THE ROLE OF THE CONFIGURATION CONTROL MODULE IN AN END TO END RECONFIGURABLE SYSTEM

Craig Dolwin (Toshiba Research Europe Ltd, Bristol, UK; craig.dolwin@toshiba-trel.com).
Stefan Mende (Nokia Research Center, Bochum, Germany; stefan.mende@nokia.com)
Jörg Brakensiek (Nokia Research Center, Bochum, Germany; jorg.brakensiek@nokia.com)

ABSTRACT

This paper discusses the role of the Configuration Control Module (CCM) when configuring and controlling the physical resources in a wireless platform. We focus on the interfaces between the CCM and higher layer entities in the configuration plane as well as the interface to the signal processing and communication resources it controls.
We describe the role of the CCM as part of an end to end reconfiguration of a wireless connection.
The CCM is responsible for the interpretation of platform independent configuration requests into the detailed programming and scheduling of individual signal processing elements and communication resources. The interfaces supported by the CCM allow multiple levels of configuration granularity as well as supporting proprietary adaptation algorithms and hardware solutions. The hardware resources the CCM control include the baseband signal processing stages, source coding, analogue front end and RF components.

1. INTRODUCTION

In an end to end reconfigurable system, as envisaged in the EU Project, E^{2}R [1], the protocol stack, including the physical layer and application layer, at either end of a wireless link can be configured to implement multiple Radio Access Technologies (RAT) as required to meet the overall objective of a re-configurable system.
It is anticipated, at a system level, the configuration algorithm will want to modify basestations and their associated terminals to match the prevailing operating environment. Typically this might be to address congestion on the system or to implement improved proprietary modem technology.
The high level network entities in such a system have to assess the capabilities of a wide range of equipment and decide on what the best configuration is for the E^{2}R system as a whole. To support this decision equipment must be able to define its capabilities in a common format. This will be made especially difficult at the terminal where a range of hardware architectures will be used. Naturally the E^{2}R system should be designed to operate well into the future and so will have to support multiple generations of equipment as well as different manufacturers.
Once a decision has been made about the configuration of all the local basestations and associated terminals a command must be sent to each component instructing it to implement a new configuration. The transition to this new configuration must be sequenced across the network so the user is not aware of any loss of service. Due to the potentially massive number of different hardware platforms a central configuration entity cannot be expected to send implementation specific configuration instructions. So the Configuration Control Module (CCM) is responsible for interpreting a platform independent command into a precise set of configurations and schedules for each resource in the physical layer.
In addition to the system level configuration described above it is also anticipated that the equipment might reconfigure itself. This might occur when a Radio Resource Controller (RRC) in the protocol stack decides to change the mode of operation e.g. in WCDMA the RRC may alter the number and type of transport channel or simply change the channel frequency. Alternatively an optional Adaptation Control Unit (ACU) might decide, based on channel quality metrics, that the level of receive processing can be altered to ensure the power consumption is minimised while still maintaining the target QoS. In both these cases the change to the physical layer can be implemented either through a direct interface to the physical layer or via the CCM. If the interface is via the CCM the data processing resources are scheduled and the communication fabric configured to implement a more power efficient solution.

2. BACKGROUND

In the present day terminal the range of configurability is limited to those operating modes known at design time. The system designer will analyse the overall requirements and determine critical paths in the data flow for each mode of operation. The designer will then define a set of hardware resources (e.g. DSP, CPU, RAM, Hardware accelerators, RF blocks and data busses) and associated operating parameters such as clock frequency and power supply voltage that will *just* support the highest complexity mode. These decisions will based on a desire to minimize cost, power consumption and development time. Currently the constraints of power consumption and cost mean that the physical layer will be implemented using ROM to store DSP code and ASIC's for implementing logic functions. The only flexibility allowed in the physical layer is a small area in RAM that is allocated for patching bugs.

Once the designer has specified the resources they must map functional blocks to specific hardware resources and determine how and when each resource will execute the function [4].

