POWER EFFICIENT AND REAL-TIME CONFIGURATION OF RESOURCES IN AN END-TO-END RECONFIGURABLE SYSTEM

Craig Dolwin (Toshiba Research Europe Ltd, Bristol, UK, craig.dolwin@toshiba-trel.com);
Rollo Burgess (Toshiba Research Europe Ltd, Bristol, UK, rollo.burgess@toshiba-trel.com);
Bernd Steinke (Nokia Research Center, Bochum, Germany, bernd.steinke@nokia.com)

ABSTRACT

This paper discusses the mechanism developed in E^2R [1] to reconfigure hardware and software to implement a low power wireless system while maintaining the robust and reliable operation expected from telecommunication equipment.

From the perspective of reducing power consumption the key signal processing elements (e.g. FFT, rake fingers) should be implemented in the most power efficient approach possible. Ideally this would mean it is implemented in a resource such as an ASIC or hardware accelerator [2] but also allowing solutions in reconfigurable logic or DSP's.

The algorithms and mechanisms that enable this are enclosed in a module called the Configuration Control Module (CCM). The input to the CCM is a Functional Description Language (FDL) written in XML (eXtensible Markup Language) syntax. From this functional description the CCM will be able to generate a detailed configuration for the underlying hardware. We will present a basic XML schema to define the Functional Description Language and show how it can be interpreted for a specific hardware platform. Finally to demonstrate the ideas discussed we will present a scenario for the handover between WCDMA and WLAN.

1. INTRODUCTION

The design for a commercial wireless terminal is severely constrained by size, cost and the ever increasing demands for increased functionality. Very few if any designs will start from a blank sheet of paper so many of the key modules will be legacy components or supplied by 3rd parties. The role of the lead design team will be to develop the control code that will integrate each component. As the number of wireless standards increase and the number of modes of operation within each standard also increase the task of integration and verification becomes more and more complex and lengthy. The traditional approach to developing a wireless modem requires the design to be hard coded into ROM and ASIC with a small amount of RAM allocated for patching bugs in a DSP or RISC processor. Typically this processor contains two types of code:

1. Control code
2. Data processing code

The control code is substantially more complicated and therefore takes longer to test and debug than the data processing code. As product lifetimes decrease and complexity increases the development of control code that will support all envisaged RAT and associated modes becomes unsustainable. A reconfigurable solution addresses these issues by:

1. Simplifying the control code to a single RAT per configuration.
2. Freezing the hardware design across a range of products.
3. Allowing the fast development of new standards and standards not known at design time.

2. OVERVIEW

In this paper we describe a framework that supports the dynamic configuration of the signal processing components and their associated communication paths. A signal processing component can be a highly optimized fixed function logic block (e.g. a hardware accelerator) or can itself be implemented on a reconfigurable resource such as a RAM based DSP or reconfigurable logic. We show how a platform independent language, based on XML, is used to communicate a new functional description to each piece of equipment irrespective of model or manufacturer. The information supplied in the XML file allows the CCM to use a library of signal processing modules (supplied by the equipment manufacturer) to compile a set of binary images specific to the platform and then using a set of constraints and timing information, which are part of the RAT-standard and the processing module, it creates a schedule that will ensure the correct real time operation.

This E^2R framework differentiates it self from other schemes by consigning aspects of a RAT implementation which effect the power consumption and resource usage to the equipment manufacturer. This allows the manufacturer

to decide how best to optimise the design between low-power consumption and minimal resource usage. Some examples could be:

1. Using an array of hardware accelerators rather than reconfigurable logic,
2. Multiple signal processing elements rather a single DSP with a complicated scheduler,
3. Simple low overhead communication channels rather than a CORBA based approach

3. ARCHITECTURE

In the proposed configurable system there are three types of modules:

- Configuration Management Module (CMM). The CMM is responsible for deciding on the correct functional configuration for the equipment.
- Configuration Control Module (CCM). The CCM is responsible for interpreting configuration requests from the CMM into actual implementations on the hardware.
- Configurable Execution Module (CEM). A CEM is a configurable hardware resource. One or more CEMs are configured by the CCM

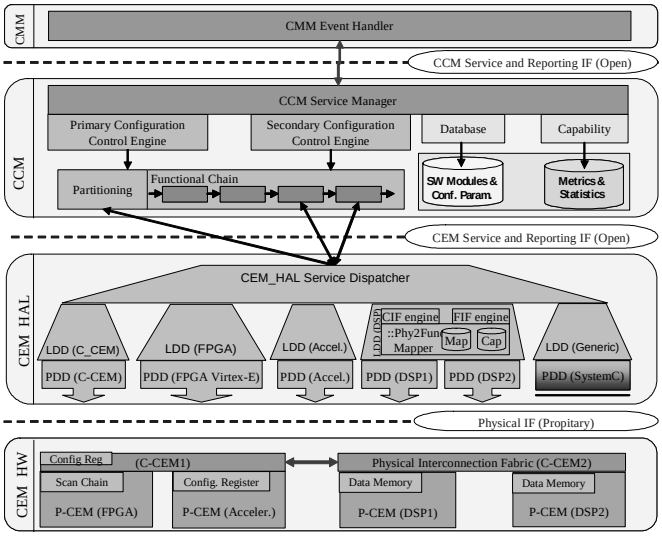


Fig 1. Overview of configuration plane in an E2R terminal

Fig 1 shows the relationship between the different modules [19]. The CEMs are the lowest (physical) layer and represent the signal processing CEM (P-CEM) and Communication CEM (C-CEM) hardware. The CCM is the next higher layer. Its task is to trigger reconfigurations and provide data derived from any kind of database. In the E²R environment the CMM provides coarse control. This is elaborated in great depth in E²R D4.4 [18]. To enable the actual CEM hardware abstraction through the CEM_HAL, three sub layers are introduced in [20]: The CEM_HAL Service Dispatcher, the Logical Device Driver (LDD) and the Physical Device Driver (PDD).

In this paper we focus on the operation of the CCM. The interface between the CMM and CCM has been defined so it is independent of the underlying hardware. Information transferred between the CMM and CCM is contained in a data structure called a *Configuration Package*. A Configuration Package can contain, amongst other things, a *Functional Configuration* and one or more *Processing Modules*. The Functional Configuration is described using a Functional Description Language, further details are given in section 5. A Processing Module is a data structure describing a *binary* (i.e. object code, bit stream, parameters etc) which is used to configure a CEM. Fig 2 shows a UML class diagram describing the processingModule class and its relationship with other classes. In this diagram the P-CEM class defines the attributes of a processing resource in the equipment (e.g. DSP, reconfigurable logic etc). The binary class contains information about the data used to configure a CEM as well as a URL to identify the location of the code itself.

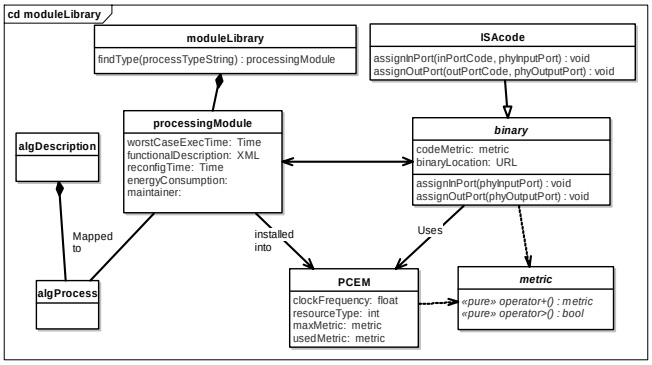


Fig 2 UML diagram for processingModule

4. SWITCHOVER SCENARIO

This scenario is used to show how it is envisaged that a reconfigurable terminal can be used to dynamically switch between two Radio Access Technologies (RATs) while maintaining a Voice over Internet Protocol (VoIP) telephone call [3]. The term *switchover* is used rather than handover because the two RAT are assumed to have no knowledge of each others existence i.e. only the configuration plane has this knowledge.

