



216715 NEWCOM⁺⁺ DC.1

Report on the state for the art on hardware architectures for flexible radio and intensive signal processing

Contractual Date of Delivery to the CEC: T0+12

Actual Date of Delivery to the CEC: T0+12

Editor(s), (names and affiliations):

Dominique NOGUET (CEA)
Amer BAGHDADI (CNRS/TELECOM Bretagne),
Christophe MOY, (CNRS/SUPELEC
Guido MASERA (CNIT/Politecnico di Torino)

Participating institutions: CEA, CNRS, CNIT, Iasa, RWTH Aachen, UPC

Contributors: (names):

Laurent ALAUS, Fabien CLERMIDY, Dominique NOGUET (CEA)
Guido MASERA, Fabrizio VACCA (CNIT/Politecnico di Torino)
Luca FANUCCI, Massimo ROVINI, Giuseppe GENTILE (CNIT/ U. Pisa)
Raymond KNOPP, Karim KHALFALLAH, Najam Ul Islam MUHAMMAD (CNRS/EURECOM)
Christophe MOY, Loïg GODARD, Jacques PALICOT (CNRS/Supelec)
Amer BAGHDADI, Olivier MULLER, Hazem MOUSSA (CNRS/TELECOM Bretagne)
Konstantinos MANOPOULOS (IASA)
Vuk MAJOREVIC, Ismael GOMEZ, Antoni GELONCH (UPC)
Filippo BORLENGHI (RWTH Aachen)

Internal Reviewer(s) (names and affiliations): Sergio Benedetto (ISMB)

Workpackage number: WPRC

Nature: R

Total Effort Spent: 17 MM

Dissemination Level: Pu

Version (1,2,...): 1

Abstract:

This deliverable presents the main trends related to advanced digital communication system hardware design and implementation. The emphasis is put on the PHYsical layer operators because they are the most hardware demanding operations. Besides, these PHY functions have to be housed in terminals which are most often handheld battery operated devices, leading to stringent requirements.

The document is organised in 3 core parts. The first one captures the state of the art in Multi Processor System on Chip which has been given a strong emphasis over the last past years. The second core section focuses on multi-standard architectures which provide the flexibility required to address various Radio Access Technologies. Finally, the most computational intensive blocks are discussed and the example of the FFT and the LDPC decoders is detailed.

Keyword list:

Architecture, hardware, flexible radio, parallel processing, LDPC/Turbo decoders, FFT processors

Acronyms	3
List of figures	5
List of tables	6
1. Introduction	7
2. Flexible MP-SOC architectures	8
2.1. On-chip Communication Network (NoC)	8
2.1.1. Introduction and overview of NoC and their importance in MPSoC design	8
2.1.2. Taxonomy and classification of NoC	10
2.2. Application Specific Instruction-set Processors (ASIPs)	11
2.2.1. ASIP Design methodologies and tools	11
2.2.2. ASIP-based LDPC decoders	17
2.2.3. Multi-ASIP architecture for flexible turbo decoding	19
2.3. State of the art implementation of NoCs / MP-SOCs based channel decoders	22
2.3.1. IntraIP NoC for a flexible LDPC decoder	23
2.3.2. Binary de Bruijn NoC for a flexible LDPC/turbo decoder	25
3. Multi-standard processing for cognitive radio	29
3.1. Software Defined Radio and parametrisation techniques	31
3.1.1. The techniques of parameterization : Introduction	32
3.1.2. The techniques of parameterization : Different Aspects	32
3.1.3. The techniques of parameterization : Common Functions	33
3.1.4. The techniques of parameterization : Common Operators	34
3.1.5. The techniques of parameterization : Contribution to NEWCOM ⁺⁺	34
3.2. Resource management in multi-mode radios	34
3.2.1. Computing Resource Management	34
3.3. State of the art implementation of multi-standards platforms	35
3.3.1. Low-power mutli-standard NoC architecture	35
3.3.2. Open platforms for digital communication systems	38
3.3.3. Generic flexible platform management: P-HAL	44
3.3.4. Cognitive radio equipments management architecture: HDCRAM	47
4. Flexible hardware architecture for computationally intensive processing	53
4.1. FFT operators for OFDM	53
4.1.1. FFT architectures for telecommunication requirements	53
4.1.2. Approaches for Designing FFT Architectures within the NEWCOM ⁺⁺ t	54
4.2. FEC co-decoders	56
4.3. LDPC operators	57
4.3.1. Partially parallel LDPC codes and their implementation	57
4.3.2. Algorithms and Preliminary Architectures For Turbo-LDPC Decoders	60
5. Conclusions	64
6. Reference	65

ACRONYMS

ADL	Architecture Description Language
ASIC	Application Specific Integrated Circuit
ASIP	Application Specific Instruction set Processor
B3G	Beyond Third Generation (of cellular radio-communication systems)
BCJR	Bahl, Cocke, Jelinek and Raviv (algorithm)
CDMA	Code Division Multiple Access
CF	Common Function
CGRA	Coarse Grained Reconfigurable Array
CO	Common Operator
CPO	Carrier Phase Offset
CR	Cognitive Radio
CRC	Cyclic Redundancy Check
DECT	Digital Enhanced Cordless Telecommunications
DSP	Digital Signal Processor
DAB	Digital Audio Broadcasting
DFT	Discrete Fourier Transform
DVB	Digital Video Broadcast
DVB-T	Digital Video Broadcasting-Terrestrial
DVFS	Dynamic Voltage and Frequency Scaling
EDA	Electronic Design Automation
ESL	Electronic System Level
FCC	Federal Communication Commission
FDMA	Frequency Division Multiple Access
FEP	Front End Processor
FFT	Fast Fourier Transform
FIFO	First In First Out (queue)
FPGA	Field Programmable Gate Array
FPU	Floating Point Unit
FU	Functional Unit
GALS	Globally Asynchronous Locally Synchronous
GP	General Purpose
GPP	General Purpose Processor
GSM	Global System for Mobile Communications
GUI	Graphical User Interface
HDL	Hardware Description Language
HLL	High Level Language
Hiperlan	High Performance Radio Local Area Network
IC	Integrated Circuit
IS95	Interim Standard 95 (CDMA)
ISA	Instruction Set Architecture
ISE	Instruction Set Extension
ISEF	Instruction Set Extension Fabric
ISS	Instruction Set Simulator
ITRS	International Technology Roadmap for Semiconductor
LAN	Local Area Network
LCG	Local Clock Generator
LDPC	Low Density Parity Code
LFSR	Linear Feedback Shift Register
LISA	Language for Instruction Set Architectures
LPDP	LISA Processor Design Platform
MAC	Multiply Accumulate
MC-CDMA	Multi-Carrier CDMA
MIMD	Multiple Instructions Multiple Data

MIMO	Multiple Input Multiple Output
OFDM	Orthogonal Frequency Digital Multiplex
MPSoC	Multi Processor System on Chip
NoC	Network on Chip
PDC	Personal Digital Cellular
PE	Processing Element
rAGU	Reconfigurable Address Generation Unit
RF	Register File
RAT	Radio Access Technology
RISC	Reduced Instruction Set Computer
RTL	Register Transfer Level
SDR	Software Defined Radio
SIMD	Single Instruction Multiple Data
SoC	System on Chip
SWR	Software Radio
TLM	Transaction Level Model
UMTS	Universal Mobile Telecommunication System
VHDL	Very high speed Hardware Description Language
VLIW	Very Long Instruction Word (processor)
VLSI	Very Large Scale Integrated circuit
WCDMA	Wideband CDMA

LIST OF FIGURES

Figure 2-1: the main actors in NoC Design worldwide.....	9
Figure 2-2: Performance-Flexibility Trade-off for Processing Elements [Noll].....	11
Figure 2-3: LISA Architecture Exploration Flow	14
Figure 2-4: rASIP Design Flow [Kar08].....	16
Figure 2-5: Iterations, speed gain and efficiency for sub-block parallelism with message passing technique, DVB-RCS, R=6/7, 188 bytes frame, Log-MAP algorithm.....	20
Figure 2-6: (a) ASIP architecture, (b) BCJR computation unit, (c) Adder node, (d) Max node, (e) Control unit	20
Figure 2-7: ASIP based multiprocessor architecture for turbo decoding.....	21
Figure 2-8: torus NoC topology	23
Figure 2-9: Processing Element architecture.....	24
Figure 2-10: Extrinsic information exchanges in parallel turbo decoder and partially-parallel LDPC decoder	26
Figure 2-11: Schematic view of an 8-processor network with processors, routers, and network interfaces	26
Figure 3-1: Processing chain of a DCH UMTS channel	30
Figure 3-2: Typical OFDM transceiver.....	31
Figure 3-3: universal receiver.....	32
Figure 3-4 : Two Techniques for parameterization.....	33
Figure 3-5. The cognitive computing cycle.....	35
Figure 3-6 : DVFS NoC architecture	36
Figure 3-7 : NoC unit architecture	37
Figure 3-8 : USRP base board with example daughterboards.....	39
Figure 3-9 : USRP Architecture	39
Figure 3-10 : USRP Receive Path Architecture	40
Figure 3-11: hardware platform block diagram overview.....	41
Figure 3-12: digital board and housing picture	42
Figure 3-13: Express MIMO overview	44
Figure 3-14: P-HAL Layers Schematic.....	47
Figure 3-15: (a) Mitola's cognitive cycle, (b) simplified version	47
Figure 3-16 – Simplified OSI model for cognitive radio context.....	48
Figure 3-17 – Cognitive radio equipment functional block diagram	48
Figure 3-18 – Meta-model of the HDCRAM.....	49
Figure 3-19 – From an HDCRAM to an HDCRAM simulator.....	50
Figure 3-20: HDCRAM meta-model	51
Figure 3-21: Blind standard recognition sensor	52
Figure 4-1: decoder network	58
Figure 4-2: PE architecture for LT-BPA	58
Figure 4-3: 2-state equivalent trellis of a parity-check constraint.....	62

LIST OF TABLES

Table 2-1: ASIP for serial LDPC decoding performance.....	18
Table 2-2: Performance for WiMAX double binary turbo decoding	21
Table 2-3: ASIC Synthesis Results of Butterfly, Beneš 2N-N and de Bruijn Networks for a 16-Processor Turbo Decoder	27
Table 2-4: Comparison ASIC Synthesis Results of LDPC Decoders	28
Table 3-1: Definition of Unit Power Modes.....	37
Table 3-2: MAGNET board available resource	42
Table 4-1: Air Interface Operations and relevant Macro Processing blocks.....	55
Table 4-2: LDPC implementation report.....	59

1. INTRODUCTION

Recent advances in digital wireless communications introduced the use of complex and computational intensive algorithms. This is particularly true as far as the PHYSical and MAC layers are concerned. Indeed, a general trend of these digital communication systems is to improve as much as possible the use of the spectrum resource, which is a scarce and expensive commonality. Efficiency means in this case spectrum efficiency (bit/Hz/s) but also coverage, coexistence, and quality of service provision. To that end, many techniques have been proposed over the last decade mainly, such as new modulation schemes (eg. WCDMA and OFDM), space time coding (or in a broader sense MIMO) techniques, channel coding, etc that has pushed performance close to theoretical capacity limit [Hoch03].

This trend has put hardware designers under pressure; as they have to tackle these highly demanding schemes, while coping with power consumption issues and limited evolution of the silicon technology. Considering both the International Technology Roadmap for Semiconductors (ITRS) [ITRS] and the evolution of wireless communication standards, is indeed a good way to understand that the evolution of the wireless world cannot be caught up by the Moore's law alone and that new architectural concepts have to be found to fill the gap. This moved the centre of attention to the exploitation of parallelism and, unavoidably, opened new questions about how to exploit the different levels of parallelism and how to develop consistent interconnection systems between the processing elements. These open questions are at the core of MP-SoC research. These systems try to exploit independent-tasks (or functional parallelism) by mapping them to a large number of processors, interconnected via a proper communication structure (e.g., NoC). Considering power consumption leads to even more stringent requirements since battery technology moves yet at a slower pace. The paradigm of MP-SoC, including state of the art architectures and trends is discussed in chapter 2 of this document.

As it was stressed above, wireless technology moves fast and more and more standards are to be considered at design time and used/maintained over their lifetime. This makes flexibility a must for current transceiver design. Over the past few years, chipset and equipment manufacturers have adopted a platform approach for the design of a new release to enable to consider the evolution between standards in an incremental efficient fashion in which a new chipset is considered as an evolution of its predecessor rather than a brand new design. However, such a methodology, though using a flexible approach at the design stage, does not necessarily lead to a flexible instantiation eventually. Yet, another approach consists in considering that a transceiver needs to handle flexibility in operation. This interest has been increasing over the past years and is referred to as Software Defined Radio (SDR) [Mit95]. In fact, two levels of flexibility can be considered. The first one captures the ability of a transceiver to support a variety of different modes within a given standard. This is used for instance in adaptive modulation and coding schemes. The second one relates to the fact that a modern transceiver has to handle different standards that are switched from one another depending on availability and user needs. However, current solutions still exhibit low performance either in terms of flexibility or in terms of power consumption. Recent state of the art and trend on this topic is discussed in chapter 3. The use of flexible architectures in the framework of cognitive radio is briefly addressed also.

Finally, some specific functions have extreme requirements in terms of computational power and throughput. For these specific blocks it is impossible with today's technology to handle highly flexible design which most of the time introduces an increased overhead in area/power consumption. For these functions, parallelism exploitation can be a good approach and the designers often consider the use of parallel or even MP-SoC based approach to tackle them. The examples of the FFT block and channel decoders like turbo and LDPC are highlighted in chapter 4. They benefit from the concepts introduced in chapter 2.

2. FLEXIBLE MP-SOC ARCHITECTURES

The implementation of complex (iterative) wireless communication systems becomes essential to reach the performances nowadays required in term of quality of transmission. Dedicated hardware architectures (i.e. ASIC) implementing parts of these systems are already tackled in several academic and industrial research teams. However, the requirements of: (1) very low error rate, (2) very high throughput and (3) increased flexibility of the implementation, make the resort to adequate multiprocessor architectures inevitable. Parallelism exploitation at the task level becomes crucial due to the algorithm-specific inherent limitations of the lower parallelism levels. In this context, MultiProcessor System-on-Chip architectures have been widely investigated in the last years in order to accommodate the increasing throughput and flexibility requirements of emerging wireless communication standards.

Considering this approach implies typically diverse skills and tasks:

- parallelism exploration of the considered algorithms (leading to parallel or loosely coupled sub-tasks)
- computational resource design (already available cores to be embedded in the SoC, or ASIPs specifically tailored around the class of applications to be supported)
- on-chip communication resource design (ranges from simple bus architectures up to sophisticated Network-on-Chip solutions)

Defining architecture and instruction set of the processing elements and designing the communication structure involve the exploration of several alternatives that need to be investigated, functionally validated, characterized and compared in terms of cost and performance.

Besides application parallelism exploration and application-specific processor design, the on-chip communication network connecting the multiple on-chip cores constitutes a major issue. Conventional on-chip busses become inefficient in large systems and the nanotechnology integration issues (propagation delay, crosstalk, etc.) make their use no more practical. In this context, Network-on-Chip has recently emerged as a new paradigm allowing to cope with these major design issues, and more particularly with the on-chip interconnection issues, and to accommodate future on-chip integration of several hundreds of components. The concept of network on chip aims at adapting the models, the techniques and the tools from the field of computer and telecommunication networks to the context of silicon integration. This is on the way to become crucial for the design of the future embedded systems.

Indeed, for future massively multi-core architectures, the problem will not be any more to design powerful components but mainly to assemble them efficiently. The principal difficulties relate thus to the heterogeneity of the interfaces of the components and their interconnection. In these systems, the communication architecture relies mainly on a set of hierarchical busses. Although they present an attractive solution from the point of view of a simple implementation and low cost, they become the bottleneck when the number of components to be connected increases. Indeed, busses cannot manage anymore the bandwidth required by the processors, memories and IP blocks of heterogeneous architectures. The interconnection and the communication between the various components to be integrated on the same chip become, thus, the critical part of the system.

2.1. On-chip Communication Network (NoC)

2.1.1. *Introduction and overview of NoC and their importance in MPSoC design*

NoC is a growing research field aiming at developing new interconnection schemes inside a chip. It can be used both inside IPs and at a system level. The purpose is to optimize the use of links, to improve the bandwidth while reducing the power consumption. Another task of NoC will be to provide communication primitives for heterogeneous systems, simplifying the platform programming. The main actors in NoC development are described in Figure 2-1. The research is very active worldwide (like the HERMES project in Brazil), both in academic and industrial teams like ST

Microelectronics or NXP. Some start-ups are developing tools and designs for NoC, and few industrial products or prototypes are claiming the use of NoC inside their architectures.