In a reconfigurable system the hardware is also defined at design time but it is envisaged that some of the elements may be programmable e.g. reconfigurable logic or RAM based micro-processor. In an attempt to reduce the possibility of bottlenecks between processing elements the communication fabric (data bus, dual port memory etc) may also be configurable.

The role of the CCM is to take a configuration request and, like the designer working on a traditional design, determine the best mapping of functions to resources and then schedule when resources execute these functions.

3. MOTIVATION

Objectives

Encapsulation

The CCM must hide details of the underlying hardware from higher layer entities such as the Configuration Management Module (CMM). This encapsulation allows these entities to define the optimal configuration across the system without having to have a detailed knowledge about each and every piece of equipment in the system.

Power Consumption

One of the main obstacles to the successful introduction of a reconfigurable platform into the commercial market is its tendency to increase the power consumption in the terminal. While it can be anticipated that process technology will improve the mW/MIPS figure [2][3] and this maybe argued as enabling Software Defined Radio, we should also expect the demand from the user for higher data rates, higher system capacity and extra services to match any possible technology improvements [10]. Given this scenario we cannot allow the addition of a reconfigurable solution to significantly increase the power consumption from its non-configurable but functionally similar predecessor.

Constraints

Real-time Operation

Many of the applications in a wireless environment will impose hard timing deadlines e.g. echo delay in a voice connection. In addition the Radio Access Technology (RAT) itself will impose control deadlines to ensure the dynamic nature of the channel is tracked.

Secure and reliable operation

In traditional equipment all modes of operation will be verified at design time against a reference system, the standard specifications and marketing requirements. In a reconfigurable system with a high level of flexibility it becomes infeasible to verify each and every possible configuration.

In practice only individual components will be verified. The CCM and associated framework must ensure that these components interact in a predictable and deterministic fashion to ensure the correct operation of the complete system.

4. DESCRIPTION

The Configuration Control Module (CCM) works with the Configuration Management Module (CMM) to facilitate an agreed configuration in hardware. In general the CMM addresses issues of functionality while the CCM is focused on implementation [8].

The Configuration Management Module (CMM) is responsible for the following:

1. Monitoring and discovering the capabilities and status of the network in the vicinity of the equipment.
2. Negotiating and selecting the appropriate configuration.
3. Supporting general procedures, namely, software and protocol download and installation.
4. The co-ordination of equipment to realise a new system configuration.

In a practical implementation of a terminal the hardware used to implement a reconfigurable platform will be shared across multiple functional blocks so the CCM acts as central controller. The CCM uses spatial and temporal scheduling algorithms to determine which hardware resource implements which function and how or when a function gets to access to a resource.

In this document we focus on the radio modem aspects but the principle could be applied to other layers in the protocol stack especially where resources are shared between functions and hard real time constraints are required. Figure 1 graphically describes the relationship between hardware resources, the CMM and the CCM.

As mentioned earlier the major role of the CCM is to supply an abstract configuration interface that allows higher layer entities like the CMM to configure resources without having a detailed understanding of the underlying implementation. This interface is realized with a ServiceAPI, which implements a message-passing interface [6]

Primary Configuration Control Interface

This is the primary interface between the CMM and the CCM which sets-up a new configuration; it translates the configuration in to a set of configurations for the different layers (application, protocol and physical layer).

The configuration requests could be at the level of a complete Radio Access Technology but could also be at a higher granularity. In a high granularity description sub components of the radio modem or protocol stack would be defined as well as the interconnection between them. Typically the functional components could be channel or speech codec's. While it is recognized that supporting high granularity configuration descriptions will add significantly to the workload of the CCM we aim to create a framework which will support this approach in the future.

Secondary Configuration Control Interface

The secondary interface between the CMM and the CCM supplies a set of functions unique to the selected RAT. These functions might initiate a change request from the RRC/Radio Link Controller (RLC) in layer 3 of a WCDMA protocol stack and would, for example, request a change in the number of transport channels. Precisely how this functionality is changed is implementation dependant but could involve actually changing the system configuration (i.e. changing allocation of resources) or by modification of parameters within the system.