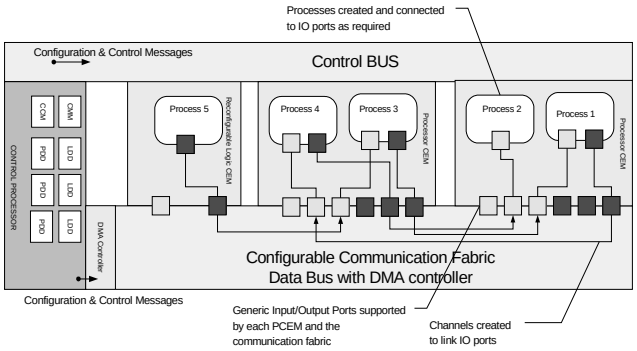


Fig 3. Terminal Hardware Architecture

To assist in the description the hardware architecture has been defined in Fig 3 and is composed of reconfigurable logic block, a DSP and a General Purpose Processor. These blocks are in addition to a control processor which contains the CCM and CMM. The communication between each P-CEM uses multiple DMA channels each with their own access priority to a common data bus.

The terminal has instantiated a WCDMA connection and is currently using one of WCDMA data services to set up a connection to the internet and is participating in a VOIP telephone call. The Configuration Management Module (CMM), within the configuration plane determines, that a WLAN hotspot maybe in range of the terminal and so decides to modify the functionality to include a WLAN transceiver as well as WCDMA. It has been assumed that two separate RF modules are available and each includes an ADC and DAC. After the WLAN stages have been instantiated a connection is made to the Internet and the VOIP application starts receiving IP packets via the WLAN connection. The configuration plane can now reconfigure the terminal to just implement a WLAN connection. It is important to note that for such a system to work the equipment must maintain a number of timing constraints e.g.

- The time delay between receiving speech data at the antenna and the associated acoustic output is constrained so the user does not experience degradation in quality from excessive echo[3].
- The WLAN standard 802.11a/b/g constrains the time taken to acknowledge a receive packet. This is known as the Short Inter-Frame Space (SIFS) [4].

The timing constraints make the implementation a *Hard realtime* problem and has significant impact on the design.

5. CONFIGURATION LANGUAGE

A configuration language is used within the proposed E^2R system to describe and communicate functional description data between different entities. The Meta language being proposed in this paper is XML (eXtensible Markup Language). XML has been proposed for similar usage in several earlier projects such as TRUST [5], SCOUT [6] and CAST [7] and is being used by the Joint Tactical Radio System's Software Communication Architecture [8]. A configuration language, not based on XML, but using a proprietary text based language called RDL (Radio Description Language) was defined by Vanu Inc to perform a similar purpose [9]. Further information about the configuration language and the choice of XML can be found here [10].

The Functional Description Language (FDL) in E^2R defines the functionality of the system in terms of one or more processes linked together by communication channels. An important aspect of this description is that it is independent of the target platform. So when defining a process it only references the functionality of signal-process element e.g. *wlan802.11gphy*. It is the role of the CCM to determine the precise processing module and associated binary image that can implement this function. A complete functional description is described as an *algorithm*, where an algorithm can encompass the RF, baseband processing, protocol stack and signal processing algorithms.

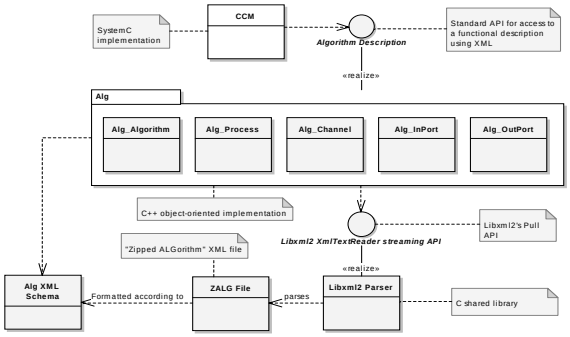


Fig 4. High level description of test parser implementation

In addition to defining the signal processing elements the FDL defines how a process communicates. This is done by defining a set of input and output ports per process and then linking them to a channel. FDL also supports hierarchy so process can contain sub-processes and the ports on these sub-processes can be attached to ports associated with the higher level process. Each channel has a set of generic metrics which allow the CCM to determine how many resources will be required to implement the channel. These metrics include latency and peak bandwidth.

FDL has been made as generic as possible so it does not constrain the granularity of a configuration description. In principle the description could be of a single module corresponding to a complete RAT or a set of interconnected sub-modules which make up a RAT. This flexibility is important because the former solution is the most likely in the short term as it allows completely validated modules to be installed. But in the longer term, as Software Defined Radio matures, the ability to combine modules which have

individually been validated but not validated as combined entity may become important.

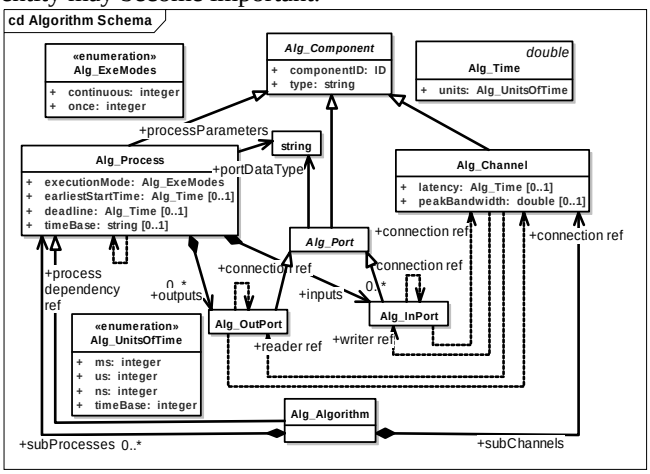


Fig 5. UML class diagram: Data model for the Functional Description Language

To define the configuration language a schema was specified in UML (Fig 5). The role of the schema is to enable validation of documents claiming to conform to FDL. A test parser was then written in C++ based on GNU libxml2 parser using a proprietary interface. For testing purpose the parser was developed in a Visual C++ environment but to understand the complexity issues associated with implementing an embedded XML parser we did a review of several lightweight parsers suitable for implementation in a terminal, see Table 1. The memory size is also affected by the size of the document being parsed; to minimize this effect a pull parser is preferred.