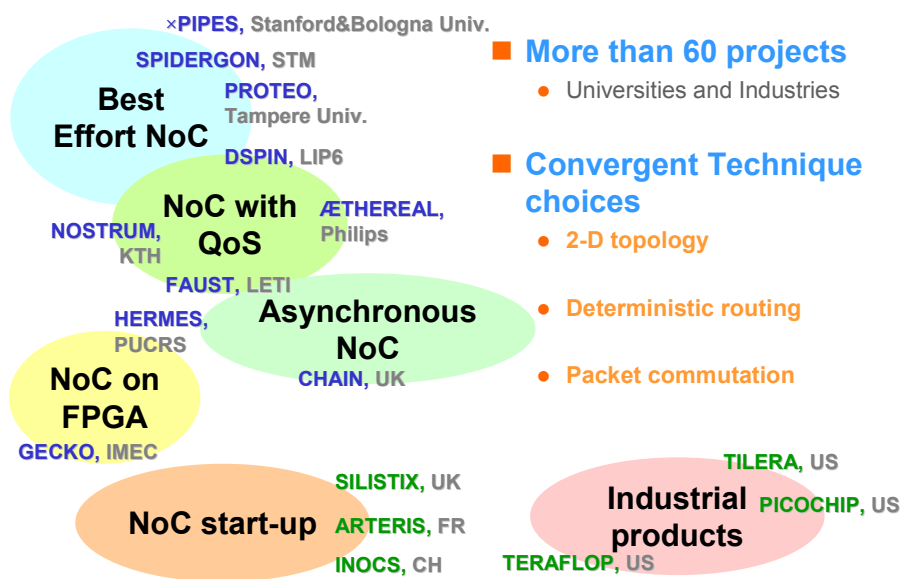


Figure 2-1: the main actors in NoC Design worldwide

The challenges faced by the NoC community include:

- design, with Globally Asynchronous, Locally Synchronous (GALS) and low-power requirements
- architectures with topology choices and low-level communication primitives with Quality of Service, routing policies,...
- Multiple cores programming issues.

The main NoC teams are now briefly described:

- SPIN [Gue00]
 - o LIP6 Laboratory, P. and M. Curry Univ., France
 - o One of the first proposition of NoC
 - o Scalable, Programmable, Integrated Network
 - o Based on a fat-tree topology, wormhole switching, adaptive routing
 - o Given up in 2004, started a new project DSPIN (Distributed SPIN) based on a 2D-mesh topology
- PROTEO [Saa03]
 - o Tampere Univ. of Technology, Finland
 - o Packet switched NoC architecture built from a library of interconnected IP blocks (synthesizable VHDL) for various topologies
 - o Works on the definition of a design methodology and the prototyping of software tools to assist choosing and configuring a particular instance of NoC
- XPIPES [Ber04]
 - o Univ of Bologna (Italy) and Stanford University (US)
 - o Dedicated very high throughput thanks to many repeaters => up to 800 Mhz
 - o Instance-specific network components to support both homogeneous and heterogeneous architectures
 - o Specific tool : XpipesCompiler
 - o Drawback is the power consumption
- NOSTRUM [Mil04]
 - o Royal Institute of Technology (KTH), Sweden
 - o Proposes a packet switched communication platform and a protocol stack for mesh based NoC architecture

- Support both BE and GT traffic based on a hot-potato routing algorithm and by reserving time slots called looped containers
- ÆTHERAL [Goo05]
 - Philips Research Laboratories, Eindhoven, The Netherlands
 - Routers and modular network interfaces which implement high-level services
 - GT traffics utilize a centralized scheduler or a distributed schema for allocation of link bandwidth
 - Backward compatibility with bus and existing on-chip communication protocol (AXI, OCP, DTL)
- QNoC [Dob07]
 - Technion – Israel Institute of Technology, Israel
 - Irregular mesh-custom topology
 - 4 classes of service, implemented with virtual channels, insure the QoS
 - Asynchronous router implementation
- Turbo-decoding ASIP [Mou07]
 - ENSTB, France
 - Multi-stage topologies: Butterfly, Benes
- FAUST [Lat08]
 - CEA-LETI, France
 - Mesh-based topology
 - Asynchronous implementation with QoS
 - Communication primitives for data-flow support
 - Telecommunication application.

2.1.2. Taxonomy and classification of NoC

The traditional NoC approach [Ben01] has been seen as a viable alternative to shared bus topology when scalability and performance are required. Conversely, the continuous evolution of wireless communication standards has ignited the interest toward flexible solutions in order to decrease the time-to-market for new products. These facts have been previously discussed. In this subsection we will focus on NoCs taxonomy and classification first, and then on how they can be exploited to build flexible decoders.

The strategy to achieve flexibility is twofold: on one hand the decoder's processing is carried out using smart processing elements (such as ASIPs), on the other hand the interconnection fabric used shall offer the capability of being re-arranged, providing dynamic connectivity between processing cores. It is well understood that LDPC codes are potentially well suited for parallel decoding: Message Passing Algorithm in its two-phase version is inherently parallel, hence the idea of using a partially parallel architecture to implement it. As far as the communication structure is concerned a short classification has to be made. Traditionally [Ben01], NoCs have been successfully exploited to connect together different IPs, sometimes even provided by different IP vendors. These networks are not tailored around any kind of application, hence their performance in terms of throughput or area occupation can result suboptimal in some specific case as the one targeted here. In [Ben01] the idea of Application Specific NoC (ASNoC) has been firstly explored: as for ASIC, ASNoC are NoC specifically built around an application, hence providing the maximum possible performance yet retaining scalability and flexibility properties of NoC.

Beyond this taxonomy, another classification of NoC can be made depending upon their spatial domain. We will call InterIP NoC any network devoted to interconnect different IPs, hence covering significant portions of the chip's active area. Contrasting this, we will call Intra-IP NoC any network whose domain area is restricted to a single IP. This classification tend to mimic what has been done in the past regarding computer networks: with a similarity InterIP NoC can be viewed as a WAN while an Intra-IP NoC is more similar to a LAN. In a word, at least two orthogonal classification of NoC: can be considered whether they are general purpose NoC or ASNOC, and depending on their *range* we call them InterIP or IntraIP.

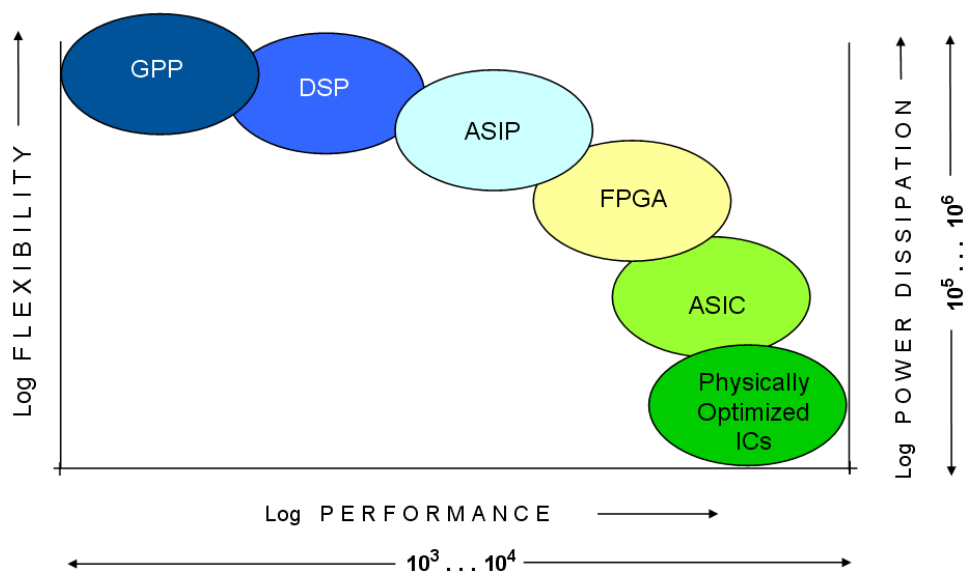
2.2. Application Specific Instruction-set Processors (ASIPs)

2.2.1. ASIP Design methodologies and tools

2.2.1.1. Introduction

Application Specific Instruction-set Processors (ASIPs) are a class of microprocessors with a specialized Instructions Set Architecture (ISA) dedicated to a specific task. They are increasingly used in complex System on Chip (SoC) designs. ASIPs are tailored to particular applications, thereby combining performance and energy efficiency of dedicated hardware solutions with the flexibility of a programmable solution. The main idea is to design a programmable architecture tailored to a specific application, thus preserving a much higher degree of flexibility than a dedicated ASIC solution.

However, several alternative solutions to ASIPs are available in order to implement the desired task, depending on the requirements: for functions which need lower processing power but should be kept flexible a software implementation running on a GPP or a microcontroller may be the best solution; if some additional processing power is needed, moving to a domain specific processor, like a Digital Signal Processor (DSP) optimized for signal processing and offering some additional specialized instructions (e.g. Multiply-Accumulate, MAC), may be beneficial. On the contrary, system modules with very high processing requirements are usually implemented as hardware blocks (Application Specific Integrated Circuits, ASICs, or even physically optimized ICs), with no flexibility at all. If some flexibility is required, field programmable devices like Field Programmable Gate Arrays (FPGAs), which allow for reconfiguration after fabrication, may be the right choice if some price in terms of performance, area and power can be paid. ASIPs represent an intermediate solution between DSPs and FPGAs in terms of performance and flexibility, thus becoming in many cases the best choice to play this trade-off.



Source: T.Noll, RWTH Aachen

Figure 2-2: Performance-Flexibility Trade-off for Processing Elements [Noll]

Several options are available to the ASIP designer for developing his own processor with different degrees of freedom, ranging from the complete specification through an Architecture Description Language (ADL) to a limited post-fabrication reconfigurability achieved via software. This document provides an overview of methodologies and tools for ASIP design.

ASIPs are often employed as basic components of heterogeneous Multi Processor System on Chips (MPSoCs), due to the ever increasing demand for flexibility, performance and energy efficiency, which forces designers to exploit heterogeneous computational fabrics in order to meet these conflicting requirements. Using ASIPs as Processing Elements (PEs) in complex systems is a viable solution in order to play this trade off.

As a consequence, throughout the development process ASIPs for SoCs have to be integrated in their final environment for verification and profiling purposes. This global view of the ASIP running within the complete system is extremely important since design decisions on the ASIP often have side effects on the rest of the system and vice versa. Especially, a sophisticated final verification of the design can only be done when the ASIP is tested in its real environment. Moreover, the communication structure connecting the PEs has a strong influence on the performance of the complete system and therefore it has to be explored early in the design process. Some platforms which allow integrating ASIP development in the overall SoC building process are cited in this document.

2.2.1.2. ASIP Design Approaches

An ASIP design process aims at specializing a processor core. Typically, the development flow of ASIPs starts with the analysis of the application in order to identify its “hot spots”. Then, an initial architecture is defined, in particular with special custom instructions to improve the efficiency for handling those hot spots. Afterwards the applications are run on the processor in order to verify if the target specifications have been met. If that is not the case, the whole flow is iterated to meet the design requirements for given applications.

From this quick overview of the design flow it is clear that some tools are required for implementing it: first, an assembler and a linker are needed in order to run the application code on the processor, together with a compiler if a high-level programming language is used; these tools are required both for design space exploration, when the target application has to be tested in order to improve the architecture, and for software development after the final architecture has been defined. Moreover, if the Instruction Set Extension (ISE) is performed before fabrication, as it is usually the case, an Instruction Set Simulator (ISS) has to be provided so that the application can be run both for profiling and for verification purposes. All these tools directly depend on the instruction set of the processor and hence they have to be retargeted each time that the instruction set is modified. All the suites presented in the following sections provide these tools and the capability to retarget them when needed, while some of them also include the further ability to automate the process of profiling the application and identifying the instructions which are most suitable for ISE.

By looking at available commercial solutions for ASIP design, it is possible to identify three main classes based on the degree of freedom which is left to the designer [Leu06a]:

- Architecture Description Language (ADL) based solutions (e.g. CoWare Processor Designer [CoWare], Target IP Designer [Target]), which can be also defined as ASIP-from-scratch since every detail of the architecture, including for instance pipeline and memory structures, can be accessed and specified by the designer by means of a proper language. This approach results in the highest flexibility and efficiency, but on the other hand it requires a significant design effort.
- Template architecture based solutions (e.g. Tensilica Xtensa [Tens], ARC ARChitect [ARC]), which allow the designer to add custom ISE to a pre-defined and pre-verified core, thus restricting the degree of freedom with respect to the previous approach to the instruction set definition only.
- Software configurable processors and reconfigurable processors (e.g. Stretch [Str], ADRES [Mei05]), with a fixed hardware including a specific reconfigurable ISE fabric which allows the designer to build custom instructions after the fabrication.

For each of these approaches some representative solutions are presented hereafter.

▪ **ADL-based Design Tools**

The most powerful approach for designing ASIPs is based on the specification of the architecture by means of an Architecture Description Language (ADL), specifically aimed at defining programmable architectures at a higher level of abstraction compared to hardware description languages, like VHDL and Verilog. This characteristic allows automating the generation of both the synthesizable RTL description of the processor and all the tools required for software development, which is difficult if lower-level HDLs are used.

Several ADLs have been developed over the past years, including LISA (Language for Instruction-Set Architecture) [Ziv96][Pee99][Hof02], ISDL (Instruction Set Description Language) [Had02][Had97], nML [Fau95], Sim-nML [Raj99][Bas01], Expression [Hal99]. Some of these languages are used in commercial tools. A particularly interesting example is LISA, which is the base of a widely used commercial ASIP design environment, namely CoWare Processor Designer. Therefore, the following paragraphs describe in detail LISA and CoWare tool flow, which is a good example to introduce the characteristics of ADLs and the related ASIP design flow.

CoWare Processor Designer [Pee99] is an ASIP design environment entirely based on LISA. The language syntax provides a high flexibility to describe the instruction set of various processors, such as SIMD, MIMD and VLIW-type architectures. Moreover, processors with complex pipelines can be easily modelled.

Processor Designer's high degree of automation greatly reduces the time for developing the software tool suite and hardware implementation of the processor, which enables designers to focus on architecture exploration and development. The usage of a centralized description of the processor architecture ensures the consistency of the Instruction-Set Simulator (ISS), software development tools (compiler, assembler, and linker etc.) and RTL implementation, minimizing the verification and debug effort.

The LISA machine description provides information consisting of the following model components:

- The memory model lists the registers and memories of the system with their respective bit widths, ranges and aliasing.
- The resource model describes the available hardware resources, like registers, and the resource requirements of operations. Resources reproduce properties of hardware structures which can be accessed exclusively by a given number of operations at a time.
- The instruction set model identifies valid combinations of hardware operations and admissible operands. It is expressed by the assembly syntax, instruction word coding, and the specification of legal operands and addressing modes for each instruction.
- The behavioural model abstracts the activities of hardware structures to operations changing the state of the processor for simulation purposes. The abstraction level can range widely between the hardware implementation level and the level of high-level language (HLL) statements.
- The timing model specifies the activation sequence of hardware operations and units.
- The micro-architecture model allows grouping of hardware operations to functional units and contains the exact micro-architecture implementation of structural components such as adders, multipliers, etc.

By using these various model components to describe the architecture, it is then possible to generate a synthesizable HDL representation and the complete software tool suite automatically.

CoWare Processor Designer is an automated design and optimization environment which utilizes LISA language description to develop a wide range of processor architectures, like SIMD and VLIW as well as processors with DSP- or RISC-specific features. The generation of the software development environment by Processor designer enables to start application software development prior to silicon availability, thus eliminating a common bottleneck in embedded system development.

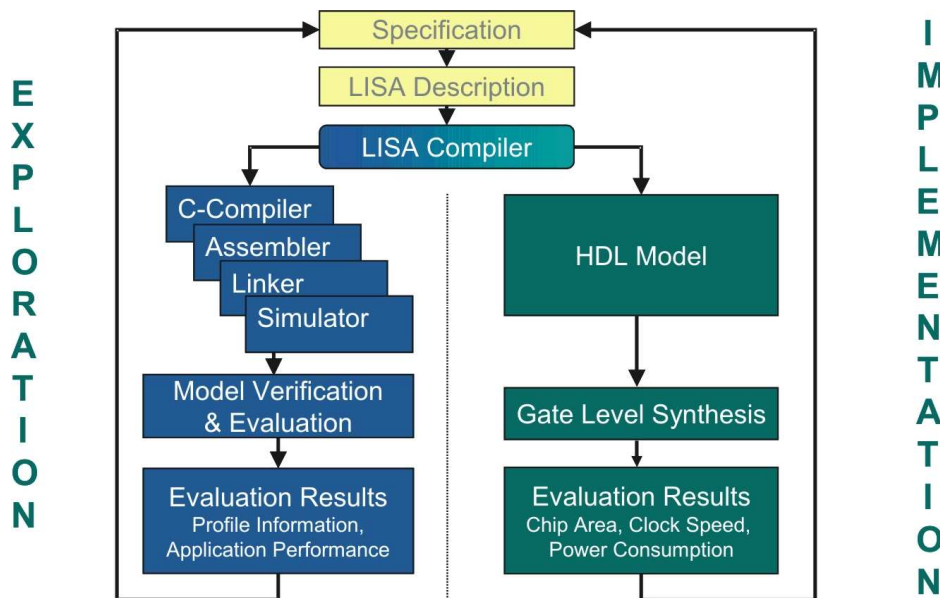


Figure 2-3: LISA Architecture Exploration Flow

As it is shown in Figure 2-3, the design flow of Processor Designer is a closed-loop of architecture exploration for the input applications. It starts from a LISA 2.0 description, which incorporates all necessary processor-specific components such as register files, pipelines, pins, memory and caches, and instructions, so that the designer can fully specify the processor architecture. Through Processor Designer, the ISS and the complete tool suite (C-compiler, assembler, linker) are automatically generated. Simulation is then run on the architecture simulator and the performance can be analyzed to check whether the design metrics are fulfilled. If not, architecture specifications are modified in LISA description until design goals are met. At the end, the final version of RTL implementation (Verilog HDL, VHDL and SystemC) together with software tools is automatically generated.

As previously mentioned, ASIPs are often employed as basic components of more complex systems, e.g. MPSoCs. Therefore, it is very important that their design can be embedded into the overall system design. Processor Designer provides possibilities to generate a SystemC model for the processor, so that it can be integrated into a virtual platform. In this way, the interaction of the processor with the other components in the system can be tested. Furthermore, the exploration as well as the software development of the platform at early design stage becomes possible.

