investigation.

ACKNOWLEDGEMENTS

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

REFERENCES

[1] End-to-End Reconfigurability (E²R), IST-2003-507995 E²R, http://www.e2r.motlabs.com.
[2] 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.
[3] J. Brakensiek, M. Darianian, S. Walter, C. Dolwin, M. Halimic, L. Maurer, A. Kountaris: Reconfigurable Physical Layer Architecture supporting End to End Reconfiguration, WWRF Meeting New York, October 2003.
[4] C. Moy, A. Kountouris, A. Bisiaux, HW and SW Architectures for Over-The-Air Dynamic Reconfiguration by Software Download, SDR Workshop of the IEEE Radio and Wireless Conference, Boston, USA, Aug. 2003.