Name	Type	Lang	Size
Fusion RT [10]	push	C	15kb
TinyXML [11]	model	C++	12kb
XML Parser [12]	model/push	Java	6kb
kXML 2/3 [13]	pull	Java	9/?kb
Xparse-J [14]	model	Java	6kb

Table 1. Embedded XML parsers

Fig 6 contains snippets of the xml document used to describe the wlan and wcdma configuration shown in Fig 6. Because the original XML text file can be quite larger the file is compressed using the GNU utility gzip. The size of the compressed file for the combined WCDMA and WLAN configuration, shown in Fig 6, was 1.1Kbytes while the original text file was 8.6Kbytes. A compressed XML document is called a *ZALG* file. The parser will take ZALG files as its input. The relationship between parser, schema and .ZALG file is shown in Fig 8.

```
<?xml version="1.0" encoding="UTF-8"?>
<Alg_Algorithm xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="AlgorithmSchema.xsd" executionMode="once"
type="wlan802.11g_wcdmafdd384" componentID="wlanwcdma">
    <!-- Define timebase-->
    <timeBase>10MhzTimebase</timeBase>
    <!-- WCDMA Baseband Process -->
<subProcess executionMode="once" type="wcdmafdd384_phy"
componentID="wcdmaPhyProcess">
<processParameters> "wcdmafdd384_phy_mode_1_parameters"</processParameters>
                <input type="InputPort" componentID="wcdmaIQDataIn">
                    <portDataType>IQDataStream</portDataType>
                    <inputConnection ref="IQDataCh2" refType="Alg_Channel"/>
                </input>
                .
                .
                <output type="OutputPort" componentID="wcdmaSymbolOut">
                    <portDataType>SymbolDataStream</portDataType>
                    <outputConnection ref="SymbolCh2" refType="Alg_Channel"/>
                </output>
</subProcess>
  …..
<!-- WCDMA Channels -->
<subChannel type="Channel" componentID="IQDataCh2">
                <latency units="us">1.0</latency>
                <peakBandwidth>1e6</peakBandwidth>
                <writer ref="wcdmaIQDataOut" refType="Alg_OutPort"/>
                <reader ref="wcdmaIQDataIn" refType="Alg_InPort"/>
</subChannel>

</Alg_Algorithm>
```

Fig 6. Snippets of configuration description for WCDMA – WLAN Configuration.

6. DESCRIPTION

The following section describes how we envisage a terminal will implement the *switchover* scenario described in section 4. It is assumed that the terminal has previously been configured to use WCDMA and is shown in Fig 7, the transition configuration where both WLAN and WCDMA are supported is shown in Fig 8 and the final configuration running just WLAN is shown in Fig 9. The process of initiating a change in configuration is done by sending a *prepInstal* (Fig 10) message to the Configuration Control Module (CCM) via the CCM service interface. A parameter of this message is a ZALG file. In this scenario the ZALG file describes an update from the current WCDMA configuration to a combined WCDMA and WLAN configuration. The CCM will parse this file and create a data structure defining the requested functionality. The data structure contains objects defining process, channels and the relationship between each object. The role of the CCM is to interpret this data and determine how it can be implemented on the hardware. The precise method for doing this will be dependant on the equipment and its manufacturer. The method outlined in this paper describes a two stage approach. First mapping functional blocks to CEMs (known as *spatial scheduling*) and then defining when a resource should execute that function (*temporal scheduling*) this approach is similar to that taken by Wiangtong et al in [16].

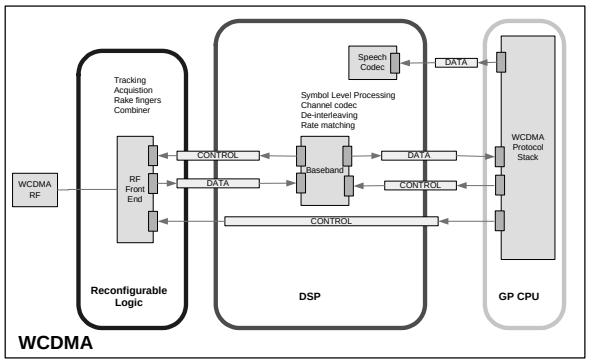


Fig 7. Mapping of WCDMA functions to CEMs

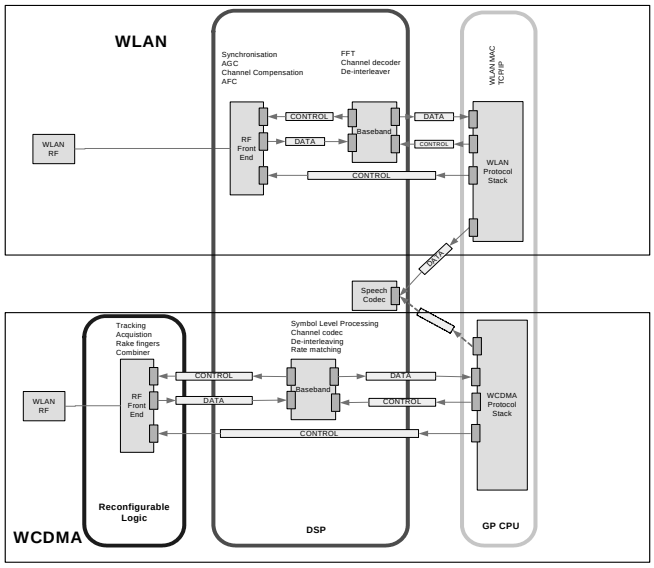


Fig 8. Mapping of WCDMA and WLAN blocks to CEM

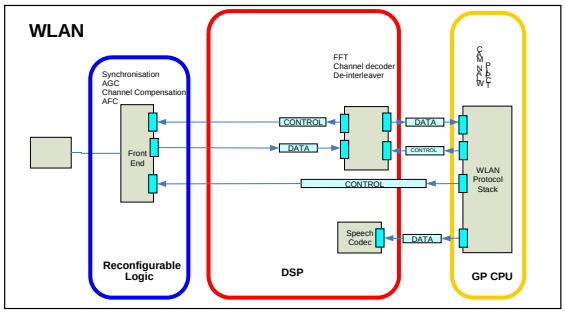


Fig 9. Mapping of WLAN blocks to CEM

The spatial scheduler looks through the manufacturer's library of functions, for this platform, to find a suitable match. As it is possible that more than one implementation is available the CCM must determine which is the most suitable given the overall configuration. Different implementations maybe located on different resources or may have different implementation characteristics. Obviously the complexity of the spatial scheduling will scale with number of possible permutations.

In this scenario the DSP and GPP are shared by multiple processes so the CCM must ensure that sufficient resources (i.e. clock cycles, program/data RAM etc) are available to support all the processes and that each process will maintain the timing deadlines set by the library module and the overall configuration description. This calculation is done by the *temporal scheduler*. The temporal scheduler determines how a CEM such as the DSP shares its resources between processes. A large number of algorithms exist for determining whether a set of tasks can be scheduled on a resource and what the schedule should be. A summary of some of these algorithms for realtime systems can be found in [17].

As can be seen from the hardware architecture in Fig 3 the communication channels all share the same C-CEM (i.e. the databus) and so the temporal scheduler must ensure that sufficient bandwidth is available to support the complete configuration. This calculation is done based on the peak bandwidth and latency requirements provided in the configuration description and judged against the physical limit description defined for each C-CEM.

After the CCM has determined that the new configuration can be supported it requests that any objects not stored locally to the CCM are downloaded. In this scenario the WLAN front end for the DSP is downloaded. This Object can be stored in a database local to the terminal or remotely at a URL address. This is done using the *prepInstallRes* message. Once all object codes are stored in the CCM the *install* message is sent. This initiates the downloading of CEM specific object code into the CEMs. After the object code is downloaded the newly installed code can be started. This is done using the *execute* message. A further, optional, stage is then gone through where the precise sequence in which data is passed through the processing stages is controlled. This is required to ensure a seamless switch between one configuration and the next; it is also required to avoid possible deadlock situations.