Another example of ADL-based design environment is Target IP Designer [Target], which is based on nML, first developed at TU Berlin and can support the automatic generation of tools. Target Compiler Technologies provides a tool suite for processor development, which uses nML. This retargetable tool suite consists of a compiler (Chess), a linker (Bridge), an assembler/disassembler (Darts), an instruction set simulator (Checkers), a test-program generator (Risk) and an HDL generator (Go).

▪ Template Architecture-based Design Tools

The ADL-based approach gives the designer full freedom in specifying the architecture, meaning that the optimization level in terms of performance and energy efficiency can be very high. However, the design process may become relatively long, especially when the ASIP is designed from scratch. An alternative approach, which restricts the degree of freedom of the designer so as to shorten design

time, takes as a starting point for the design a generic processor template, whose architecture and tools can be configured at a certain degree. In this way, the designer is required to tune the existing architecture and to extend it rather than defining every architectural detail. Hence, the design process is shorter but it cannot reach the optimization level of an ADL-based solution.

An example of template-based tool flow is the Tensilica Xtensa solution [Tens]. In this case, the starting point is a predefined 32-bit RISC-like processor core, which can then be extended in several ways. First, the designer can choose to add some optional execution units, like a multiplier, a MAC unit or a floating-point unit, and interfaces to busses and memories: these options are predefined so that the designer is not free to define every architecture detail like in the previously presented cases; on the other hand, this template-based solution has the advantage of speeding up the design process since all the additional features have already been verified and tested a priori and hence they are easily supported by the tool chain. The second degree of freedom left to the designer is the ability to add custom instructions to the basic instruction set, either by specifying them explicitly in a Verilog-like language named Tensilica Instruction Extension (TIE) or by using the appropriate tool, named Xtensa Processor Extensions Synthesis (XPRES) compiler, which automates the profiling and ISE operation.

The Xtensa development environment also automatically generates all the required tools, such as the compiler and the ISS (SystemC-compatible, meaning that it can be inserted into a more complex system), and the synthesizable RTL code for the customized processor.

A quite similar approach to Tensilica Xtensa is ARC ARChitect [ARC]. In this environment, the designer just needs to configure a pre-existing processor core rather than designing it from scratch. In this case, the starting template is a 32-bit core optimized for low power consumption, to which the designer can add the required execution blocks, like cache, FPU and DSP, and then tune their parameters.

Like in Tensilica solution, there is a second degree of freedom in the design process, which is the extension of the instruction set. However, in this case the extension process is not automated. In fact, it is up to the designer to select the custom instructions and then code their functionality in a proper way so that they can then be integrated into the tool chain. Some examples of possible extensions are compound instructions, SIMD instructions and integrated coprocessor instructions.

Similarly to the previous cases, a SystemC model of the processor can be automatically generated and integrated into the development environment of a virtual platform.

Another example of template-based tool flow is represented by CoWare CORXpert [MIPS], which allows extending pre-defined processor cores, such as MIPS CorExtend processors.

▪ **Reconfigurable Processors**

Another methodology, which is different from those described above, is the so-called software configurable processor. As in the Stretch family of configurable processors, an FPGA-like Instruction Set Extension Fabric (ISEF) is integrated into a Tensilica Xtensa RISC processor core. This ISEF can be reconfigured by system designers to extend the processor instruction set. The advantage of this approach is, that any granularity of computation which can be accommodated into ISEF can be implemented as extended instruction, because the ISEF is an FPGA-like programmable logic fabric.

Apart from the reconfigurable processors with FPGA, there is another approach, which combines the architecture of the processor with a Coarse Grained Reconfigurable Array (CGRA), on which computationally intensive kernels of applications can be efficiently executed. One representative architecture is ADRES, which is introduced in [Mei05]. This architecture can be considered as a close coupling of a VLIW processor with a CGRA. The design of ADRES is based on an architecture template. In the template, parameters, like number of Functional Units (FUs) and Register Files (RFs),

the interconnection topology, the operation set of each FU and the sizes of the distributed RFs are tunable for architectural exploration, regarding the application.

Instead of giving a general tool flow for designing reconfigurable processors, the approaches introduced above can only be applied for a class of architectures. In [Kar08], an ADL-based approach is proposed, which enables general modelling of the reconfigurable ASIPs (rASIPs).

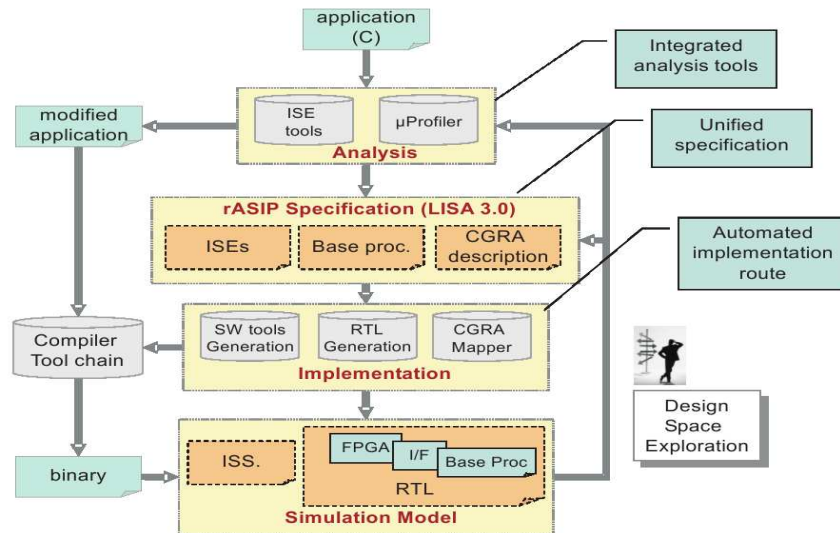


Figure 2-4: rASIP Design Flow [Kar08]

The design flow is shown in Figure 2-4 and its peculiarity is that it combines the various ASIP and CGRA exploration concepts into one architecture exploration tool chain. The key elements of this design flow are introduced below.

1. Unified rASIP Modelling Formalism: The central component of the design flow is a unified specification format which allows the user to describe an entire rASIP architecture. This format is an extension of the existing ADL LISA (the extended language is named LISA 3.0) where the designer can clearly specify which parts of the processor belong to the base ASIP, and which parts go to the FPGA fabric. The unified specification eliminates any need to combine together different unconnected design tools to build a single flow.
2. Automatic Software and Hardware Generation: The software tool and RTL generation capabilities of LISA ADL have been adapted for the rASIP specific extensions. Therefore, the implementation of rASIP hardware and software tools can be easily generated from LISA 3.0.
3. CGRA Exploration: A CGRA format which can be used to describe a variety of coarse grained re-configurable structures is designed. Besides, during the hardware generation process, a complete RTL implementation of the CGRA structure is generated. Additionally, tools necessary to map the reconfigurable parts to the CGRA are automatically derived.
4. Application Analysis Tools: To assist the user in the application analysis phase, a fine-grained application profiler [Kar05] and automatic ISE identification [Leu06b] tool have been integrated into the design flow.
5. Support for Pre- and Post-Fabrication Design Flow: The design flow presented here is for pre-fabrication design space exploration. However, it can be also applied for post-fabrication rASIP customization. The same set of tools can be used for this purpose. New custom instructions can be identified and mapped to reconfigurable fabric in the post-fabrication phase.

Inside the design flow, two important tools for analysing the input application are used: the µProfiler and Instruction-Set Extension tools.

Profiling is a very important step in the ASIP development, which gives hints where the architecture optimization should focus. Traditionally, application profiling can either be done at high level (C/C++) or assembly level. In C-level profiling, standard tools like GNU gprof/gcov can be utilized to extract statistic information about the frequency of line execution and the time spent on the different parts of the application. However, the information obtained from C-level profiling is not accurate enough in many cases since it is done at a very high abstraction level. Another profiling scheme, which is done at the assembly code level, is mostly machine-specific and requires at least processor models (or virtual prototype). However, the execution time of the profiling is quite long due to the low simulation speed.

A fine-grained source code profiling scheme is proposed in [Kar05], which aims at combining the advantages of both classical profiling technologies stated above. This approach begins with lowering the original source code into three address code, which is an intermediate representation containing at most one operation in each line. Instrumentation code lines are then added, acting as aggregate counters of operations for the entire block. All primitive C-level operations are created explicit and hence they can be profiled like regular operations. Moreover, built-in standard code optimizations of this three address code can be done in order to increase the profiling accuracy.

The profiling information can be used for selecting useful instructions to extend the ASIP instruction set, which is called Instruction-Set Extension (ISE). An ISE scheme, which is used to make a partially predefined, configurable RISC-like embedded processor core quickly tuneable to given applications, can also be integrated into the LISA ASIP design flow so as to automate this operation which otherwise should be performed manually by the designer. In [Leu06b], it is proposed that how an ISE scheme can be integrated into the LISA ASIP design flow automatically.

The design flow can be divided into the following major steps: first, the application specification, given as C code mostly, needs to be analyzed to identify major dynamic execution characteristics and hot spots. Custom Instructions (CIs), whose feasible implementation can provide the desired speed-up of hot spots, need to be identified based on the analysis results. This identification of CIs is an optimized selection problem with a huge searching area under the constraints imposed by both the processor core and the human designer. The selected CIs then need to be implemented into the predefined configurable processor template. Interface specification, latency and area constraints need to be met, for instance by means of proper pipeline stage balancing. Finally, software adaptation, in order to make it compatible with compiler, assembler, linker and simulator, tools generation and hardware architecture implementation (generation of RTL HDL model which is merged with predefined processor model) are performed in order to embed the CIs in the hardware processor and its software tool chain.

2.2.2. *ASIP-based LDPC decoders*

As previously introduced, ASIPs are basically highly specialized microprocessors, tailored around the specific application's needs. Starting from this definition, it seems they could be good candidates to implement flexible channel decoders. In particular in [Qua06] presents one of the first ASIP design for LDPC decoding. In this work a single processor approach is used, leading to a serial decoder architecture.

Giving the intrinsic high parallelism of LDPC decoding, this design is not expected to be particularly attractive under the throughput standpoint. To enable some sort of comparison a (1024, 512) code as the one used in [Ber05] is considered. We also consider 8 bit quantization to represent both VN/CN messages as well as CN/VN ones. To implement this serial decoder, the starting point is a general purpose 32 bit RISC core described in LISATeK: then some modifications are applied to this core to obtain a proper architecture able to support Belief Propagation Algorithm. Specifically a four-stage processor pipeline is considered, i.e.:

1. Instruction Fetch stage (IF), which loads (fetches) instructions from code memory and activates the second stage;

2. DeCode stage (DC) that decodes instructions by selecting the proper operation on which these operations are applied;
3. EXecution (EX) stage, devoted to perform the selected operation;
4. Write-Back (WB), which stores results into internal registers or external memories.

Additionally a full bypass mechanism from EX stage back to DC stage was implemented in order to avoid data dependency that could lead to stall into processor's pipeline. As far as the CN implementation is concerned the so-called min* operator is used: since min* is a rather complex operation involving arithmetic processing and table look-up it was decided to map it directly on a custom instruction trying to mitigate its hardware impact.

The obtained ASIP has been fully implemented in two different technological node, namely 0.18 μm and 0.13 μm , standard cell process. In table 2-1 Synopsys Design Compiler results are shown for the designed ASIP processor. Using the application-specific customizations of the processor, 7740 clock cycles are needed to perform a single iteration of the decoding algorithm for the (1024, 512) LDPC code.

Table 2-1: ASIP for serial LDPC decoding performance

Technology	Freq. [MHz]	Area [ηm^2]	Equiv. Gates
0.18 ηm	150.3	405794.5	36 kGates
0.13 ηm	212.7	189089.9	36.4 kGates

Given the figure of the aforementioned table a net decoding throughput of almost 14 MBps / iteration is obtained, when a clock frequency of 212 MHz will be used. With a typical iteration number of 5 the total equivalent throughput would be of 2.8 Mbps. Even if this result seems fairly unsatisfactory if compared with ASIC-based decoders, some considerations ought to be made:

- The proposed ASIP methodology does not rely in any way on the code structure: it does only assume that the min* operator can be used for the CN and that message processing can be performed serially;
- The throughput is mainly spoiled by the completely serial nature of the decoder, being able to process just a single Tanner graph edge per each clock cycle;
- The designed decoder is completely flexible since the code rate, the block size and the internal data representation can be changed without affecting the designed ASIP;
- Anyway even this result should be regarded with respect if one considers that a 3.06 GHz Xeon processor is able to achieve only 450 kbps when a similar code is concerned (C software model performance).

After this first LDPC serial decoder attempt our focus has been directed toward flexible architectures for partially structured LDPC codes, [Din06]. Here the ASIP approach is used in order to deal with some "random placed" ones, possibly leading some collisions in memory access path. The idea behind this second work is to leverage ASIP flexibility to be able to support different code rates and generally different LDPC codes without the need for a complete redesign of the channel decoding unit. Also in this work a 5-stage RISC-like pipeline was used, mainly optimized for the Omega function computation. We also perform the logical synthesis of a 4-stage RISC-like ASIP in order to assess how different pipeline depth would impact on overall system performance. As far as the results are concerned, a decoding throughput of 15 MBps per iteration was achieved, i.e. a net throughput of 3 MBps @ 5 iterations. The total estimated area occupation was of nearly 14000 equivalent logic cells, running at a nominal clock frequency of 400 MHz.

In the last few months, research of ASIP architectures for forward error correction has been focusing of solutions able to map both turbo and LDPC decoding at high throughput, with full memory architecture sharing, full datapath reuse and high energy efficiency.

An application-specific instruction programmable architecture addressing in a unified way the emerging turbo- and LDPC coding requirements of 3GPP-LTE, IEEE802.11n, IEEE802.16(e) and DVB-S2/T2 was recently presented in [Nae08]. The proposal shows a throughput from 0.07 to 1.25Mbps/MHz with efficiencies round 0.32nJ/bit/iter in turbo mode and 0.085nJ/bit/iter in LDPC mode. The area is lower than the cumulated area of dedicated turbo and LDPC solutions. The proposed architecture includes a large SIMD datapath that implements dedicated instructions to execute both LDPC and turbo decoding. Distributed background memory architecture supports the datapath with as many single-ported banks as SIMD slots. These are connected to the core with a read and a write-back cross-bar. Reconfigurable Address Generation Units (rAGU) based on look-up tables (LUT) enable the message scrambling needed by the LDPC and turbo-decoding. Virtual address space where collisions never happen is assumed by the core SIMD application and it is mapped to physical background memory addresses by the rAGUs.

2.2.3. Multi-ASIP architecture for flexible turbo decoding

Channel decoding is one of the most critical blocks in wireless digital communication systems due to the underlying extensive computation and communication of iterative decoding techniques (Turbo decoding and LDPC) and the severe requirements in terms of throughput and flexibility. Flexibility and high-throughput requirements are being widely investigated in this application domain during the last few years. Several implementations have also been proposed. Some of these implementations succeeded in achieving high throughput for specific standards with a highly dedicated architecture (see previous section). However, these implementations do not take into account flexibility and scalability issues. Conversely, others implementations include software and/or reconfigurable parts to achieve the required flexibility while achieving much lower throughput.

In this context, regarding turbo decoding, the application-specific instruction set processor (ASIP) approach was used for a parallel multiprocessor implementation in [Gil03]. Despite the advanced heterogeneous communication network that optimizes data transfer and enables parallel turbo-decoding implementation, the platform lacks performance due to the predefined basic processor core imposed by the *Template Architecture-based* approach.

In [Mul06c], the first ASIP dedicated to turbo decoding using the *ADL-based Design* approach was proposed. Thanks to its performance and the multiprocessor template proposed, the solution is able to cover almost all standards. It can be configured to decode all simple and double binary turbo codes. Besides the specific arithmetic units that make up this processor model, special care was taken with the memory organization and communication busses. Its architecture facilitates its integration in a multiprocessor scheme enabling an efficient and flexible implementation of the turbo decoding algorithm.

2.2.3.1. Parallel processing levels

To define proper limits in ASIP design, parallelism in convolutional turbo decoding was widely investigated. A three-level classification of parallelism techniques is proposed [Mul06b]: BCJR metric level, BCJR-SISO decoder level, and Turbo-decoder level. The second level of this classification, which includes sub-block parallelism and component-decoder parallelism, is thoroughly analyzed on the basis of parallelism efficiency criteria (Figure 2-5) [Mul06c], since it offers the best trade-off between achievable parallelism degree and area overhead. Besides, in order to maximize component-decoder parallelism efficiency, new interleaving rules minimizing propagation time effect are proposed [Mul06c].

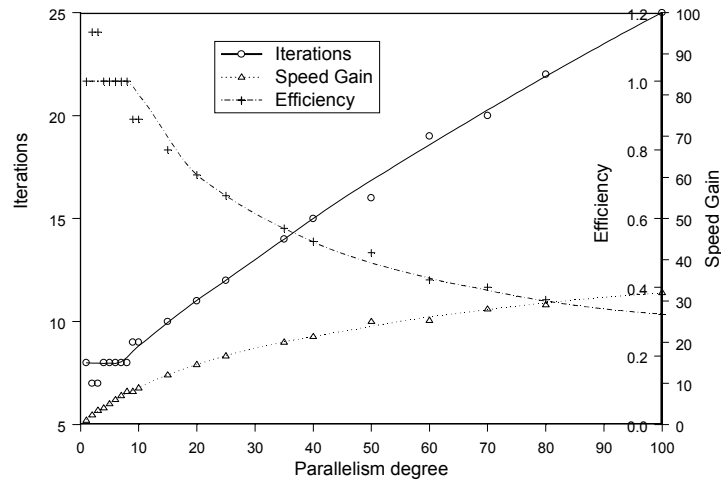


Figure 2-5: Iterations, speed gain and efficiency for sub-block parallelism with message passing technique, DVB-RCS, R=6/7, 188 bytes frame, Log-MAP algorithm

2.2.3.2. ASIP for turbo decoding

Based on this classification, a new flexible and high performance ASIP model for turbo decoding (Figure 2-6) was designed. With its specialized and extensible instruction set and a 6-stage pipeline control, this model presents the flexibility required to support all simple and double binary turbo codes. Furthermore the ASIP achieves high performance by fully exploiting the BCJR metric level parallelism (first level) with a SIMD architecture.