Capability Interface

The Capability Interface allows the CMM to determine what configurations the underlying layer is able to support. In simple cases this can be defined by the equipment manufacturer using Class Marks. Typically a Class Mark might indicate that the equipment can support a specific mode in WCDMA and it would define the maximum number of physical channels, transport channels and associated data rates. The Class Mark approach is limited, in a reconfigurable platform, because it does not take into account how functions within a RAT are implemented. So if a component, such as a turbo decoder, was replaced to fix a bug or replaced by a lower power version this may alter the capability of the platform e.g. alter the maximum data rate. In addition the number of operating scenarios in a reconfigurable platform can be very large e.g. the equipment might be expected to support multiple types of primary RAT while also monitoring a range of secondary RAT's and support multiple speech codecs. Because all these functions will share common resources it is quite likely that not all combinations will be supported.

From this initial analysis it seems clear that further work needs to be done to dynamically define the capabilities of a reconfigurable platform.

Database Interface

The Database Interface provides access to databases in the different layers. The database will contain layer specific configurations and their properties. It may also contain metrics that can be used by the configuration plane to determine future configurations.

Auxiliary Service Gateway Interface

The Auxiliary Service gateway allows an optional customized and proprietary interface to the CCM to be created. Typically this might be used for adaptation or cross-layer optimisation.

Configuration Language

For specification and setting of configuration specific parameters for a given RAT a generic and extensible interface description has to be used. An XML based configuration description language is being investigated [7]. A simplified XML parser is accessible via the message-parsing interface (ServiceAPI) and has the intelligence to understand the XML description and translate it to the right function call.

Interface with Hardware Resources

To achieve low power consumption within a reconfigurable platform the CCM configures at a signal processing block level. A signal processing block refers to resource configured to implement a specific function. Each block is optimised for low power consumption and could typically be a key functional building block in a wireless system e.g. rake receiver, turbo decoder, FFT etc [8]. The CCM must then link these building blocks to create the requested system. While it is recognised that the most power efficient implementation for such a building block is an ASIC [4][5] we also allow these building block to be reconfigurable components in themselves e.g. FPGA, reconfigurable logic or a task on a DSP. This adds another dimension to the control problem but significantly enhances the flexibility of our system.

Two layers of abstraction are used when configuring hardware resources see Figure 2 [12]. At the lowest level, the *Generic Physical Layer Interface,* we use device drivers, supplied by the equipment manufacturer. The driver is responsible for installing new configurations into hardware and creating communication channels to stream data between processing elements. To do this it must have detailed knowledge about control registers and the address space. Taking an FPGA implementation of a turbo decoder as an example the creation of a communication channel to stream data into the turbo decoder might involve reserving a DMA channel to transfer data from a buffer in global memory into a FIFO implemented on the FPGA. The implementation of the turbo decoder itself would require the bit stream to be loaded into the FPGA memory. Typically device drivers could be embedded in ROM or alternatively could be downloaded into RAM.

The next level of abstraction is at the *Generic Resource Interface* and contains resource driver's specific to a RAT. A resource driver allows functional element to be configured without the higher layer having to know about the implementation detail. This allows the implementation resource to change without affecting the high layer configuration entities e.g. the turbo decoder might need to be configured to use a specific set of polynomials. This would be requested by calling the *setPolynomial* method in the TurboDecoderInterface and because this had previously been associated with the FPGA resource it would be realized by the method in *TurboDecoder_Xilinx_FPGA_Driver.* In figure 2 we use a UML diagram to identify the relationship between the generic resource interface and the generic physical interface.