In this example the initial WCDMA configuration is distributed across the reconfigurable logic, a DSP and a General Purpose Processor (GPP), Fig 7. The combined WLAN and WCDMA configuration is implemented by adding further processes to the DSP and GPP. The extra communication channels are implemented by allocating DMA channels to each port. Fig 8 shows how the spatial mapping algorithm could map the RF front end processing for the WLAN into the DSP rather than the reconfigurable logic block. This could result in higher power consumption but would not require the reconfigurable logic block to support partial re-configuration. This is an example of where the manufacturer can optimise their design accordingly. Once the WLAN connection is made and the voice codec is being successfully fed with voice packets the

equipment can be reconfigured to a low power consumption configuration by removing the WCDMA processes and moving the WLAN RF front end into the reconfigurable logic. To ensure that throughout this transition the user and therefore the VOIP application are unaware of the underlying changes the interface to the CCM supports a multiple stage configuration procedure. i.e. prepInstall → transferObject →install →execute →switchConfig. A detailed sequence diagram showing the transfer of messages between the CCM and CMM is shown in Fig 10.

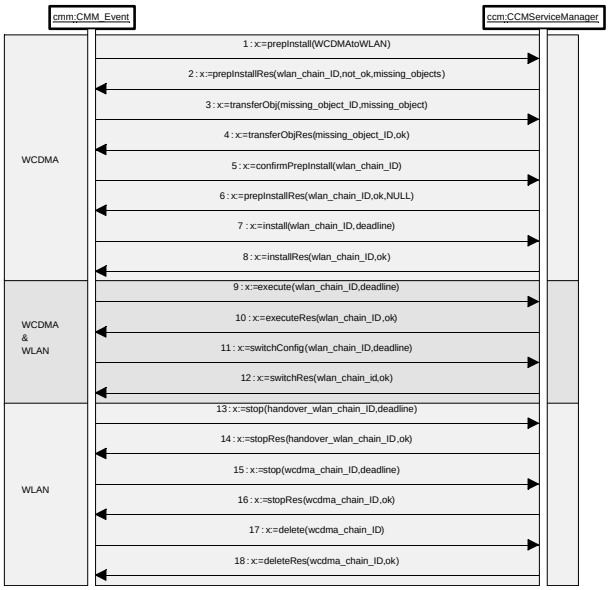


Fig 10. Sequence diagram for the switchover scenario

7. CONCLUSION

This paper has outlined a framework that will enable a commercially viable implementation of Software Defined Radio within an End to End Reconfigurable system. The key requirements for a commercial solution have been identified as low power consumption and optimum resource usage. We have addressed the need for low power consumption by defining the framework so the equipment manufacturer supplies, through the use of library modules, the data processing and communication elements to match the functional requirements. Minimising the use of resources, i.e. silicon area, is addressed by allowing the processing and communication modules to be implemented on a shared resource. As this inevitable involves a trade off with power consumption the precise mechanism for temporal and spatial scheduling is left to the manufacturer. By using FDL and a set of library functions the granularity of the configuration description and hence the sophistication of the SDR solution can evolve with technology and the regulatory environment.

The work in phase 1 of the E2R project has been mainly conceptual, in the next phase we will develop a proof of concept platform. We will also start converging our work with bodies such as OMA and OMG as well as work being done in the Commercial Working Group of the SDR Forum.

7. ACKNOWLEDGMENT

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

8. REFERENCES

[1] IST-2003-507995 E²R Project, http://www.e2r.motlabs.com

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

[3] J. Chou, "Design a successful VoWLAN system", EETimes online, 9th September 2005, http://www.wirelessnetdesignline.com/howto/showArticle.jhtml?articleID=170101775

[4] IEEE Wireless LAN Edition, A compilation based on IEEE Std 802.11TM-1999 (R2003) and its amendments

[5] IST-1999-12070 TRUST project

[6] IST-2001-34091, SCOUT Project, http://www.ist-scout.org/

[7] WP 4.1 Report on WP4.1 Research & Developed Technology, IST-1999-10287 CAST Project Deliverable

[8] Software Communications Architecture Specification, JTRS 5000, SCA V3.0, Joint Tactical Radio System (JTRS) Joint Program Office, August 27th 2004

[9] J. Chapin, V. Lum and S. Muir, "Experiences Implementing GSM in RDL (The Vanu Radio Description Language™)," MILCOM 2001, October 2001

[3] J.A. Stankovic, M. Spuri, M. Di Natale and G. Buttazzo, "Implications of Classical Scheduling Results For Real-Time Systems", Computer, Vol. 28, No. 6, June 1995.

[10] R. Burgess and S. Mende, "The Role of Configuration Data and a Configuration Control Module in an End-to-End (E²R) Software Radio System," 14th IST Mobile & Wireless Communications Summit, Dresden, 19th - 23rd June 2005

[11] Unicoi Fusion real-time XML parser, http://www.unicoi.com/fusion_web/fusion_xml_sax_microparser.htm

[12] TinyXML, http://www.grinninglizard.com/tinyxml/

[13] Argosy TelCrest, XML Parser, http://www.argosytelcrest.co.uk/xmlparser/index.html

[14] kXML, http://kxml.org/

[15] Xparse-J, http://www.webreference.com/xml/tools/xparse-j.html

[16] T, Wiangtong, P. Y.K. Cheung, W. Luk, "Hardware/Software Codesign" IEEE Signal Processing Magazine,Vol 22, No 3, May 2005

[17] J.A. Stankovic, M. Spuri, M. Di Natale and G. Buttazzo, "Implications of Classical Scheduling Results For Real-Time Systems", Computer, Vol. 28, No. 6, June 1995

[18] E²R Deliverable D4.4 "Configuration Control Module Architecture and Interface Specification", IST-2003-507995 E²R Project Deliverable, July 2005

[19] E2R Deliverable D4.5 "Configuration and Control Strategies" , IST-2003-507995 E²R Project Deliverable, July 2005

[20] B. Steinke et al, "Hardware Abstraction Architecture based on Configurable Execution Modules for Functional System Chains", WG6 Reconfigurability, WWRF#13 Meeting, Jeju Island, Korea, March 2005

TOSHIBA

Power efficient and real-time configuration of resources in an end-to-end reconfigurable system

Craig Dolwin Toshiba Research Europe Ltd
Rollo Burgess Toshiba Research Europe Ltd
Bend Steinke Nokia Research Center

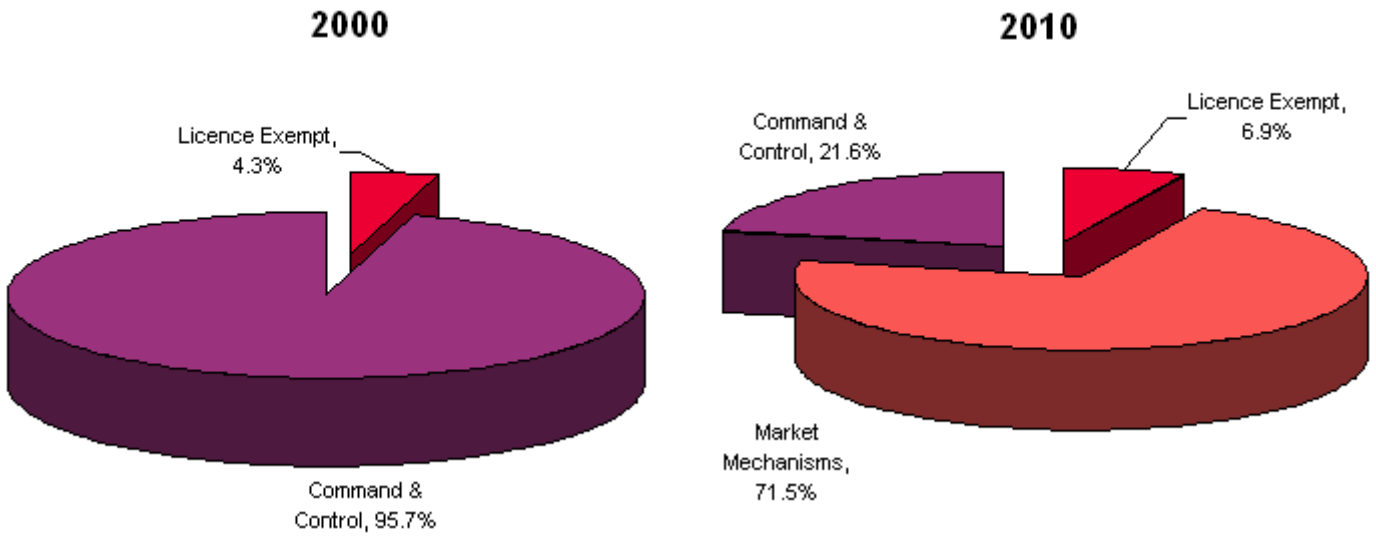
- Motivation
- Challenges
- Architecture
- Scenario
- Description
- Summary
- Future Work