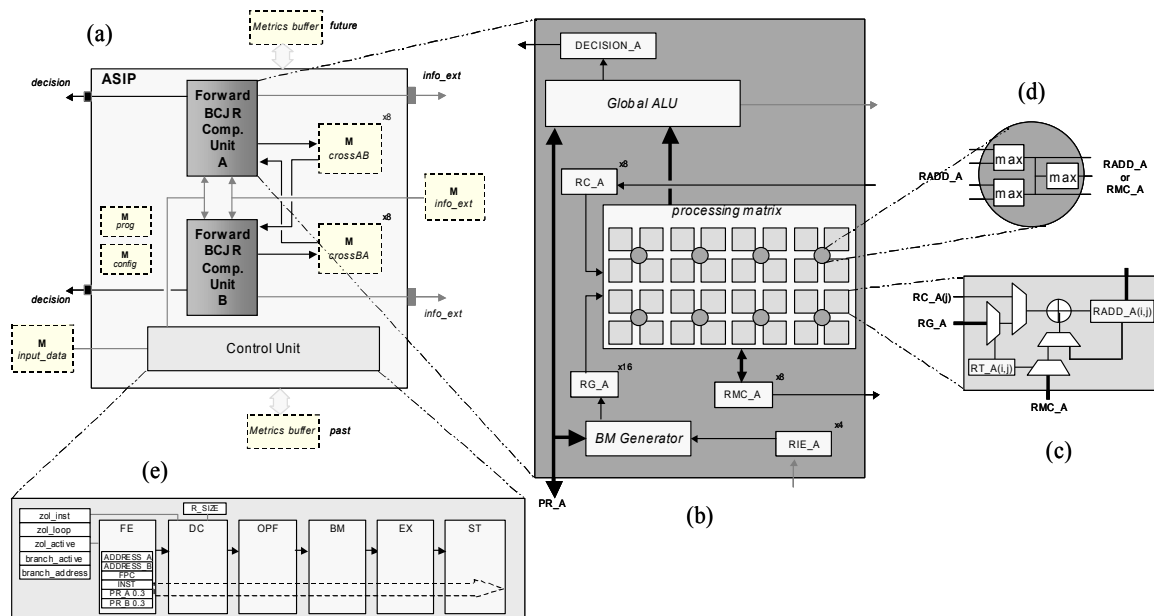


Figure 2-6: (a) ASIP architecture, (b) BCJR computation unit, (c) Adder node, (d) Max node, (e) Control unit

Besides the specific arithmetic units that compose this processor model, special care was taken with the memory organization and communication busses, so that its architecture facilitates its integration in a multiprocessor scheme.

2.2.3.3. Multi-ASIP platform

To perform on-chip communication between ASIPs (Figure 2-7), multistage interconnection networks, such as Beneš Networks [Mou07] is suggested. The designed on-chip networks are scalable, offer all the required routing flexibility, and are transparent from the application point of view thanks to a mastered propagation time. To improve the overall performance, an algorithm modifications is suggested, so that turbo-decoding algorithm could provide priorities to the communication network and enable a better network resource management.

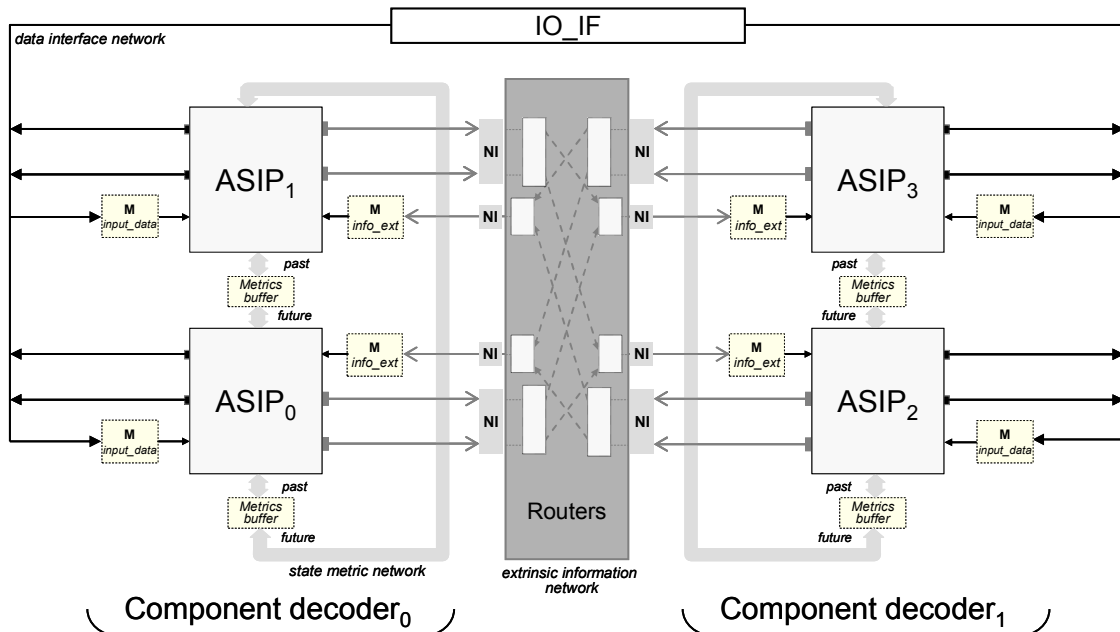


Figure 2-7: ASIP based multiprocessor architecture for turbo decoding

2.2.3.4. Results

Using LISATek *Processor Generator*, a VHDL description of the ASIP architecture was generated. It was synthesized with a Synopsys Design Compiler using ST 90 nm ASIC library under worst case conditions (0.9V, 105C). The ASIP can process 228 Mbits/s in double binary mode and 114 Mbits/s in simple binary mode for an occupied area of 0.37 mm² (Table 2-1). Supporting double binary turbo codes constitutes a unique feature compared to existing dedicated processors.

Table 2-2: Performance for WiMAX double binary turbo decoding

<i>Number of ASIPs</i>	1	4	8	16	32
<i>Throughput [Mbit/s] @ 90 nm</i>	22.8	65.7	135.0	270.0	503.4
<i>Overall area with Butterfly topology (memories+NoC+ASIP) (mm²)</i>	0.37	1.71	3.63	7.68	16.22
<i>Number of routers with Butterfly topology</i>	0	8	24	64	160

Table 2-2 shows also that exploiting the diverse parallelism levels of turbo decoding induces a reasonable overall area overhead (including memories, NoC, and ASIPs) while achieving outstanding throughput rates.

Thus, in order to meet flexibility and performance constraints of current and future digital communication applications, multiple ASIPs combined with dedicated communication and memory architectures are required. A flexible and scalable multiprocessor platform based on a novel ASIP

architecture for high throughput turbo decoding is proposed. The multiprocessor platform has been derived from a three-level classification of turbo decoding parallelism techniques. The proposed platform supports turbo codes of all standards. Results obtained for double binary codes demonstrate a 270 Mbit/s throughput using 16-ASIP architecture.

2.3. State of the art implementation of NoCs / MP-SOCs based channel decoders

The implementation of NoC has to face many challenges. Among them, the most important are:

- GALS features: how to assure that the interconnections can be realised at the assembling state starting from a set of IP cores
- Low-power : how to minimize the cost of the interconnection in term of power consumption
- Test&Debug : how to provide the test and debug features to deep embedded cores.

Three solutions exist for the GALS design of NoC, with the following advantages/drawbacks:

- Bi-synchronous Gray FIFO based [Che00, Cha03, Bei06]
 - ✓ Simple solution,
 - ✓ no additional cells
 - ✓ high throughput
 - ✗ area cost
 - ✗ power consumption
- Pausable (or stretchable) clocks [Yun96, Mut00]
 - ✓ Low area overhead
 - ✓ Low consumption
 - ✓ Adaptable to DFS
 - ✗ Need local clock generator & specialized cells
 - ✗ Throughput lowered
- Boundary Synchronization (mesochronous) [Dob04, Bje05]
 - ✓ Low area overhead
 - ✓ Power consumption
 - ✗ Verification
 - ✗ Throughput
 - ✗ Latency

Several research activities are currently targeting fine-grain NoC-based MP-SoC architectures for each of the building blocks of a wireless communication system. Channel decoding is one of the most explored blocks due to the underlying extensive computation and communication of iterative decoding techniques (Turbo decoding and LDPC) and the severe requirements in terms of throughput and flexibility. In parallel turbo decoding, extrinsic information is iteratively and concurrently exchanged between two component decoders. Furthermore, these exchanges become more and more massive with decoder level parallelism and the number of iterations. Since turbo code interleaving rule varies from one standard to another and/or one mode to another; the requirement of a fully flexible on-chip communication network interconnecting the two component decoders implies its ability to support the intensive interleaved memory accesses induced by parallel turbo decoders and to convey any permutation from the network input ports to its output ports. To that purpose, several application-specific on-chip communication networks were recently introduced. Topologies based on Benes, Butterfly, 2D Mesh, chordal ring, and de Bruijn [Mou08] were explored, designed, and integrated for an MP-SoC decoder implementation.

Regarding LDPC parallel decoders, same approach is investigated. In [Qua06], application of the permutation network scheme of Tarable with a modified version of the belief propagation algorithm is proposed. Moreover, a Benes network replaces the two original crossbars used for the permutation network. Another solution is proposed based on an heterogeneous network (MDN) [Kie03] whose topology is a randomly generated graph with non-regular node degrees.

Recent research activities in this context target the design of MP-SoC platforms for flexible Turbo/LDPC decoders and first results start appearing in this year [Mou08].

2.3.1. *IntraIP NoC for a flexible LDPC decoder*

As previously presented in subsection 2.1.2 we feel that the use of IntraIP ASNOC [Ben06] represents the most desirable solution for the design of a flexible channel decoder.

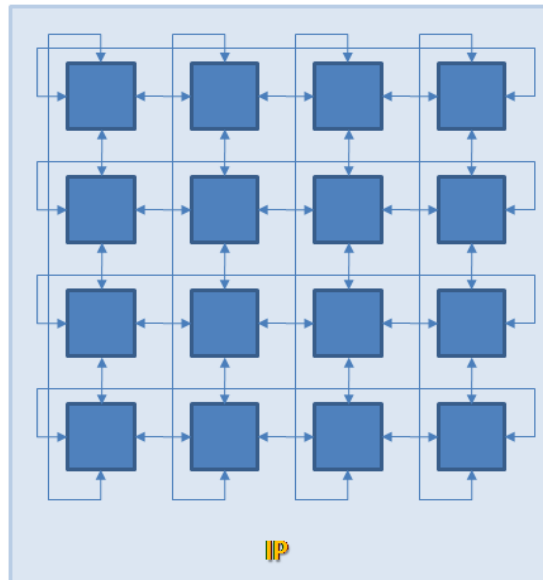


Figure 2-8: torus NoC topology

Hence the idea is to exploit network's scalability figures as much as possible to reach the desired flexibility degree. On the other hand the features of other NoCs, such as the possibility of conceal latency effects, their capability of connecting heterogeneous IPs, their simpler physical design due to their regularity, etc, are not leveraged. In Figure 2-8 the case of a 4x4 Torus NoC is displayed. Each PE is able to implement both the CN functionalities and the VN ones.

As a proof of concept a simple Intra-IP ASNOC was developed. It is suitable for LDPC decoding. The developed network is based on a 4x4 Torus topology (as in Figure 2-8) and each PE can perform both the VN and the CN functionalities. As far as the PE is concerned the current implementation relies on a dedicated VHDL implementation which employs the well known *min-sum* approximation for the

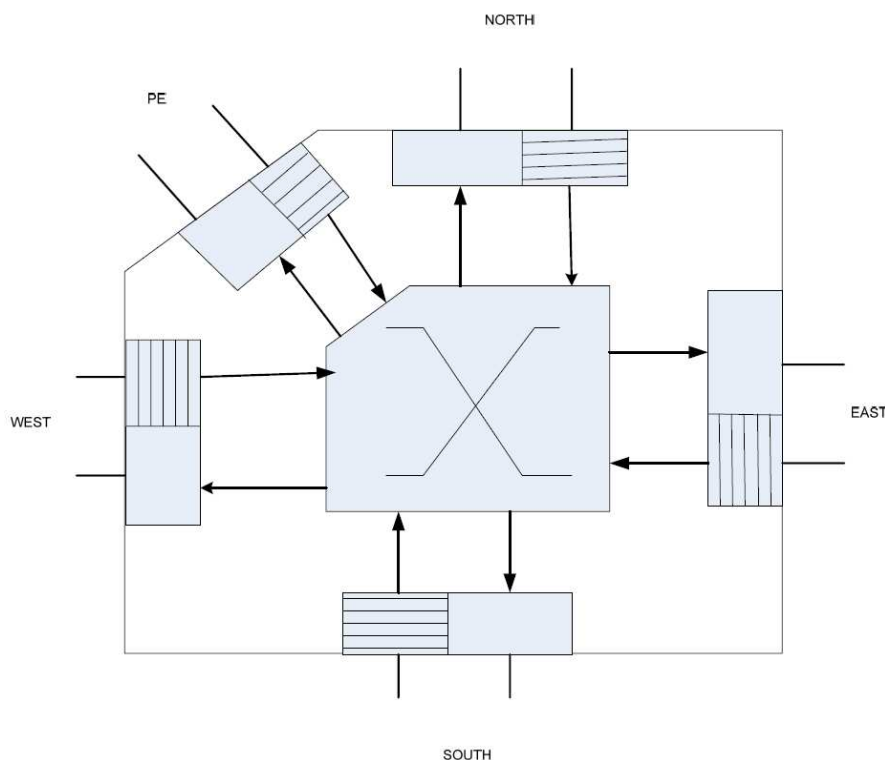


Figure 2-9: Processing Element architecture

Check Nodes. Exchanged messages are represented on 8 bits and it is supposed that each interconnection link that connects two nodes is composed by the same number of lines. In Figure 2-9 the architecture of a single PE is depicted. Roughly speaking, it is composed by five configurable FIFOs and a 5x5 non-blocking switching element.

Besides the NoC taxonomy, the main contributions of this work are the static, pre-computed routing and the job allocation on the processor array. As the former is concerned, it was observed that in general-purpose NoC [Ben06] a significant portion of the bandwidth is dedicated to control purposes, either for congestion control or for addressing functions. Since ASNoC is considered in this case, this assumption can be removed without loss of generality. By means of a proprietary tool the network evolution can be simulated at a cycle-accurate resolution: hence, at any given time it is possible to completely determine the allocation and the switching of each node. As routing strategy the so-called O1Turn as described in [Dae05] is used.

The routing logic of each PE is then replaced by a routing memory where the pre-computed node information is stored. It is important to stress how this approach retains the same flexibility of a general purpose network since if a different LDPC code is meant to be used, all is needed is to re-load the routing memory of each node. Conversely, this approach enable us to exploit all the available bandwidth without any penalty associated to control signals; additionally this makes possible to optimize FIFOs depths around the application since their sizes depend only on routing decisions.

As previously stated, the other important contribution of this work is the job allocation strategy among the 16 PEs. Here the idea is as follows: since in the goal is to minimize the communication latency, the optimal job allocation is the one which minimizes the total number of messages to be exchanged. This allocation can be easily found if one uses the Tanner graph formalism: then on Tanner graph, graph partitioning techniques can be used [Kar] in order to minimize the cut-set cardinality.

To prove the effectiveness of the presented ideas a dedicated Python tool has been developed able to map any LDPC code (specified through the ALIST format [McKay]) on a generic NxN Torus ASNoC. The tool schedules the message transmission on the network measuring the overall latency, the

obtained throughput and FIFOs length. The tool also automatically generates a synthesizable VHDL description of the whole network together with the routing memory content of each node. As a case study the biggest LDPC code from IEEE 802.16e was considered and mapped this (2304, 1152) code on a 4x4 torus ASNoC using the aforementioned methodology.

From this experimental work it has been possible to observe that:

- From the graph partitioning step a total of 3017 messages need to be delivered through the network every half-iteration (use a TPMP scheduling is used);
- From the ASNoC scheduling and simulation 251 cycles are required for the VN-CN phase and 234 cycles for the CN-VN phase. This is equivalent to a network utilization of almost 78%;
- Using a fully pipelined PE as the computation node 216 cycles are needed to compute all the VN and CN messages;
- The obtained structure has been synthesized on a Xilinx Virtex 5 XC5VFX30, obtaining a clock frequency of 300 MHz after the place and route step;
- Calling D the cycles per iteration I the number of iterations per block, f_{CK} the clock frequency and k the information bits. Then the net throughput is equal to:

$$T = \frac{k f_{ck}}{D I} = \frac{1152 \cdot 300 \cdot 10^6}{701 \cdot 10} \cong 49 \text{ MBps}$$

From these considerations it turns out that it is possible to achieve a significant throughput just resorting to a small-sized ASNoC. Additionally we feel that a high NoC utilization leads to high decoding throughput: this figure justifies the use of an Application Specific NoC tailored for LDPC decoding.