5. SCENARIO

End-to-End reconfigurability with respect to the physical layer will imply a set of different scenarios, which are of interest and importance. Among others, these are basically [11]

- ❑ Adaptation of the signal processing chain to environmental changes in order to provide a constant QoS for the connection.
- ❑ Inter RAT Handover (e.g. UMTS to WLAN)
- ❑ Intra RAT Handover (e.g. UMTS/FDD to UMTS/TDD)
- ❑ Service Addition (e.g. adaptive multi-homing)
- ❑ Service Enhancement (this is mainly on the application layer, e.g. replacement of a video codec)
- ❑ Bug Fixing and functionality enhancement of the implemented signal processing chain

Reconfiguration can be initiated either from the network/operator side or from the user/application side. Even if there are slight differences during the reconfiguration setup, the basic reconfiguration process is independent of the initiator.

The reconfiguration process is described in the following, taking a network initiated inter-RAT handover from WLAN to WCDMA as an example.

1. A network management entity is questioning the terminal on its capability to operate WLAN signal processing. This is done using the capability interface of the service API.
2. The terminal will look into its local database for information on reconfiguration scripts for the WLAN processing.
3. The terminal will provide feedback on its WCDMA capability e.g. in a kind of yes/no answering scheme.
4. The management entity may ask for additional detailed capabilities, e.g. supported modes, frequencies, code rates, number of transport channels, etc. This could be done in a kind of interview mode.
5. If the required functionality is not available inside the terminal, a software upgrade is necessary. The software download from a software database (e.g. at the manufacturer side) will be done using the database interface of the service API. Available software versions and library functions will be exchanged during this process.
6. As all required configuration data is available, the management entity will initiate the reconfiguration using the primary configuration interface of the service API. In order to synchronize the reconfiguration within the different layers and also together with the network, a time stamp could be given.
7. Additional fine granular configurations to standard specific options will be done using the secondary configuration interface of the service API. Typically the RRC/RLC, in the control plane of the protocol stack, will be connected to this interface.

Evaluating the capabilities on the terminal side is a complex process, as the current system configuration has to be taken into account (e.g. other services/applications, which are currently running). Therefore a detailed analysis of the load and usage of the different hardware resources is required, taking the additional processing and communication load of the new target system into account. This includes the feedback from the spatial and temporal scheduler as well as potential analysis of mapping alternatives. Besides this dynamic feedback analysis, a static one may be available as a first stage, which is based on predefined database entries (e.g. as defined from the manufacturer). The static analysis will therefore always be conservative in providing capability feedback.

The configuration of the hardware resources is done in a generic way, dependent on their capabilities.

- ❑ Resources having their own operating system or firmware will be responsible for storing and operating the new configuration.
- ❑ Other resources will be configured from the dedicated resource drivers. This may include e.g. download of a bit stream to an FPGA, store register values.

Specific attention has to be given to the synchronization of the reconfiguration start, especially if the resource is shared between different applications. This is to ensure that any reconfiguration does not effect the ongoing processing.

6. CONCLUSION

This paper has outlined the work currently ongoing in WP4 of the European Integrated Project, E^{2}R. We have described how the Configuration Control Module (CCM) works within the configuration plane to manage the underlying hardware resources. The focus of this work is to develop a scalable control framework that does not constrain the equipment manufacturer to a given hardware architecture but still supports low power operation and high levels of configurability. This is achieved by using a set of optimised processing blocks which can be dynamically connected and configured to implement the requested overall configuration.

A number of issues still need to be resolved and these include:

- How best to define the capabilities of a reconfigurable platform.
- What level of granularity should the configuration description be at?
- How do we validate a configuration?

7. ACKNOWLEDGMENT

This work is being performed within the framework of EU funded project E^{2}R. The authors would like to acknowledge the contributions of their colleagues from the E^{2}R consortium

8. REFERENCES

[1] E^{2}R home page: http://www.e2r.motlabs.com/e2r

[2] Gene Frantz, "Digital Signal Processing Trends", *IEEE Micro*, vol. 20, no. 6, pp. 52-59. Nov/Dec 2000

[3] Ralf E. Schuh, Peter Eneroth and Peter Karlsson. "Multi-Standard Mobile Terminals", IST Mobile & Wireless Telecom. Summit'02, pp 174-178, June 2002

[4] H.Blume, H. Hubert, H. T. Feldamper, T. G. Noll, "Model-based Exploration of the Design Space for Heterogeneous Systems on Chip", ASAP'02, pp29, 17th-19th July 2002.