Motivation

- Why reconfigurable rather than multi-mode ?
  - Bug Fixing
  - Algorithm Adaptation
  - Reduced Time to market
  - Reduced Validation and Verification Costs
  - Reduced licensing costs
    - Only pay for IP you are using.
  - NRE on SoC development amortised across more products
  - Support for RAT/waveforms not known at SoC design time
  - Support for future de-regulation of spectrum…..

Motivation – Spectrum Deregulation

- Management of spectrum will move from Command & Control to Market Driven



Source: OFCOM http://www.ofcom.org.uk//

Motivation – Spectrum Deregulation

- Spectrum will be allocated without restrictions on technology/waveform used within allocated band.
- Operators will initially use standardised technology (WCDMA, IEEE802.11 etc) as they see fit
  - Defined by equipment availability
- By using SDR enabled equipment the operator may develop proprietary RAT/waveforms to add value and differentiate their services
- Ultimately we move away from long and lengthy standardisation of RAT/Waveforms towards standardising the configuration interface for SDR

Challenges

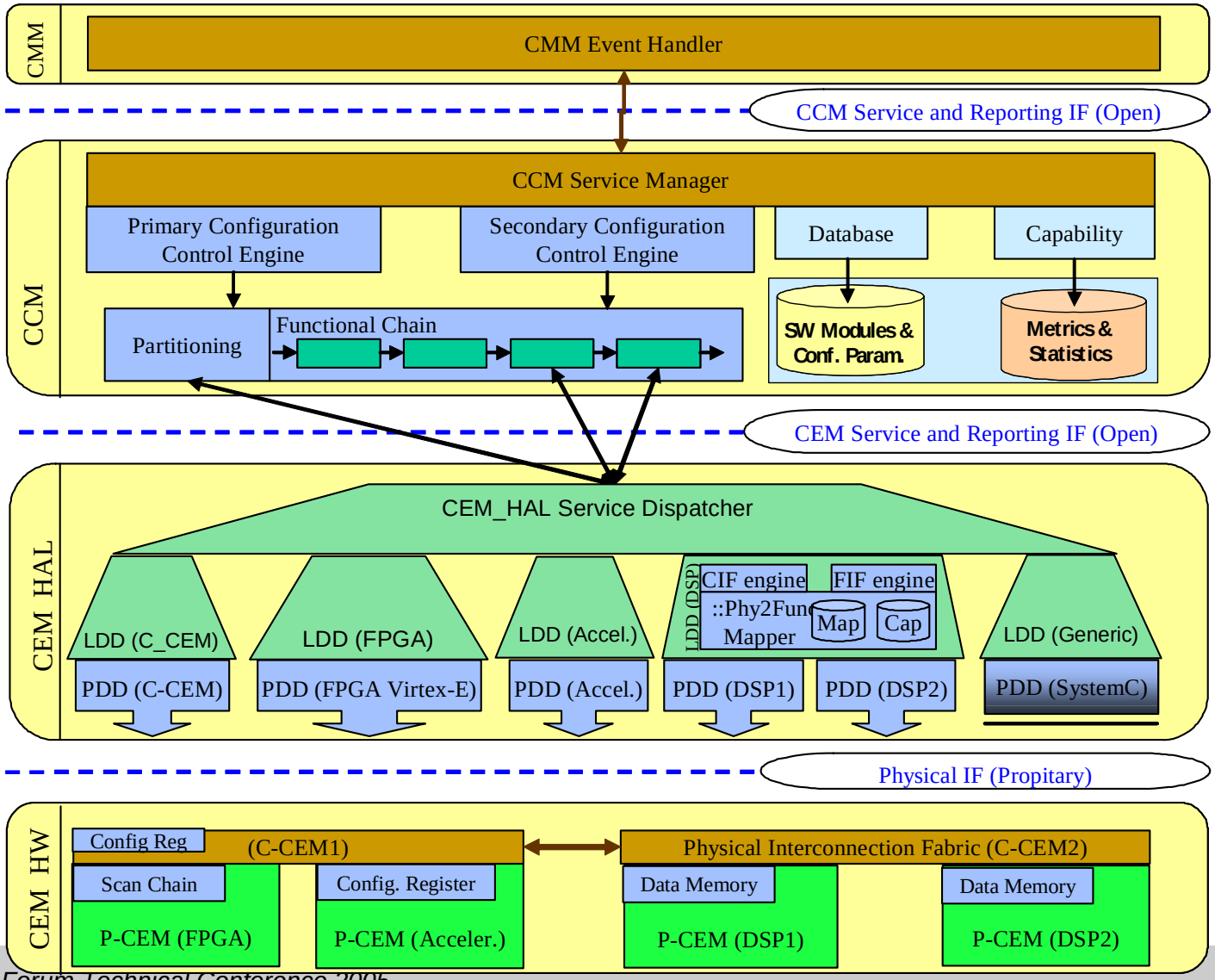
- Reconfigurability comes at a price:
  - Power Consumption
    - Reconfigurable Fabrics (FPGA) x10 (?) of ASIC equivalent [1]
  - Increased silicon area
    - Reconfigurable Fabrics (FPGA) can use x5 – x10 more silicon than ASIC solution [1]
    - Static RAM x 4 more silicon area than ROM
    - Need to reuse reconfigurable fabric by same amount to be cost effective over multimode
- Reliable Real-time Operation
  - Increased overhead for supporting reliable RTOS
  - Sophisticated Scheduling Algorithms Required
- Common framework/API/language required to support reconfiguration of platforms from multiple manufacturers with different levels of support for SDR

[1] H. T. Feldkamper, T. Gemmeke, H. Blume, T. G. Noll : “Analysis of reconfigurable and heterogeneous architectures in the communication domain”, ICCSC 2002, 28th June 2002, St Petersburg Russia.