2.3.2. Binary de Bruijn NoC for a flexible LDPC/turbo decoder

In the context of flexible channel decoders and NoC-based implementations, [Mou08] proposes the first novel on-chip interconnection network adapted to a flexible multiprocessor LDPC/turbo decoder and based on the de Bruijn network. The main characteristics of this network –including its logarithmic diameter, scalable aggregate bandwidth, and optimized routing technique– allow it to efficiently support the communication-intensive nature of the two decoding techniques.

2.3.2.1. Communication Requirements of Flexible Parallel Iterative Channel Decoder

In parallel turbo decoding, extrinsic information is iteratively and concurrently exchanged between component decoders. For each iteration, these exchanges become more and more massive with decoder level parallelism. In addition, the communication traffic profile, which depends on the turbo code interleaving rule, can be considered as random if flexibility is targeted in order to support multi-mode and multi-standard features. The two component decoders in Figure 2-10 concurrently process the iterated frame, and for both, multiple processors can be designed for the parallel processing of frame sub-blocks.

Regarding LDPC, the architecture of a decoder (Figure 2-10) can be defined as being a system made up of variable nodes processors (VNPs), check nodes processors (CNPs), and an interconnection structure allowing the exchange of messages between processors. Mapping the Tanner graph onto this architecture involves: (1) associating one or more VN (resp. CN) to one VNP (resp. CNP) and (2) implementing Tanner graph edges through a corresponding message addressing in the interconnection structure.

However, similarly to parallel turbo decoders, partially-parallel code-independent LDPC decoders, which offer the best compromise in terms of area, throughput, and flexibility [Mas07], introduce a problem of conflicting accesses to the memories containing the exchanged messages. To manage this issue, the authors in [Tar04] propose an efficient algorithm which can compute a collision-free memory mapping of interleaving laws with no constraint imposed on the code itself. Thus, a versatile implementation requires for each supported code a pre-processing step to recompute the corresponding memory mapping. Besides, considering the various block sizes and codes rates increase significantly the flexibility requirement.

In fact, the computations being relatively simple, implementation issues mainly come from the communication structure between the VNPs and CNPs. Indeed, the communications rapidly become intensive as they depend on the number of nodes, the node degrees and the number of iterations.

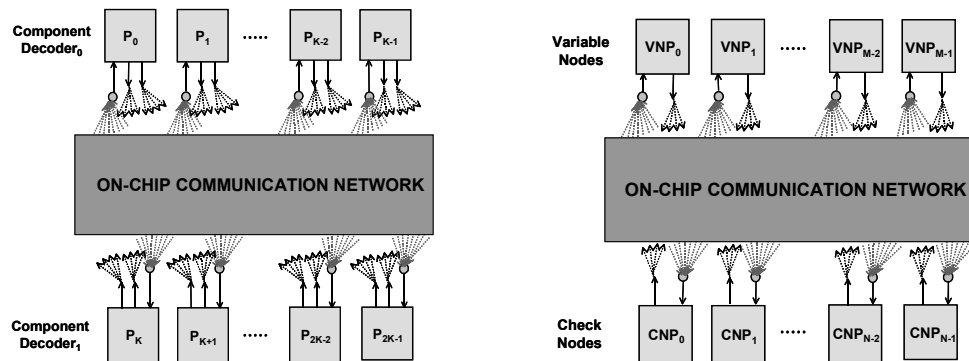


Figure 2-10: Extrinsic information exchanges in parallel turbo decoder and partially-parallel LDPC decoder

2.3.2.2. Binary de Bruijn Network

The de Bruijn network, based on the de Bruijn graph [Bru46], was chosen due to its appropriateness to support the communications of a multiprocessor LDPC/turbo decoder. In fact, the structure of the network allows any permutation to be routed efficiently thanks to the path diversity offered by the network. The conflicting memory accesses are alleviated because the conflicting packets can be deviated appropriately until they attain the targeted processor rather than being blocked or buffered. In addition, the logarithmic diameter of the network ($\log_2(N)$) leads to reduced latencies and the number of routers is reduced because it is a direct network. Moreover, its recursive structure makes it highly scalable.

The de Bruijn graph consists of $N=k^m$ nodes, where each node is identified by a vector of size m in the k -ary number system. For this network, the binary system was chosen. Thus, a node $v_m v_{m-1} \dots v_1$ has its two output edges connected with nodes $v_{m-1} v_{m-2} \dots v_1 \mathbf{0}$ and $v_{m-1} v_{m-2} \dots v_1 \mathbf{1}$. As an example, Figure 2-11 presents the schematic view of a complete de Bruijn network for $N=8$ processors ($m=3$). Each processor is connected to a network interface and each network interface connected to a router. Concerning the mapping of the decoding elements onto the network graph, it was decided to simply allocate the first $N/2$ processors to the first component decoder (in the case of turbo codes) or to the VNPs (in the case of LDPC codes).

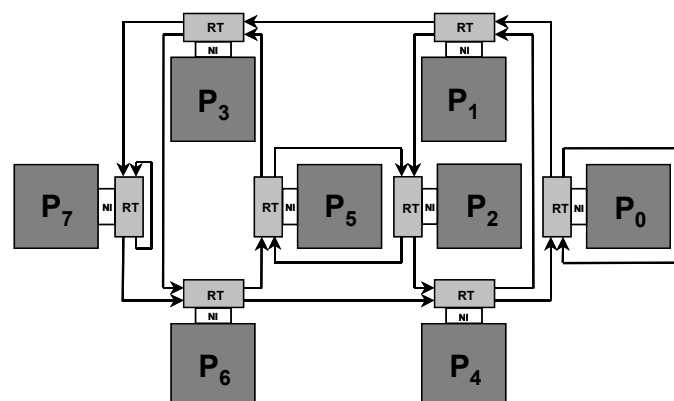


Figure 2-11: Schematic view of an 8-processor network with processors, routers, and network interfaces

The shortest path routing algorithm is implemented thanks to its ability to reduce latencies. However, the algorithm being sequential and in order to bring the number of cycles back to one, the algorithm of [Fra06] was modified by unrolling the “for” loop, and developed another method to compute the shortest path output. In order to reduce the size of routers by avoiding packet buffering when packets

are in conflict, the routing algorithm was supplemented by a mechanism deflecting the “youngest” packet. This requires storing “age” information (Time Stamp TS) within the packet updated at each hop.

Packets are divided into two parts: the header and the payload. The first one is composed of the destination router identifier, the TS value, the priority field and a one-bit flag indicating whether intra symbol permutation, as defined in certain standards like WiMAX, is applied to this packet. The payload is made up of the destination memory write address and of the extrinsic information.

The router architecture is divided into two units: operative and control units. The first one is composed of flip-flops, registers, and multiplexers implementing the packet switching and selecting the packets coming from routers and those coming from the processor. The control unit generates the multiplexer selection signals as well as the validity signals according to the routing algorithm and conflicting packet arbitration. Each port of a de Bruijn router has an associated control signal used to validate the data passing through the port. Moreover, there are two output signals corresponding to the read request of the emission FIFOs contained in the network interface.

The network interface manages the packet flow in both directions, from the processor to the network and vice versa. The interface contains a reconfigurable table (RAM) for the construction of the packet header according to the parity check matrix of an LDPC code or according to the interleaving rule of a turbo code. Indeed, the permutation of the extrinsic information is done via the addressing of the packets. Thus, the de Bruijn network can support the LDPC/turbo codes of wireless standards like WiMAX by simply programming the content of the table accordingly.

2.3.2.3. Synthesis and Results Analysis

For performance evaluation, a description in generic RTL VHDL of the complete system was developed and synthesized. The case study was the WiMAX standard for turbo decoding with a network of 16 processors. About LDPC decoding, a 1024 bits code also mapped onto the same network was considered. Thus, to estimate the area consumption and the aggregate bandwidth, the ST CMOS 0.18 μm technology is used. For each network the topology is provided, the target technology, the maximum clock frequency in MHz, the area in mm^2 , and the estimated aggregate bandwidth in Gbps.

In Table 2-2, the results obtained with two networks recently proposed and based on the Butterfly and Beneš 2N-N topologies [Mou07] are compared.

Table 2-3: ASIC Synthesis Results of Butterfly, Beneš 2N-N and de Bruijn Networks for a 16-Processor Turbo Decoder

Topology	[Mou07]		Proposed
	Butterfly	Beneš 2N-N	de Bruijn
Technology (μm)	0.18		0.18
Frequency (MHz)	302	416	266
Network Area (mm^2)	4.834	2.692	3.565
Aggregate Bw. (Gbps)	308	426	592

As the de Bruijn routing is more complex (shortest path computation + deflection) than those of Butterfly and Beneš 2N-N networks, the resulting critical path is longer, and thus the maximum clock frequency is lower. On the other hand, the de Bruijn area cost is in between because at the same time, the network interfaces are bigger (due to the routing table) and the routers are smaller and fewer. Finally, the de Bruijn network offers the best aggregate bandwidth.

Next, the results obtained with other previously proposed flexible LDPC decoder architectures (Table 2-3) are explained for a given codeword size of 1024 bits and based on an interconnection network. Although flexible, the Beneš network requires a pre-calculation related to the code to configure the switches. The MDN uses a general approach to network design with random graphs as basic topologies. This network lacks flexibility because all permutation patterns must be simulated prior to their use in order to determine buffer size at design-time. Finally, the 2D-mesh network is designed for

a flexible decoder. However, the described network can only manage codes with a maximum codeword size of 1024 bits. Consequently, the de Bruijn network, allowing to support any LDPC code, constitutes the most flexible solution. For scalability, the recursive structure of the Beneš, the 2D-mesh, and the de Bruijn networks makes them highly scalable. However, the MDN network lacks scalability because the network structure is randomly built by a complex algorithm based on a constructive heuristic. Frequency, area, and aggregate bandwidth being related to the target technology, it is not possible to compare directly and accurately the values of Table 2-4. However, it can be observed that the de Bruijn network presents the best aggregate bandwidth with reasonable area cost.

Table 2-4: Comparison ASIC Synthesis Results of LDPC Decoders

	[The05]	[Kie03]	[Mas07]	[Mou08]
Topology	2D-mesh	MDN	Beneš	de Bruijn
Technology (μm)	0.16	0.18	0.13	0.18
Frequency (MHz)	500	200	245	266
Network Area (mm^2)	25	7.92	0.98	3.565
Aggregate Bw. (Gbps)	400	19.2	62.72	592

Thus, the unique trade-off between flexibility and performance (scalability, frequency, area) offered by the de Bruijn network demonstrates its efficiency in the context of a flexible parallel channel decoder supporting LDPC and turbo codes.

3. MULTI-STANDARD PROCESSING FOR COGNITIVE RADIO

The evolution of wireless communication systems over the last years has followed a clear trend towards 4G. 4G systems are sometimes seen as the next generation of cellular networks, but it is often widely understood as a collection of wireless technologies that will coexist in a wireless ecosystem.. Several standards can be mentioned as parts of this ecosystem. UMTS, WIFI, WIMAX are already examples of standards which nowadays' terminals have to consider. In this context, future radio transceivers will have to provide a higher level of flexibility to support the integration of the different Radio Access Technologies (RATs).

Functions with relaxed real-time constraints are already implemented in software, leading to higher flexibility and easier maintenance. However a significant amount of baseband functions have to be implemented in hardware to cope with demanding processing along with strong real-time constraints. This is caused by the fact that the algorithmic complexity of wireless systems has grown faster than the silicon technology capacity. Indeed, demands for higher throughput and higher spectral efficiency have pushed digital communication to provide complex schemes reaching close to the capacity limit performance. Some specific aspects of the digital communication processing impacts the way digital signal processing has to be considered:

- The clearly separated functions in the processing chain that can be translated in almost independent process with highly different computing requirements. (Opportunities for parallel and heterogeneous processing).
- The radio time-slot division of the transmission medium that implies some computing time-slot-based execution of the different modules. (Slot based or symbol based synchronicity).
- The need to continuously maintain the transmission and reception processes active, while being aware of their real-time constraints. (Data flow oriented processing).
- The definition of RAT-specific quality of service target that could influence the computing requirements. (Impact on operand quantization, sampling rate, etc).
- The different computing constraints of different RATs and also their dependence from the current radio conditions. (Flexibility of the design to adapt to several operational modes).
- The need of supporting the dynamic partial or total reconfiguration of part of the processing chain as the most suitable mechanism to ensure flexibility. (Local embedded programmable logic).

As mentioned before the wireless processing chain can be divided in well identified functions or blocks with clearly defined functionalities. Such blocks, and their functionalities, can be grouped into operational segments. The most relevant from the point of view of the processing requirements can be the source, the bit level processing, the baseband processing and the digital IF processing. In the source segment could be included the source coding done at the proper codec, whereas the bitstream level processing assumes functionalities like Cyclic Redundancy Check (CRC) coding, convolutional block and convolutional Forward Error Correction (FEC) coding, and more advanced turbo FEC coding, the ARQ schemes, the multiplexing of different data users, the interleaving, the frame synchronization, encryption, etc.

Classifying the blocks that way clearly highlights different rate (bit, sample, symbol, frame) and several data quantization (bit, soft-bits, words). This classification rules some of the choices of the designer. For instance, functions related to coding/decoding functionalities are generally based on a linear shift register structure, or other computation involving single-bit processing. Software implementations are feasible, but hardware implementations are more power-efficient and less computing resources greedy. Since the word length is only one bit for this class of functions, software implementations are generally under-optimized, since they mostly rely on 16/32 bit processors. However, the wide variety of coding schemes requires enough flexibility, which is easier to obtain through a software implementation. Hence, a software implementation, accelerated through power-efficient dedicated co-processors could correspond to a suitable architecture for this class of processing functions.

Another example is the functions related with the data handling and sorting, like interleaving, which, perform different types of manipulation of bits and data packets. These blocks are mainly ruled by memory access, pointer handling, branches and, in a wider sense, control flow. In this case, the high flexibility offered by general purpose processors or DSP make them the most appropriate solution for such functions.

The baseband processing considers the functions performed just before upconversion to the carrier frequency and the digital IF includes the mentioned up/down conversion, the digital filtering and the sampling rate adjustment. The last two classes of functions are rather demanding in terms of computation. Furthermore, they often process oversampled data with N-bits length data words which consumes a lot of computing resources. Consequently, some of these functions can take advantage of an implementation in specific processors or in dedicated accelerators.

Two examples of processing chains can be found hereafter: a DCH UMTS channel and an OFDM system. Figure 3-1: Processing chain of a DCH UMTS channel . depicts the functional diagram of the digital signal processing chain at the physical layer of a software-defined UMTS downlink receiver. It comprises 24 SDR functions from the digital down conversion to the Cyclic Redundancy Check (CRC) [Tan04]. The computing requirements are estimates from [Tan04, 3GPP05, Fab99] and available implementations using the TMS320C6416 DSP and the Code Composer Studio from Texas Instruments [TI]. In particular, the number of MACs of an SDR function's implementation times the sampling rate f_s specifies the function's processing requirement. A bandwidth requirement, to define the data flow among modules, is the product between the f_s and the bit precision of $2 \cdot 16$ bits for the real and the imaginary components.

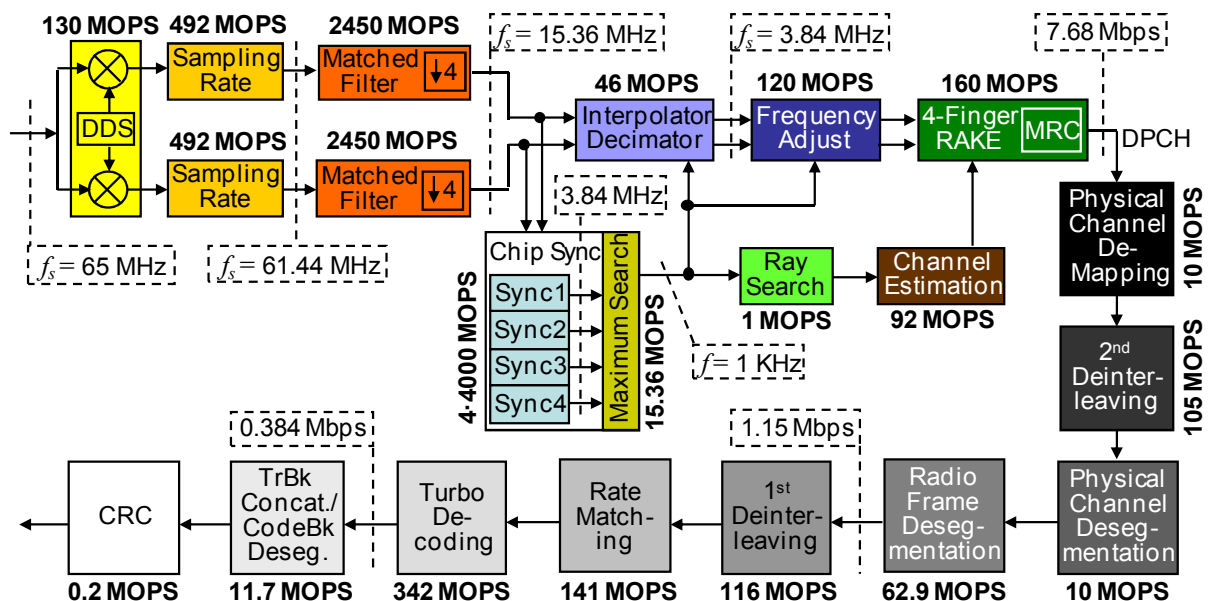


Figure 3-1: Processing chain of a DCH UMTS channel .