[5] H. T. Feldkamper, T. Gemmeke, H. Blume, T. G. Noll, "Analysis of reconfigurable and heterogeneous architectures in the communication domain", ICCSC2002, St Petersburg, Russia, 28th June 2002

[6] Home page for Message Passing Interface standard, http://www-unix.mcs.anl.gov/mpi/

[7] A basic description of XML, *http://nyphp.org/content/presentations/pvsxml/xml-sample.php*

[8] T. Farnham, C. Dolwin, S. Zhong, R. Atukula, U. Lücking, S. Mende, J. Brakensiek, S. Buljore, N. Alonistioti, F. Foukalas, A. Glentis, P. Magdalinos, P. Demestichas, V. Stavroulaki, "E2R Equipment Management and Control", Proc. of the E2R workshop on Reconfigurable mobile systems and networks beyond 3G, Barcelona, 5th September 2004.

[9] Manabu Mukai et al, "A Software Oriented Modem Architecture for 3G Terminal", IEEE VTC 2003 Fall.

[10] Jan M. Rabaey, "System on chip : A case for heterogeneous architectures", http://bwrc.eecs.berkeley.edu/People/Faculty/jan/presentations/architecture.pdf

[11] J. Brakensiek, B. Steinke, S. Walter, T. Burger, T. Dellsperger, C. Dolwin, R. Burgess, A. Bisiaux, M. Bronzel, H. Seidel, M. Halimic, "Requirement and Scenario Definition", Public E2R Deliverable IST-2003-507995/ E2R/WP4/D4.1/040630

[12] J. Brakensiek, D. Lenz, B. Steinke, M. Halimic, C. Dolwin, S. Naveen, A. Bisiaux, "Hardware Abstraction in an End-to-End Reconfigurable Device", Proc. of WWRF WG6 Meeting, June 2004, Oslo.