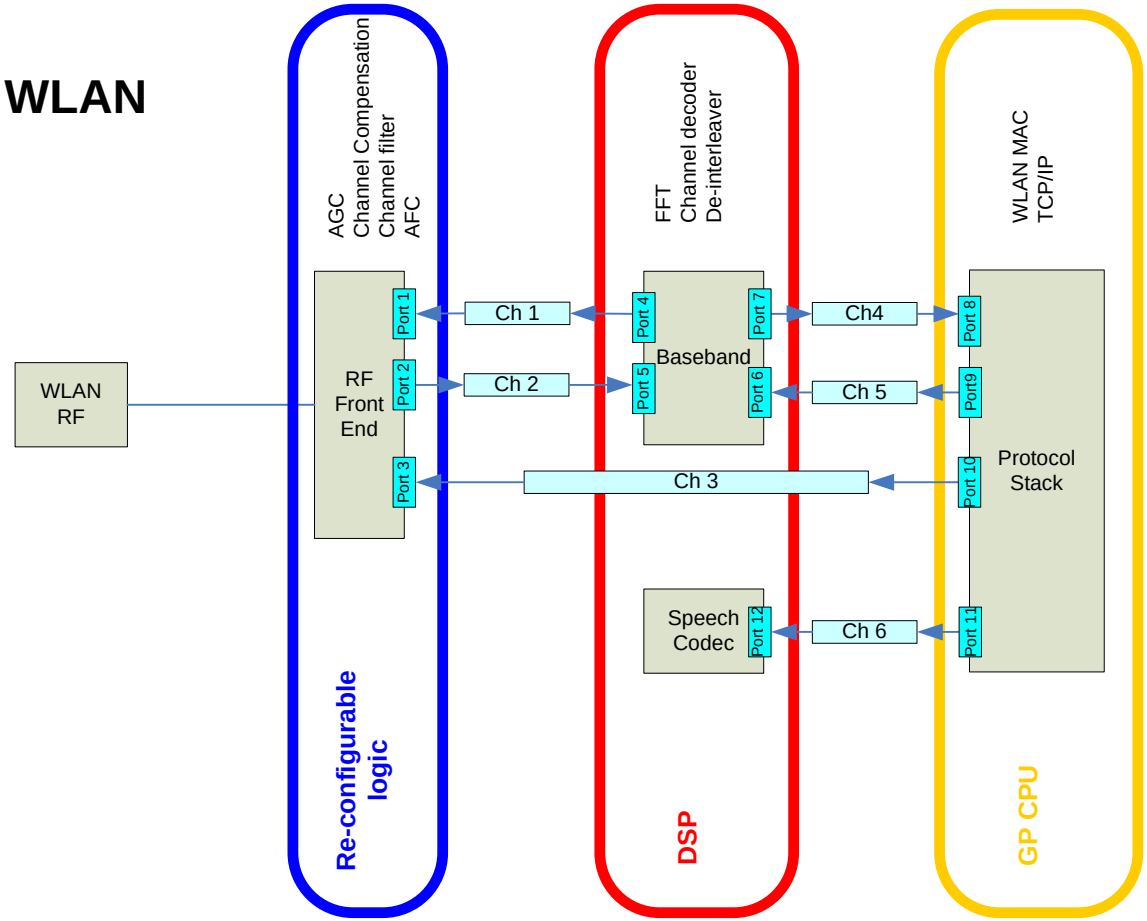
Architecture – principles

- Minimise restrictions on equipment manufacturers to develop cost effective and power efficient SDR
- Support gradual evolution of SDR
- Support different levels of configuration granularity
- Define a single module to control shared resources
  - Configuration Control Module (CCM)
- Standardise an API that will allow equipment independent configuration
  - Service API
  - Signal Processing Modules supplied by equipment manufacturer