The second example is a typical OFDM transceiver consisting of blocks performing standard operations for transmission and receive operation can be seen in Figure 3-2.

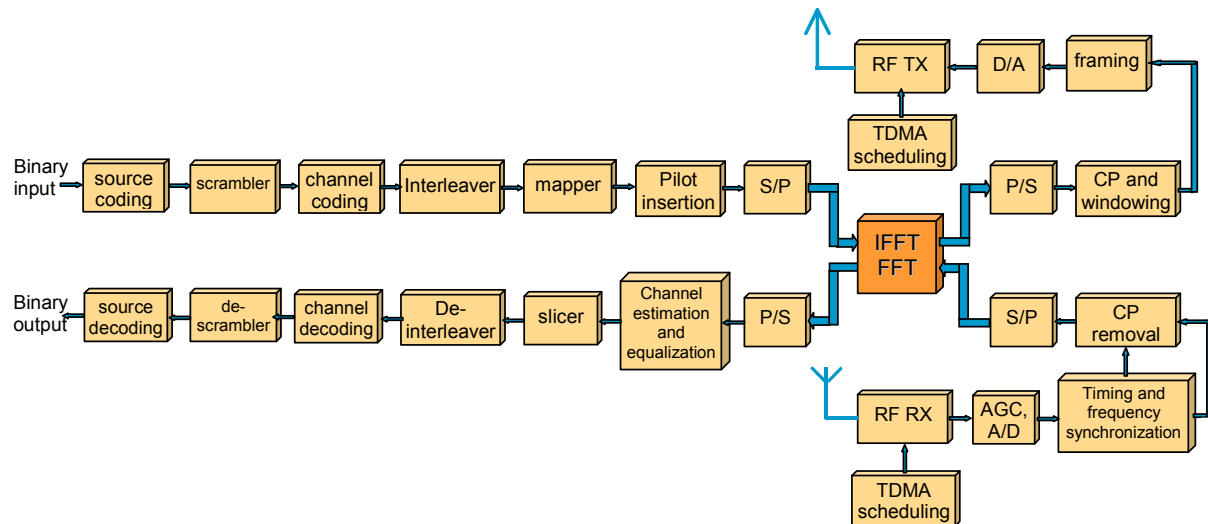


Figure 3-2: Typical OFDM transceiver.

Hence, it can be noticed that the modules or blocks that form a wireless processing chain can be grouped by classifying the type of computing resources required to execute them, the level of flexibility, to allow changes from one operating mode to another one in the same RAT standard or changing from one RAT to another, and the quantity of computing resources required. Therefore, from this classification, the design space can be tackled bearing in mind function based requirements, with a goal of providing a good compromise between the flexibility and the power consumption to afford the challenges of wireless computing platforms. Then, it is widely assumed that the different processing tasks should be grouped by the characteristics of the operations required in order to assign them to a specific approach. Nevertheless, such task pre-assignment, decided at the design time, produces in fact a loss of flexibility while improving the efficiency in power consumption. The selection of a good trade-off between both objectives is one of the most difficult tasks of the digital designer due to the fact that it is highly dependent on the state of the art of computing options, technology dependant factors and wireless technology requirement and evolution.

In the following several approaches to achieve flexible digital radio are addressed and actual hardware platforms are discussed. The flexibility of the radio-frequency parts of the transceiver, although being an active research topic, is not addressed in this document and is out of the scope of the competency of WPRC. The reader can find relevant discussion in [Ora3.3, Ora3.4].

3.1. Software Defined Radio and parametrisation techniques

The term Software Radio (SWR) was first coined around 1990, thanks to the pioneering works of J. Mitola [Mit95] and W. Tuttlebee [Tut95]. But it should be noted that this approach was also a logical consequence of the DSP's performance enhancement, and was in line with the historical trend to extend the functional perimeter of software based devices.

SWR basically refers to a set of techniques that enables the reconfiguration of a communication system without the need to change any hardware system element. SWR offers many advantages compared to its hardware based counterparts for all the technology stakeholders including designers, manufacturers, operators and service providers. One of the benefits of SWR is that it considerably enhances communication devices flexibility by, for instance, offering the ability to support several air interfaces or configurations.

Although the goal of having a purely software-based radio is still critical due to real-time processing and power consumption constraints, it can be observed that modern transceivers available on the market are more and more Software Radio powered. Thus, the community adopted the term Software Defined Radio (SDR) instead of SWR, considering that the SWR concept is not 100% fulfilled and that some part of the processing is not done in pure software. This can be concluded by considering the regularly updated surveys by the SDR Forum [www.sdrforum.org]. The versatility of such

platforms is also intrinsic. Indeed, any reconfiguration simply corresponds to a change in their software. In fact the required software does not even need to be stored in the device itself, since it can be downloaded from the network.

An illustration of this concept is given in Figure 3-3, in which an analogy between the “PC world” and the SWR transceivers can be drawn. We can summarize this analogy by saying that “the aim is to perform many different applications using the most common tools possible with a real-time operating system running on a minimum hardware”.

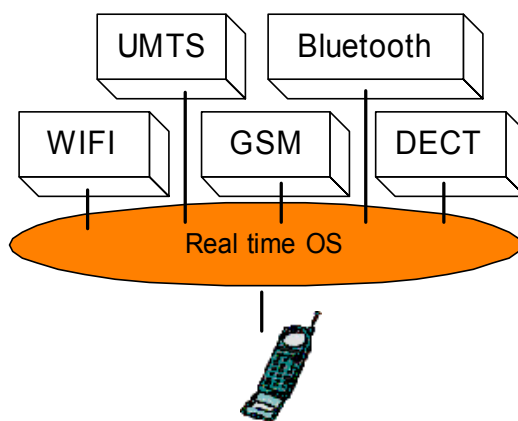


Figure 3-3: universal receiver

3.1.1. The techniques of parameterization : Introduction

Parameterization is an important aspect of the SWR concept. It consists in identifying all the common aspects of the mobile communication standards with which the receiver is intended to communicate. In addition heterogeneous wireless systems such as GSM¹, IS95², PDC³, DECT⁴ and UMTS⁵ are growing up every day as well as wireless LAN⁶ and Hiperlan⁷, or broadcasting technology like DAB⁸ and DVB-T⁹.

The main drawback of this profusion of technologies is that end users need more and more radios. This explains the interest in multi-mode terminals enabled thanks to SWR techniques. In this context, parameterization can be considered as an optimization process which can help in the design of reconfigurable equipments.

3.1.2. The techniques of parameterization : Different Aspects

According to the parameterization methodology, the common aspects of the different standards will become one common processing procedure, which could be installed in the device. This common procedure could be executed by a simple “call”, thereby resulting in a gain of both size and time with regards the amount of code to be downloaded or read in order to modify the radio behavior.

From the design point of view, the parameterization will optimize the co-design (CAD-friendly approach) and will probably reduce the time to market.

¹ GSM: Global System for Mobile Communications

² IS95: Interim Standard 95 (CDMA)

³ PDC: Personal Digital Cellular

⁴ DECT: Digital Enhanced Cordless Telecommunications

⁵ UMTS: Universal Mobile Telecommunication System

⁶ LAN: Local Area Network

⁷ Hiperlan: High Performance Radio Local Area Network

⁸ DAB: Digital Audio Broadcasting

⁹ DVB-T: Digital Video Broadcasting-Terrestrial

The techniques of Parameterization can be studied according two distinct axes: first, the common function technique proposed in [Mit95] and also in [Tut95], and second, the CEA-SUPELEC developed common operator technique [Pal03].

The Common Function (CF) technique was historically the first one proposed by several articles from Karlsruhe University (Germany) [Jon02], [Rhi02]. [Pal03] proposed a common basic function: the Fast Fourier Transform (FFT) in 2003. This paper was the starting point of our Common Operator (CO) technique.

The two methods, which will be now described, have the same goal: to reach a representation of the reconfigurable architecture, which becomes optimal.

These two technical methods are very close. As presented in Figure 3-4, depending on the point of view, either the Common Function (CF) technique or the Common Operator (CO) technique are addressed, according to the following two definitions.

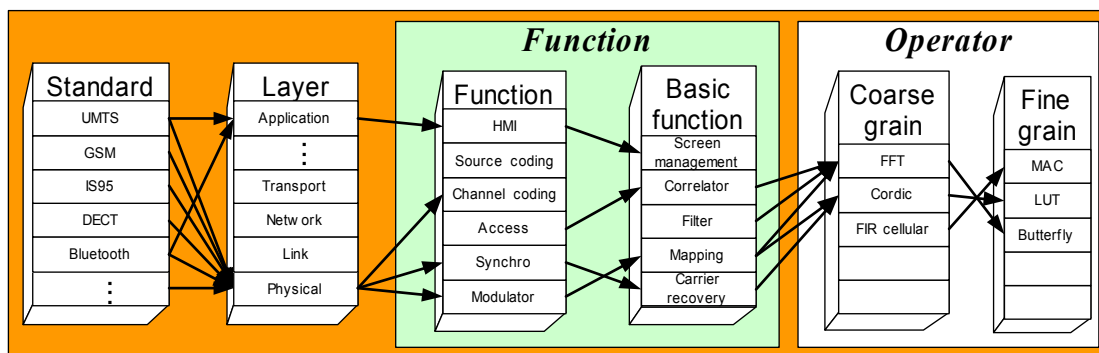


Figure 3-4 : Two Techniques for parameterization

In practical terms, the Common Functions Technique consists in seeking an optimized generic function which can replace the initial task present in a predefined set of standards.

The Common Operator technique claims to be independent of the standards by finding the smallest set of highest-level operators, which are used by the maximum functions number.

3.1.3. The techniques of parameterization : Common Functions

With the CF, the common feature of different standards will become one common processing procedure, which could be stored in the device. For the download aspect this common procedure could be executed by a simple “call”, therefore resulting in a gain of both size and time in the downloading operation.

One example is given in [Jon02] the “Generalized parameterizable modulator”. This example is a general modulator for the GMSK, QPSK, $\pi/4$ QPSK and dual QPSK modulations for the air interfaces of the following standards: GSM (FDMA/TDMA, GMSK), IS 136 (DAMPS) (FDMA/TDMA, $\pi/4$ QPSK) and UTRA FDD (DS-CDMA, QPSK). A careful study of this example shows that it is more an assembly of several modulators, with some specific (and small) common procedures. In fact, the parameter used by the CF offers the choice between several paths depending on the standard.

Another example of the parameterization study, for UMTS, GSM and PMR¹⁰, concerning channel coding aspects is given in [Rhi02]. In this paper the author highlights the fact that parameterization makes it possible to build communications systems with flexible components, under the restrictive assumption that these components belong to a predefined set of transmission modes.

¹⁰ PMR : Professional Mobile Radio

3.1.4. *The techniques of parameterization : Common Operators*

The Common Operator technique was specially developed by CEA-LETI and SUPELEC. As seen before, an operator is a component, which can be used to perform other operations without knowing the content. This means that an operator can use other operators (which, are generally hidden and depend on the hardware itself). This operator has been completely and successfully checked, optimized and tested. The higher the level of the operator, the shorter the development loop will be. Therefore, this CO technique will decrease the development time and will also decrease the time to market.

3.1.5. *The techniques of parameterization : Contribution to NEWCOM⁺⁺*

We identified and developed several operators based on FFT-butterfly and Linear Feedback Shift Register (LFSR) Structure.

Actually, it has been observed that almost all the Software Radio Receiver functions could be performed using the FFT function. In addition, the future modulations studied for the next generation wireless systems are very often performed through modulated filter banks as it is the case for the BFDM [Sic02], and also partly for the MC-CDMA System [Hel01], which combines OFDM with CDMA. These modulation evolutions will clearly benefit from “Frequency Domain” implementation.

In addition, different LFSRs, presented in [Ala08a][Ala08b], have been identify as a common operator for some cyclic block channel operations. Is has been shown that for the PN sequences, scrambling, convolutional coding or CRC, the LFSR structure could be applied.

Finally, it can be concluded that in the context of parameterized Software Radio, short FFT and R-LFSR are good candidates for becoming basic operators for almost all the transceiver’s functions. The next step is to confirm this conclusion by using an SDR receiver implementation to quantify the gain offered by these operators.

3.2. Resource management in multi-mode radios

3.2.1. *Computing Resource Management*

An intelligent (radio) resource management system is necessary to fully exploit the heterogeneous and dynamic wireless access networks of today and tomorrow. Such a system should observe the environment and react intelligently upon its changes. In order to achieve maximum flexibility of software-defined and cognitive radios, in particular, the system requires a management layer that accounts for all those resources that facilitate wireless communications. These include the radio resources but also the computing resources. Radio resource management is widely accepted as *the* research topic in B3G radio communications. This section therefore argues for the importance of computing resource management.

Computing resources include:

- processing powers and interprocessor bandwidths,
- memory resources, and
- energy resources.

These resources are and will be limited. Even though the computing power could grow faster than its requirement, related issues, such as energy consumption, will not become obsolete even in the far future of radio communications. It is therefore necessary to optimize the usage of radio applications computing resources to facilitate an anytime, anywhere, and anyhow wireless access.

Figure 3-5 shows the cognitive computing cycle. It continuously scans the computing environment and provides any changes to the computing resource management (CRM) entity, which decides upon an appropriate action. The CRM entity is the brain of the system; it reacts upon its inputs and its current knowledge. This knowledge is accumulated during the learning process of the CRM algorithm, which

observes the effects of its decisions (*learning-by-observing*). Hence, the cognitive computing cycles has to run continuously.

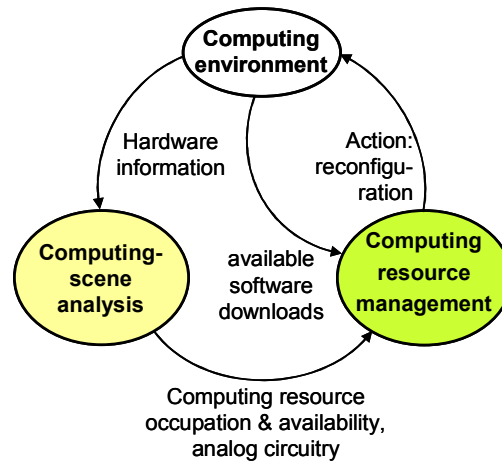


Figure 3-5. The cognitive computing cycle.

Let us consider a simple example. In a certain area two radio access technologies, RAT *A* and RAT *B*, are supported. If the radio resources of RAT *A* are overused, while those of RAT *B* are not (*radio scene analysis*) and if the network infrastructure, including software downloads, and mobile terminals facilitate a switch to RAT *B* (due to the computing scene analysis of Figure 3-5), the action could be to reconfigure some mobile terminals to access RAT *B*. The above discussion addresses the intelligent reconfiguration of flexible mobile terminals. It can, however, be easily extended to aid the reconfiguration of network elements as well. Although fixed network elements are less critical in terms of computing power, their intelligent management can greatly reduce power consumption and, thus, reduce the operational costs. We further expect that many cognitive computing cycles will be globally distributed and coordinated to intelligently use the available computing resources of flexible equipments and, thus, provide a ubiquitous wireless access where the user is unaware of the RAT that satisfies its service demand.

It remains to point out that computing resource management is important for efficiently using the limited computing resources. On the other hand, it needs to be coordinated with the radio resource management: Computing resource management should support radio resource management and vice versa. For example, sufficient available computing resource can execute heavy signal processing algorithms to be able to correctly receive signals that are transmitted at lower powers (transmission power is a radio resource). The ultimate goal is thus to dynamically trade computing for radio resources as a function of the time varying radio and computing environments.

3.3. State of the art implementation of multi-standards platforms

3.3.1. Low-power multi-standard NoC architecture

In this section, we present the backbone used to encapsulate computing resources for a Dynamic Voltage and Frequency Scaling (DVFS). The NoC is used as an interconnect medium between units and as an isolation layer allowing multiple voltages domains. As a consequence, the proposed DVFS architecture is implemented within a complex GALS NoC. In this section, the system architecture is briefly described to focus on the unit architecture handling the local power management scheme.

The complete NoC backbone has been used in the MAGALI chip, a flexible architecture for multi-standards 4G telecom applications.

3.3.1.1. System Architecture

Each units of the dedicated SoC are arranged around a fully asynchronous Network-on-Chip [Lat07, Bei06]. As shown Figure 3-6, the NoC units are fully synchronous islands. Synchronization between

the communication router and the units is done thanks to a pausable clock mechanism called SAS (containing Synchronous-to-Asynchronous and Asynchronous-to-Synchronous interfaces). A programmable Local Clock Generator (LCG), using a delay line, is implemented within each unit to generate a variable frequency in a predefined applicative range. The power unit manages the local unit voltage, sharing a power switch between a V_{DD} hopping technique and an ultra-cut-off block. The Power Unit uses two external voltages: V_{HIGH} and V_{LOW} to be automatically switched during DVS phases. The Network Interface (NI) is in charge of communications with respect to NoC protocol and is also in charge of local voltage and frequency control for DVFS using a Low Power Manager.