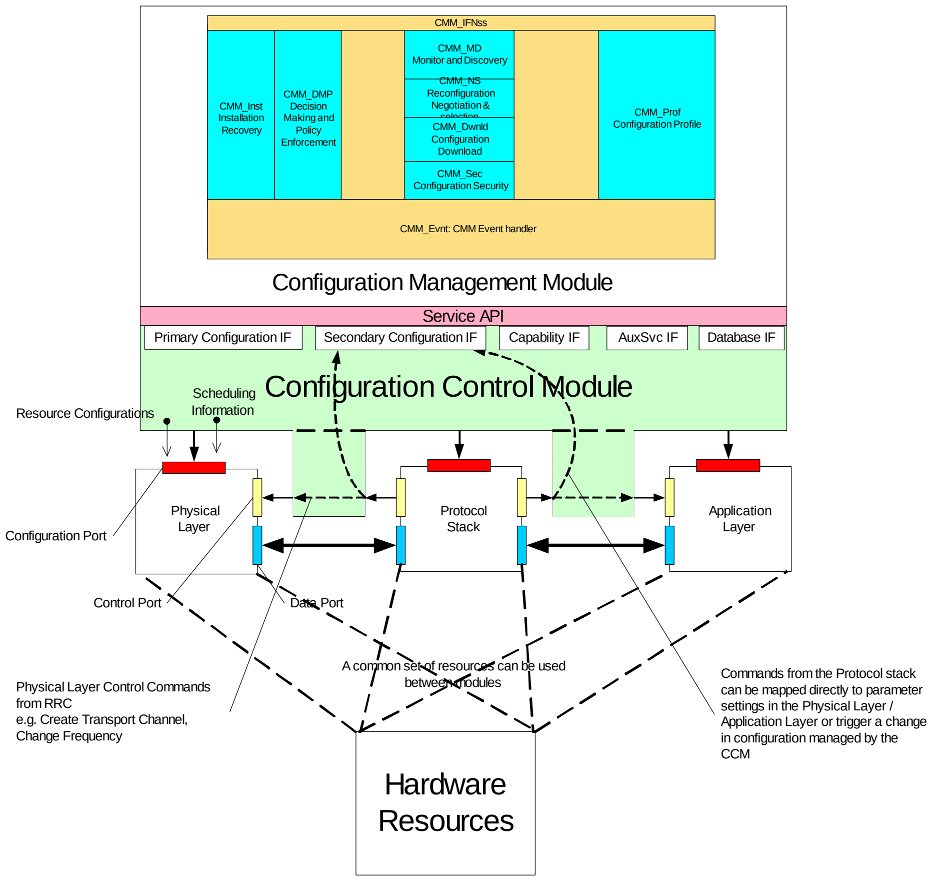
CMM_IFNss
CMM_Inst Installation Recovery
CMM_DMP Decision Making and Policy Enforcement
CMM_MD Monitor and Discovery
CMM_NS Reconfiguration Negotiation & selection
CMM_Dwnld Configuration Download
CMM_Sec Configuration Security
CMM_Prof Configuration Profile
CMM_Evnt: CMM Event handler
Configuration Management Module
Service API
Primary Configuration IF
Secondary Configuration IF
Capability IF
AuxSvc IF
Database IF
Configuration Control Module
Resource Configurations
Scheduling Information
Physical Layer
Protocol Stack
Application Layer
Configuration Port
Control Port
Data Port
A common set of resources can be used between modules
Physical Layer Control Commands from RRC e.g. Create Transport Channel, Change Frequency
Commands from the Protocol stack can be mapped directly to parameter settings in the Physical Layer / Application Layer or trigger a change in configuration managed by the CCM
Hardware Resources

Figure 1 Relationship between CMM, CCM and resources

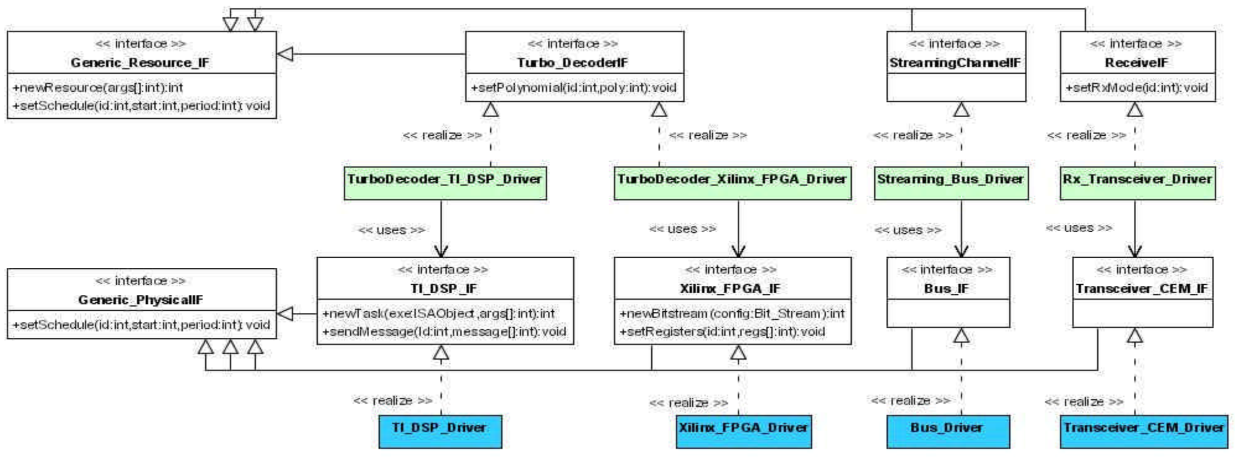

Figure 2 Class diagram illustrating hardware abstraction layers