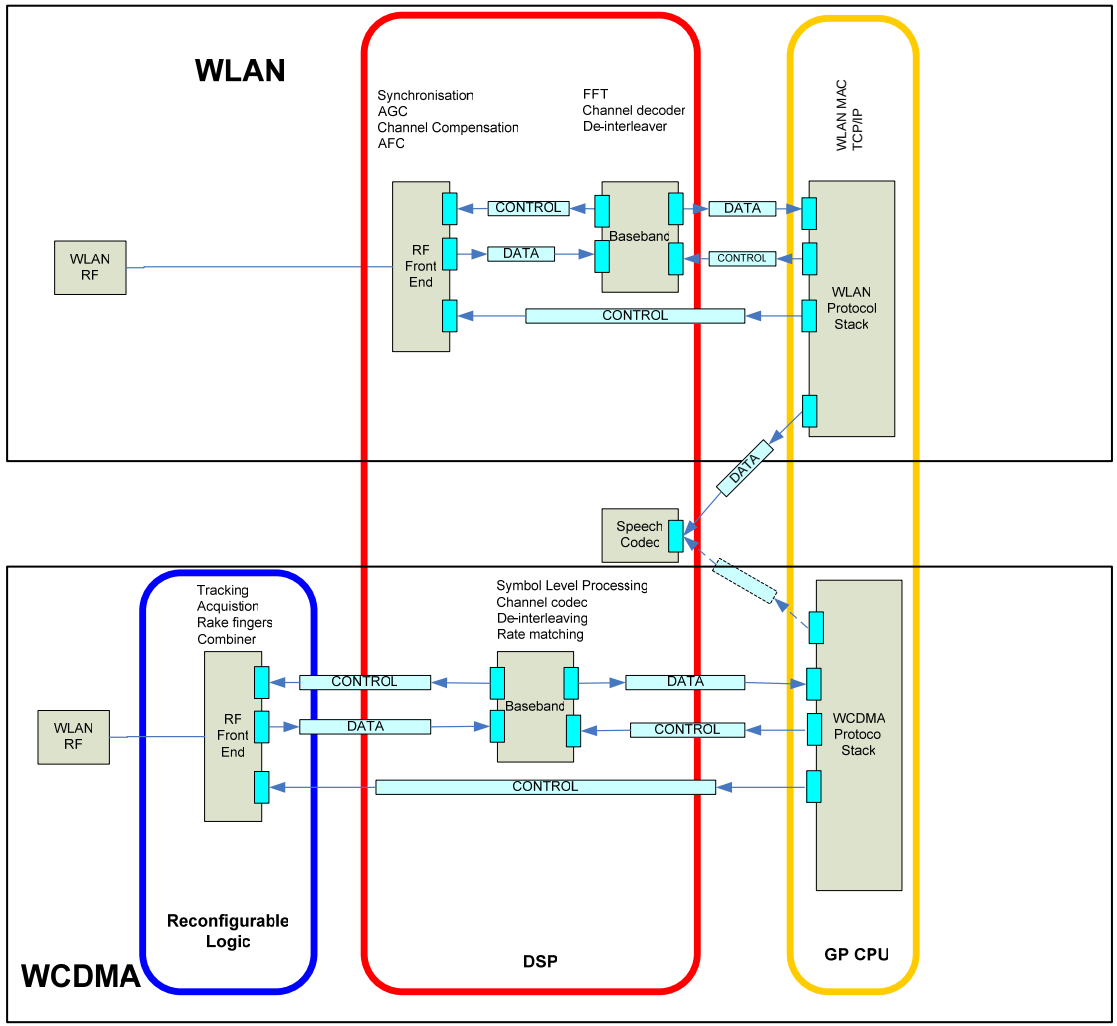
Architecture



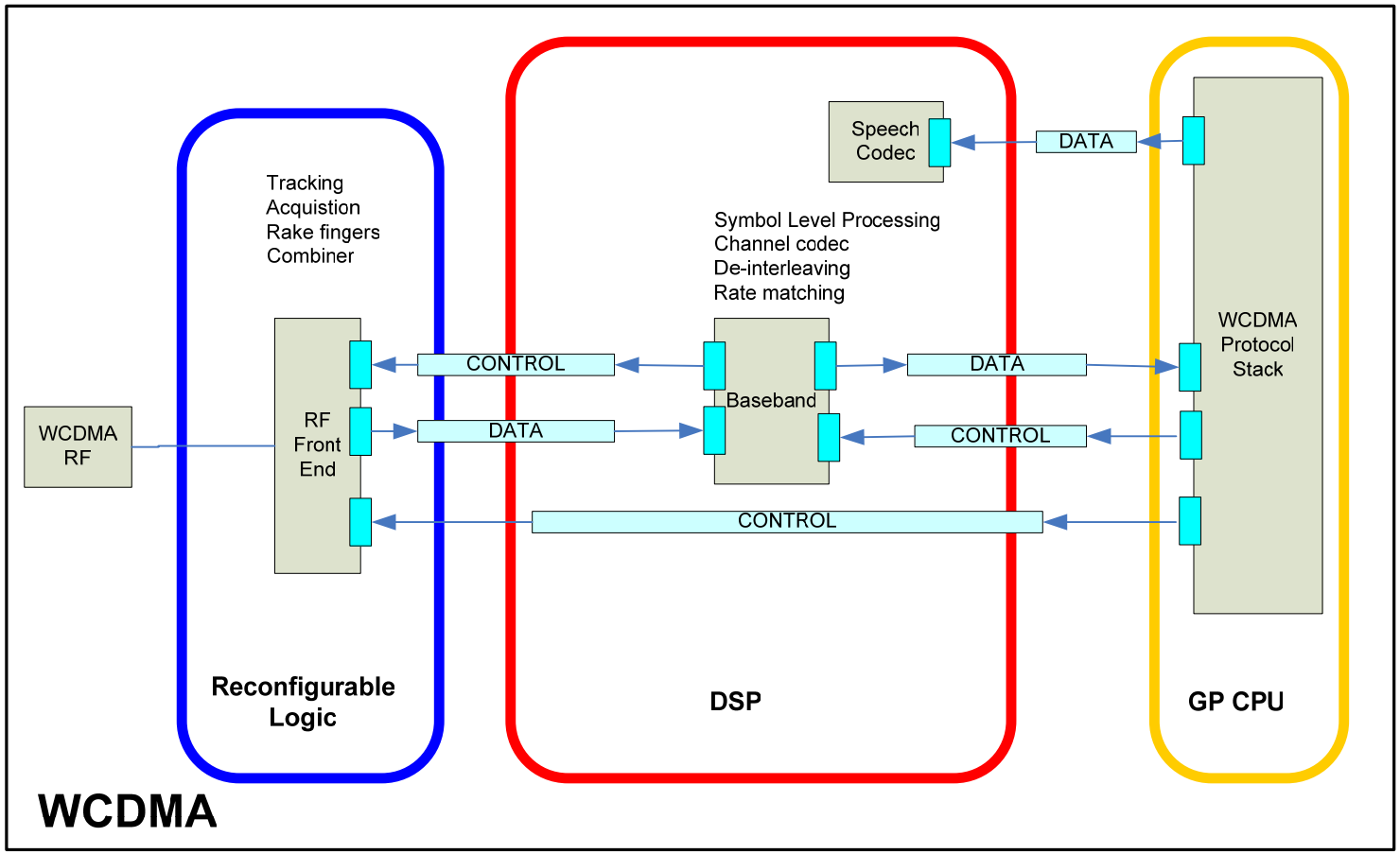
VOIP Switch over Scenario



VOIP Switch over Scenario

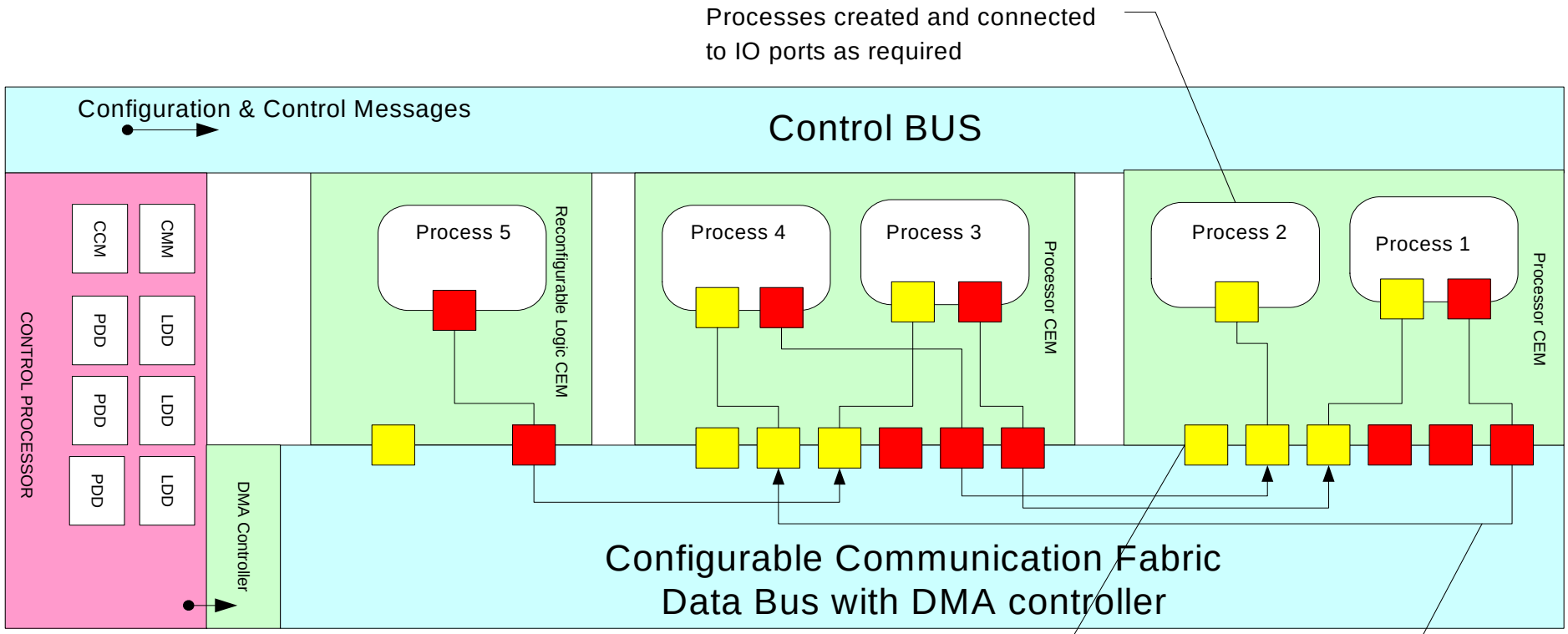


VOIP Switch over Scenario



VOIP switchover scenario

Example SoC architecture



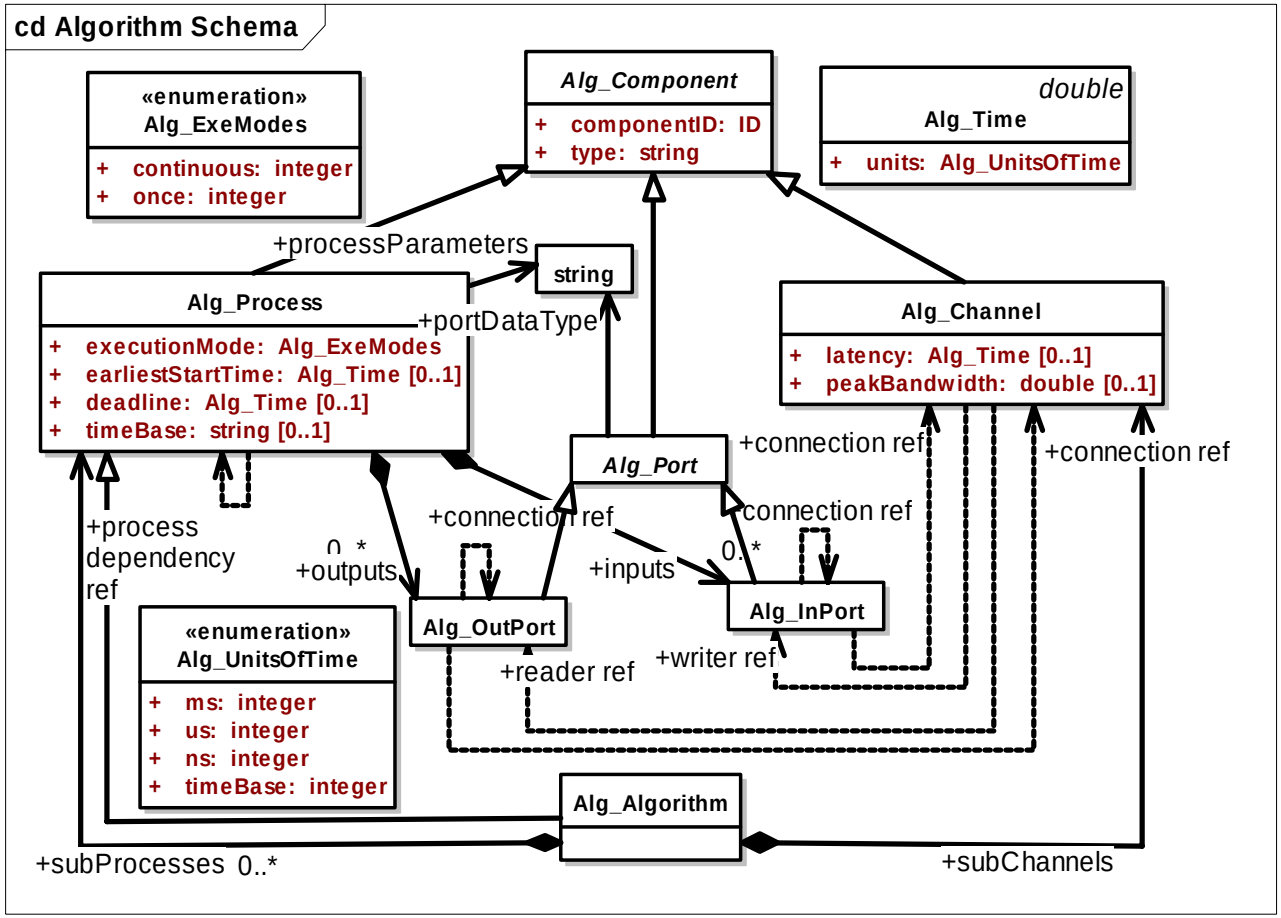
Staged Re-Configuration

- Prepare Installation
  - Supply CCM with functional description of configuration
  - CCM runs spatial scheduling followed by temporal scheduling
    - Spatial scheduler maps functions to resources and identifies missing Signal Processing Modules
    - Temporal Scheduler defines when resources execute
- CCM requests missing Signal Processing Modules
- Signal Processing Modules are installed into resources
- Each Signal Processing Module is started
- New modules are connected to data stream while old modules are disabled

Functional Description Language (FDL)

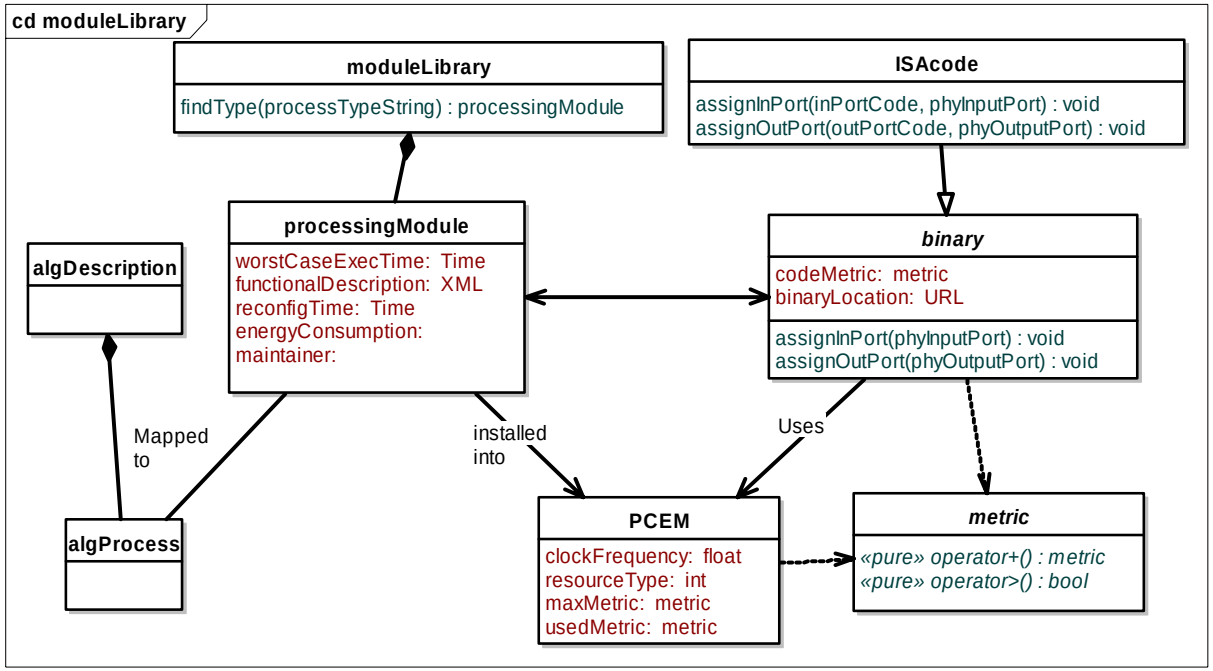
- Functional Description defined using FDL
  - Signal Processing elements
    - Includes timing constraints
    - Precedence constraints
  - Channels
    - Peak bandwidth requirement
    - Max latency requirements
  - Ports
- FDL generic enough to support multiple levels of granularity
- FDL is independent from underlying platform

Functional Description Language (FDL)



Signal Processing Modules

- Building blocks for each configuration
- Supplied by Equipment Manufacturer
- Stored in local database and network database



Configuration Control Module (CCM)

- Proprietary to Equipment Manufacturer
- Implements a standardised Service API
- Interprets functional description written in FDL
- Uses Signal Processing Modules to implement configuration

Summary

- Power Consumption & Silicon Area
  - Allow manufacturer the maximum of flexibility to implement a proprietary solution for SDR with minimal constraints
  - Centralise the configuration and control to allow optimised use of resources
  - Include communication channels into functional description
- Reliable Realtime operation
  - Specify timing constraints within functional description
  - Allocate a single entity (CCM) the responsibility of guaranteeing compliance
- Standard Configuration API
  - XML based functional description language
  - Manufacturer supplies signal processing modules
  - Multiple levels of configuration granularity supports evolution from multi-mode to SDR

Future Work

- Move from paper design to physical implementation (proof of concept) in Phase 2 of E2R
- Contribute to OMA by developing the basic functions required for implementing SDR.
- Converge with work done in OMG and SDR Forums Commercial Working Group
- Current assumption is CCM will be located in equipment → but could be located in network if complexity too high: Need to Investigate

Acknowledgment

This work is being performed within the framework of the EU funded project E²R (http://www.e2r.motlabs.com). The authors would like to acknowledge the contributions of their colleagues from the E²R consortium