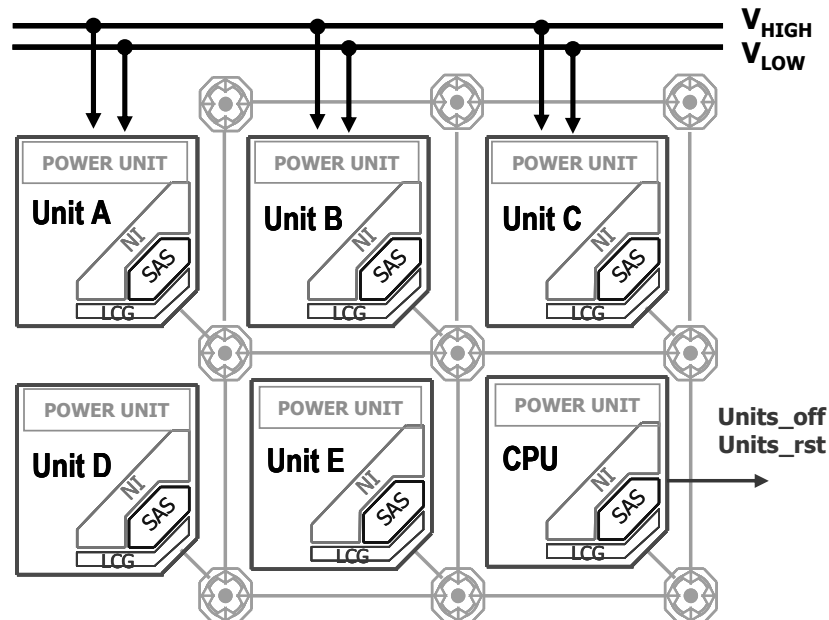


Figure 3-6 : DVFS NoC architecture

The power management strategy is programmed by the main CPU, through NoC unit attached Network Interface's Low Power Managers, according to required performance and power constraints. DVFS can be executed during IP computation and communication according to their own activity. The only global signals are regarding units' reset and off control. The main CPU is required to directly disable/enable the units for power off mode and the corresponding reset phase.

Main principles of the local control and power regulation are described thereafter.

3.3.1.2. Main principles

Each synchronous IP unit is defined as an independent power domain (using its dedicated local voltage) and an independent frequency domain (using its dedicated local clock). The unit handles a set of user-defined power modes (see section 2.2).

In order to perform efficient local DVS, the main objective is to ensure a minimal latency cost of DVS during unit's computing. Within the Power Unit, this leads to implement a hardware controller to automatically switch between V_{HIGH} and V_{LOW} . By guarantying smooth DVS transitions, the synchronous IP block can continue its own operation. To obtain an average voltage value between V_{HIGH} and V_{LOW} , the Low Power Manager automatically switches between these two values using a configurable duty-ratio.

Since the power domain and frequency domain of the synchronous unit are identical, it is expected that the local frequency scales approximately linearly with the associated voltage scaling. If the frequency/voltage scaling ratio is not exactly linear, the delay line must be re-programmed accordingly. The V_{LOW} delay value is adjusted by a correcting factor with respect to the V_{HIGH} delay value, since the delay line is supplied on the *same* power domain.

Finally, using Pausable Clock technique, fast and reliable delay line programming interface is obtained. As a consequence, the synchronous IP unit locally continues its own computations or NoC communications during any DVFS phases.

3.3.1.3. NoC unit architecture

NoC unit integration for DVFS

In order to integrate a synchronous unit within the proposed NoC DVFS scheme (Figure 3-7), each synchronous IP core (“core” level) is encapsulated within:

- Its own Network Interface (NI) and Low Power Manager (LPM). The Network Interface provides HW primitives for NoC packet generation & reception, and task synchronization [Lat07]. The LPM implements the local power mode defined for that given unit and controls respectively the Power Supply unit and the Delay line. This is the “unit” hierarchy level.
- Its own NoC Test Wrapper (NTW) for handling test-mode. For test, the NoC topology is used to carry the test patterns which are used to feed the synchronous scan chains through the NTW [Tra07]. This is the “test” hierarchy level.
- Its own Pausable Clock interface, which contains the SAS interface, the local pausable clock generator and a delay line programming interface. This is the “sas” hierarchy level.
- Its own Power Supply Unit (PSU), which contains Ultra-Cut-Off mechanism for leakage reduction, and Hopping Unit for DVFS, generates the V_{CORE} voltage. Lastly, some isolation cells (for LOW and OFF modes) are inserted on the external asynchronous 4-phase/4-rail NoC link for proper voltage conversion between the local unit and the NoC topology. This is the “top” hierarchy level.

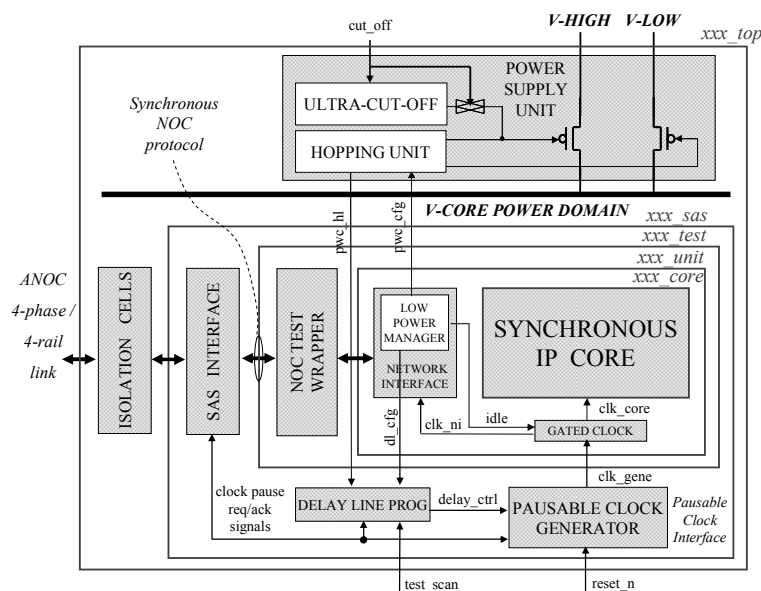


Figure 3-7 : NoC unit architecture

In the proposed scheme, the complete Synchronous IP core, up to the “sas” hierarchy level, is clocked by the locally generated clock, and supplied by the V_{CORE} voltage, locally generated by the PSU. The synchronous unit has thus its own frequency and power domain.

Power mode definition

Table 3-1: Definition of Unit Power Modes

INIT	At reset, the unit is at VHIGH with no clock
HIGH	The unit is supplied by VHIGH voltage
LOW	The unit is supplied by VLOW voltage
HOPPING	The unit is automatically switched between VHIGH and VLOW voltages, <i>for DVFS</i>
IDLE	The unit is idle, with maintain of its current state at VLOW voltage, <i>for reduced leakage power</i>
OFF	The unit is switched OFF, with no maintain of its current state, <i>for minimal leakage power</i>

In the proposed architecture, each unit can be set in one of the 6 available power modes (Table 3-1):

- in INIT mode, supply voltage is V_{HIGH} , and no clock is sent to the core. This is the “*post-reset*” mode.
- in HIGH mode, supply voltage is V_{HIGH} and core clock is on. This is the “*nominal*” working mode.
- in LOW mode, core clock is still on, but supply is switched to V_{LOW} . Clock frequency is lower than nominal, and energy per cycle decreases. This is the “*low power*” mode.
- in HOPPING mode, core clock is on and supply voltage automatically hops between V_{HIGH} and V_{LOW} . Frequency and duty-ratio of hopping transitions is configurable. The obtained performance is an intermediate between V_{HIGH} and V_{LOW} modes, depending on the given duty-ratio. This is the “*DVFS*” mode.
- in IDLE mode, core clock is off and leakage power is reduced thanks to the V_{LOW} supply voltage. This is the “*low-power dormant*” mode.
- in OFF mode, the unit is switched off by the Ultra Cut-Off device [Val07] to further reduce the leakage power. Since the NI is inactive in this mode, the OFF mode is enabled/disabled through an external “*cut_off*” signal controlled by another unit in the NoC (for instance the main host processor). This is the “*low-leakage*” mode.

All power modes, beside the OFF mode, can be selected per each unit through programming of the unit Network Interface.

3.3.2. Open platforms for digital communication systems

3.3.2.1. The Free Software Foundation GNU Radio

The FSF GNU Radio project is a community project developing a software defined radio platform for educational and research purposes. It is based on the assumption that contemporary PC hardware is sufficiently powerful to implement signal processing functions required rather than relying on dedicated DSP or FPGA based hardware preferred for commercial solutions. Currently, it is the most inexpensive way to explore the field of software defined radios, especially in conjunction with the Universal Software Radio Platform (USRP) developed by Mark Ettus (www.ettus.com). The project homepage is at <http://www.gnu.org/software/gnuradio/> and a concise overview of its history and goals can be found at http://en.wikipedia.org/wiki/Gnu_radio.

Recent GNU Radio versions rely on a mixture of Python code for configuration, graphical user interface and chaining of signal processing blocks as well as on C++ code for the implementation of signal processing blocks. The latter often utilizes x86 architecture specific single instruction multiple data stream (SIMD) for speed enhancement, but there are also activities to utilize dedicated processing hardware such as the IBM cell processor for this purpose.

A GNU Radio application usually consists of a Python main object that creates several source, sink or filter objects and chains them into a signal flow oriented “flow graph” representing the algorithm or system to realize. There are various objects available as a part of the standard GNU Radio distribution: source objects are, e.g., signal generators or files; filter objects are e.g., filters, transforms, modulators, sinks are, e.g., waveform or spectrum displays or disk files.

When a flow graph is run, it fetches blocks of signal data from an input stream, e.g., I/Q samples as complex floating point numbers, schedules this data through various signal processing blocks as defined by the flow graph and sends results towards one or more output streams.

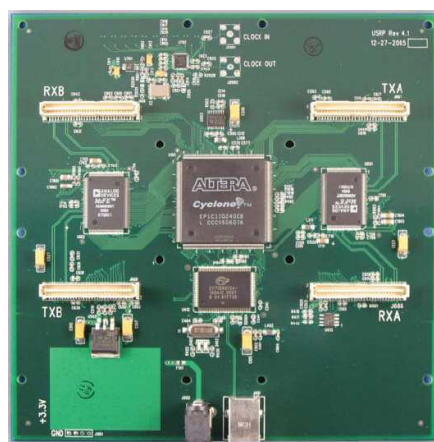
This flexible approach allows implementing rather complex systems just by chaining predefined signal processing blocks. Graphical utilities have been implemented by the developer community in Python to provide a convenient tool for creating flow graphs.

Additional signal processing blocks can be implemented in C++ and can interface with external hardware, with external software tools, or with OS-specific functions. Examples for these are the integration of external numerical processing hardware such as the TESLA/CUDA hardware, interfacing with MATLAB scripts and interfacing with protocol stack implementations. The latter has been used by BBN to implement the IEEE 802.11b PHY and lower MAC on the USRP and the upper MAC and higher layer protocols on a Linux platform by utilizing OS raw socket communication.

The USRP integrates a GNU Radio sink and a source block into the flow graph framework along with a number of Python-based configuration tools to set parameters as, for example, the RF center frequency, the baseband decimation factors, or the data format.

The USRP physically interfaces to an x86 host computer via a USB2 interface, which incorporates a serious limitation to the field of application for this hardware. The maximum sustained data rate achievable is 32 Mbytes/s limiting the baseband to 8 Msamples/s (complex I/Q samples with 16 bits resolution each). The USRP can provide 16 Msamples/s data acquisition only for a short time without producing buffer overflows. At the cost of limiting the resolution, the interface data format can be set to 8 bits per sample which unfortunately does not allow for higher sampling rates since decimation factors in up/down-conversion are limited to powers of 2 in the range of 4...256 by the USRP's FPGA firmware. These limitations have led to a re-design announced as USRP-2 for end of April 2008.

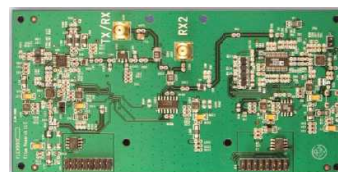
The USRP is a digital acquisition system providing a direct conversion between the RF domain and the baseband. This task is split between the USRP baseboard and up to two RF transceiver daughterboards or two transmitter and two receiver daughterboards (example shown in Figure 3-8). Various daughterboards are available covering the range between 0 Hz and 2.7 GHz.



USRP base board



TVRX daughterboard



RFX2400 Transceiver daughterboard

Figure 3-8 : USRP base board with example daughterboards

Each USRP has four high-speed analog to digital converters (ADCs), four high-speed digital to analog converters (DACs), USB interface chip, the Altera Cyclone EPIC12 FPGA as illustrated in Figure 3-9.

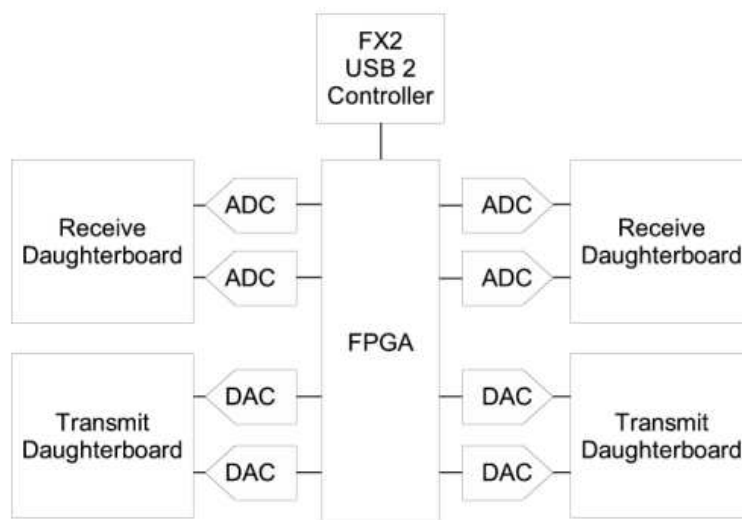


Figure 3-9 : USRP Architecture

In this architecture ADCs are sampling with 64 Msamples/s and 12bits resolution, whereas DACs are capable for 128 Msamples/s at 14bits resolution.

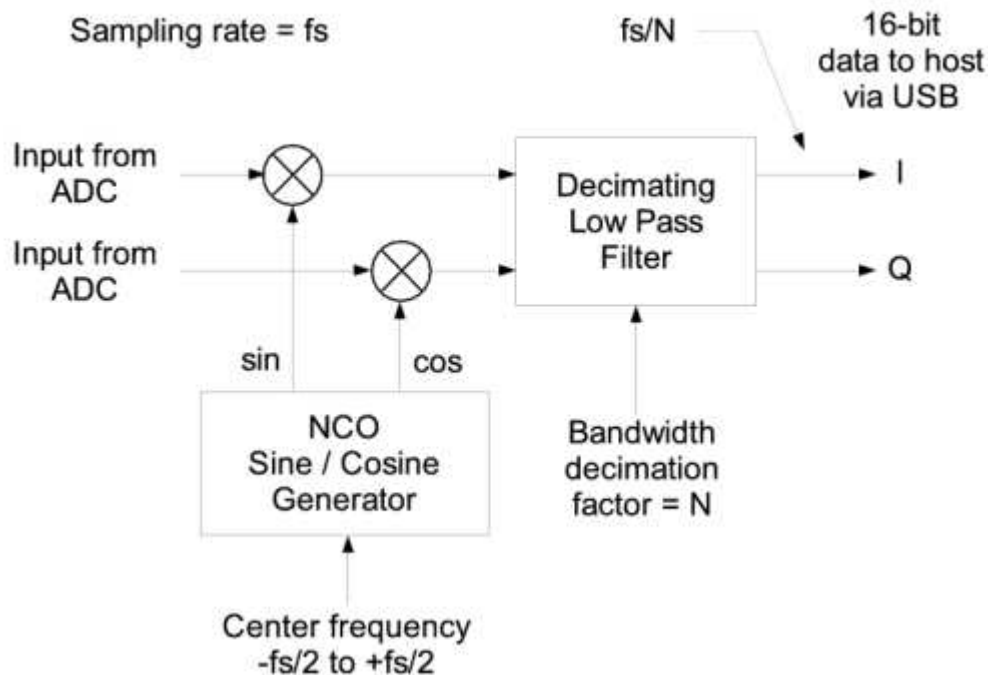


Figure 3-10 : USRP Receive Path Architecture

3.3.2.2. Commercially available platforms

This section does not aim at comparing available Commercial Off-The Shelf (COTS) platforms, since the role of WPRC is not to influence the reader's choice on any specific equipment. Most of the time the hardware platforms described have their specific pros and cons. Rather, this section aims at providing examples of platforms that may be used for communication transceiver implementation.

Hunt Engineering (<http://www.hunteng.co.uk>)

Hunt engineering provides a modular approach to real-time digital system implementation. The system is composed of a so called carrier board that hosts processing modules. A typical carrier board (eg.. HEPC9) can host up to 4 processing module and an extension module that enable multi carrier systems to be setup. A processing module consists of FPGA (Xilinx) or DSP (TI) nodes. Each module may also have embedded local memory and the processing core can be selected among a range of processing power capability. The nodes are connected by a ring bus referred to as the HEART bus, which has a time slotted protocol to enable any connection topology, provided that the bus capacity is not exceeded. The time-slot allocation is set by the user in a deterministic way ensuring fixed latency and behaviour.

The carrier board is connected to a host PC for configuration, data, and code exchange.

Nallatech (<http://www.nallatech.com>)

Nallatech provides platforms with a modular approach too. The Nallatech carriers already have embedded FPGA (Xilinx) available to the user. Additional modules, referred to as DIME modules can be chosen according to the processing power requested into a family of Xilinx based modules. Nallatech offered in the past some DSP based modules architected around Analog Devices DSPs. It seems that this option is no longer available and that only FPGA centric modules are proposed.

Pentek (<http://www.pentek.com>)

Pentek offers a wide range of solution for real time digital systems. Many formats are proposed: PCI, cPCI, VME, USB, VXI, VXS, etc. Pentek offers solutions for Software Defined Radio systems and is a member of the SDR Forum Inc. The choice among processors, DSPs, FPGA is vast too and cannot be reported here.

Spectrum Signal (<http://www.spectrumsignal.com/>)

Spectrum's *flexComm* portfolio offers high performance, scalable, rugged and semi-rugged software defined radio transceiver platforms. FlexComm range of solutions includes various capabilities and scales across a large range of embedded computational power, including processors, DSPs and FPGAs. Spectrum signal provides SDR oriented systems including support for the SCA core framework. Hence, Spectrum Signal can offer solutions for JTRS compliant modules.

3.3.2.3. Open platforms in EU projects

Some EU projects have designed digital processing platforms for digital communication systems. A recent example for this is the platform designed in the framework of the IST Magnet Beyond project (www.ist-magnet.org). Figure 3-11 provides an overview of this hardware platform which is described with more details in [Nog08]. It is composed of a digital mother board and RF modules that can operate in the 2.4 and 5.2 GHz bands..

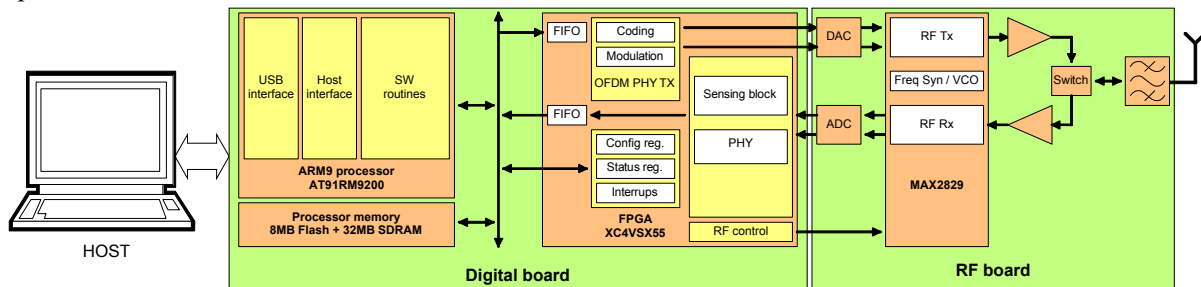


Figure 3-11: hardware platform block diagram overview

The digital board houses an ARM9 processor (AT91RM9200) for SW implementation. For the hardware building blocks implementation a large FPGA (Xilinx Virtex 4 XC4VVSX55-12 is available as explained hereafter. The digital board has several interfaces with the host PC: RS232, USB, Ethernet.

This board includes the following components:

- 1 Xilinx XC4VVSX55-11FF1148C
- 1 Microcontroller AT91RM9200
- 2 SDRAM MICMT48LC16M16A2BG-75 :D (16Mx16)
- 1 Flash-memory AT49BV6416 (4Mx16)
- 1 dual ADC AD9238 (12 bits / sampling frequency = 65 MHz max)
- 1 dual DAC AD9765 (12 bits / sampling frequency = 125 MHz max)
- Interfaces (through microcontroller AT91RM9200) :
 - 1 USB device interface
 - 1 RS232 interface for FPGA configuration
 - 1 Ethernet interface 100 Mbits



Figure 3-12: digital board and housing picture

Table 3-2: MAGNET board available resource

	CPU (ARM920T core)			FPGA Xilinx Virtex4			
	Computational power	ROM	RAM	Logic (slices)	Multipliers (18x18)	Block RAM (18kb)	Clock domains
Available	200MIPS@180MHz	8MB	64MB	24576	512	320	8 DCM

A key interest of this board is its limited form factor (160x75mm), with a significant processing power. Another interest is the use of up-to-date components which are consistent with System on Chip transposition (ARM core, VHDL programmable logic). Finally, this platform is quite open, and a good illustration of this is the fact that it was reused in IST-ORACLE and ICT-WHERE in which a completely different firmware was installed.

3.3.2.4. Baseband and RF platform for multi-standard communications

In this section we will give a description of the baseband and RF platform hardware developed by Institut Eurecom. The purpose of the hardware development is specifically to address the issue of multi-standard communications and dynamic reconfigurability from any standard to another, so the description below is more than a simple description of a particular design, for the choices made for implementation, technology, and architecture described below were highly driven by the multi-standard constraints and needs.

As far as hardware is concerned, the platform is divided in two major parts that interface one with each other:

- a baseband board, called **Express-MIMO**
- a set of three RF cards called **Agile-RF**.

The **Express-MIMO** board is in the PCI-Express format, capable of up to 8x transactions, therefore offering a maximum sustained throughput of 2 Gbytes/s for both RX and TX baseband or low-IF data. The card is basically aimed at being used in a PC (either desktop or laptop) but it can also be used in standalone mode.

The interfaces and the architecture of the Express-MIMO card are given on Figure 3-13. The figures bring out the most important features of the board :

- It is populated with two high density 65 nm FPGAs from the most recent Xilinx Virtex5 family: an **LX110T** and an **LX330**.
 - The LX110T FPGA contains:

- approximately 70K Flip-flop/Latches and 70K LUTs
- 64 Multiplier-Accumulators (MAC) 25bits x 18bits
- 5 Mbits SRAM (organised as 36 Kbits blocks)
- 6 Digital Clock Management/PLL tiles
- 1 PCI Express endpoint block (PCIE 1.0 compatible)
- 1 Ethernet MAC block, Gigabit capable.
- The LX110T FPGA contains:
 - approximately 200K Flip-flop/Latches and 210K LUTs
 - 192 Multiplier-Accumulators (MAC) 25bits x 18bits
 - 10 Mbits SRAM (organised as 36 Kbits blocks)
 - 6 Digital Clock Management/PLL tiles
- It is populated with 4 AD/DA converters with the following features:
 - Resolution of 12 bits in RX (AD), 14 bits in TX (DA)
 - Maximum sampling frequency of 64 Msamples/s in RX (AD), 128 MSamples/s in TX (DA).
 - These 4 AD/DA converters allow for up to a MIMO 4x4 system.
- The LX110T Virtex5 FPGA holds a 32-bit interface with a DDR-SDRAM 64 Mbytes memory module, and hosts a Leon3 Sparc compliant CPU. It shall run the eCos real-time operating system and, on top of this, the software layers involved in the control of the different standard's PHY layers, and possibly a part of the implementation of their MAC layer.
- The LX330 Virtex5 FPGA holds a 64-bit interface with a DDR2-SDRAM 1 Gbyte memory module. Its design will consist in a collection of DSP hardware accelerators each dedicated to a specific functional processing task of the PHY layer. Each one of these DSP hardware accelerators is flexible & dynamically configurable enough for each one to be compliant with the majority of air modulations family (OFDM, CDMA, and single carrier systems). The complete LX330 digital design shall therefore be the core feature to support the multi-standard capability, although this also rely much on the software flexibility offered by the multi-threaded applications that will run, for the different standards, on top of the eCos real-time OS.

In addition to its main PCI Express interface, the Express-MIMO board also holds 3 LVDS interface connectors:

- one 80 LVDS pairs (or 160 single-ended) connector directly connected to the LX110T FPGA
- one 80 LVDS pairs (or 160 single-ended) connector directly connected to the LX330 FPGA
- one 32 LVDS pairs (or 64 single-ended) connector directly connected to the LX330 FPGA

These connectors allow remote interface to any third-party card using these same (popular) LVDS connections, making the Express-MIMO board a flexible system that could be easily integrated in a wider system. Besides, the point-to-point connection from the AD/DA converters to the LX330 FPGA, and the point-to-point connection from the LX330 FPGA to the LVDS connectors make it also possible for an external card to directly interface with the RF front-end (see Agile-RF paragraph below) at the baseband level.

The **AgileRF** boards are a set of 3 cards: one RX board, one TX board, and one LO (Local Oscillator) card, providing synthesized carrier frequency to the RX and TX boards.

These cards are based on up-to-date discrete RF components. Their original design makes them a powerful, **highly configurable RF wideband system**, since they can support, both in TX and RX, a carrier frequency spanning **with no discontinuity from 200 MHz to 7.2 GHz**, with a **bandwidth of 20 MHz at any carrier frequency** in the span. The TX board can emit at up to 21 dBm, and the Noise Figure on the RX path is of 8 dB.

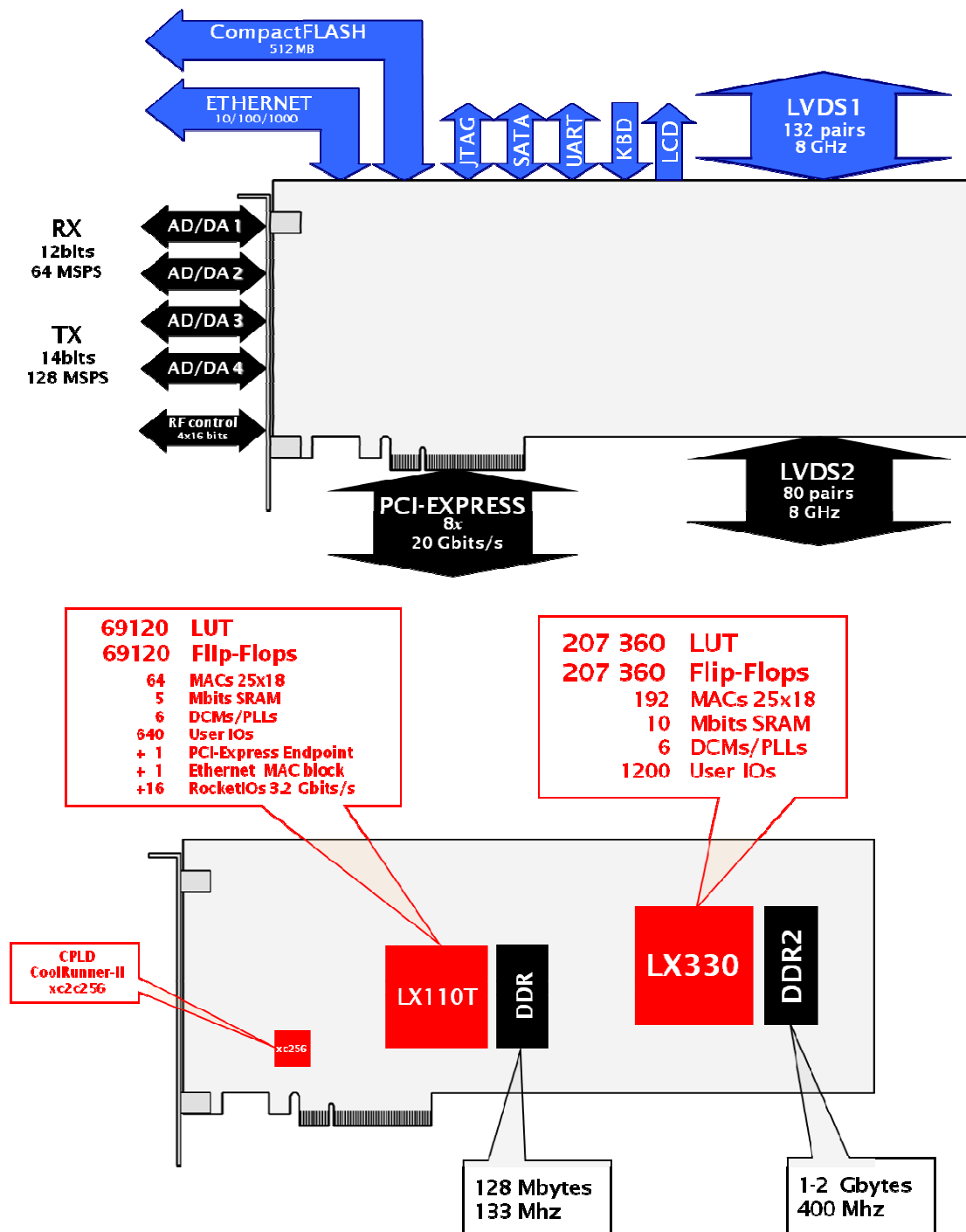


Figure 3-13: Express MIMO overview

3.3.3. Generic flexible platform management: P-HAL

The current and new generation of mobile systems requires high complex signal processing techniques with strong computational demand. As it was discussed above, such demand will only be achievable by large heterogeneous processing systems to benefit from the best efficiency/flexibility tradeoffs. Therefore, signal processing applications development and deployment will require additional management schemes and entities, centralised or distributed, capable to efficiently match computing demands with available computing resources, would they be either hardware or software implemented.

To enable the mapping of transceivers functionality onto these systems in a flexible and convenient way: better upgrade capability, easy maintenance and debug functionality over development time, an abstraction layer is desired. This layer enables the functionality to be designed with a very limited influence coming from the underlying hardware. The first option that comes to mind is to use a set of features that can be found in current Real-Time Operating Systems (RTOS), which offer, to a certain extent, such an abstraction layer. The RTOS functionality can be extended by including block based waveform deployment, easy parameters adjustment, execution time constraints (real time execution) and its monitoring, or computing resource utilization improvement by balancing processor tasks, etc. Nevertheless most of such functionalities are strongly related with underlying platform hardware and in addition, the specific solution for one concrete hardware platform can not be easily ported to an other one. This shortcoming has a strong impact into the transceiver development time.

Therefore, a more general framework to provide an SDR oriented abstraction layer is to be suggested. The first step in the process of defining such a framework were to develop, deploy and test Software Radio applications is to remove platform (hardware and support software) dependencies. Radio functionalities (referred to as radio application – from a programming perspective) can be represented by a graph of different processing functions exchanging data and control information. Thus, the second step consists in simplifying radio application specification trying to concentrate on algorithm description.

Overall, the abstraction layer must be given the possibility to assign as much processing resources as required to a given application, though without any impact on the underlying platform. As expressed before, the flexibility of the platform is not only requested at design time, but also at run time to come with the multi-mode behaviour of modern transceiver to adapt to variations in channel conditions or QoS demands. This leads to an additional requirement on the platform and the abstraction layer which have to ensure a dynamic computing resource management, taking into account radio requirement changes.

Besides, modularity in radio equipment design facilitates the possibility to modify existing platform by adding (or removing) plug-and-play hardware coming from different providers. This enable, multi-provider IP based design and also makes possible the evolution of the platform by including additional feature without impacting the existing ones. Therefore, hardware topologies, configuration mechanisms and specially, tasks assignment to specific processors implies important restrictions to software radio platforms made of different hardware integration.

Many efforts to define a software framework capable to efficiently manage the adaptation of waveforms have been made in the past decades [Mit00],[Ber04],[Kim03],[Ree05]. Only a few of these try to provide a complete development framework, integrating development tools for FPGAs, DSPs, and so forth, and address portability issues. The most relevant framework is probably the Software Communications Architecture (SCA) proposed by the US Dept. of Defence and adopted by the SDR Forum and many research groups [OSSIE],[Lee06],[Agu07]. The flexibility of these frameworks comes at the cost of important additional resources that are needed to control the reconfiguration process. In addition, monitoring, management, and actuating elements must be incorporated for a cognitive resource management, further increasing the computational requirements of the system.

Bearing in mind all these requirements on the radio application oriented abstraction layer an the state of the art, a lightweight multi-platform software abstraction layer and operating environment, P-HAL-OE (*Platform-Hardware Abstraction Layer Operating Environment*), has been defined, designed and developed trying to help to introduce flexibility and increase efficiency to the wireless systems.

P-HAL-OE defines an underlying software structure for independently developed radio application modules interoperability and also to assure P-HAL-OE routines portability (from one platform to another). We define as platform any system, compound of one or more processor, which acts like a

stand alone P-HAL-OE processing entity and uses its associated functionalities to coordinate its execution with other ones. P-HAL-OE is based on a small set of interconnected tasks or services. In order to achieve previous purposes, it is possible to identify a number of tasks which do not change from one platform to another, while others which are platform dependent. The larger the number of functions that are platform independent, the easier it is to port the P-HAL-OE from one platform to another. Conversely, the larger the platform dependent part, the more difficult is to adapt services to a new platform. Therefore, the P-HAL-OE platform dependent part defines elementary services that can potentially be implemented with low cost and a low software depth.

Figure 3-14 shows a schematic of how radio applications are designed and how P-HAL-OE layer unifies platform computing resources, as a single Virtual Platform. P-HAL-OE provides the following set of features:

- Real-time seamless exchange of information from one P-HAL-OE compliant platform to another.
- Isochronisms of data and processes running on different platforms.
- Platform-wide coordinated process control, scheduling, logging and control of errors.
- Real-time system monitoring, data and statistics retrieval and adaptation of processes set-up parameters.

The three first features in the list produce the effect, from the application point of view, of having a single and unified platform. Then, the same application description works on a single machine or on multiple distributed machines. It does not matter where a specific part of the application is running, this part will just observe other parts as being aside.

In principle, hardware platform components, or hardware platforms when considering a more complex framework, only have some minor information about other ones they have to work with. One or more (direct or indirect) shared interfaces are the only liaison between each pair of platforms (e.g. network, bus, etc. interface). Through such interfaces, P-HAL-OE compliant software on each platform builds-up a virtual larger platform. It is worthy to mention that the performance of this interface may limit the aggregate computational capacity of the overall platform but in general not more than internal platform interfaces may do.

Since some parts of the application have to deal with real-time issues, two important timing considerations have been added to P-HAL-OE. First one, isochronisms of platforms, requires that existent interfaces between them are used to create a local time reference adjusted with the local reference of one platform. A synchronism procedure is required to overcome those delays present in the aforementioned interfaces. Second, a distributed pipelined task scheduling ensures a deterministic end-to-end latency fulfilling real-time application requirements while minimizing resource utilization.

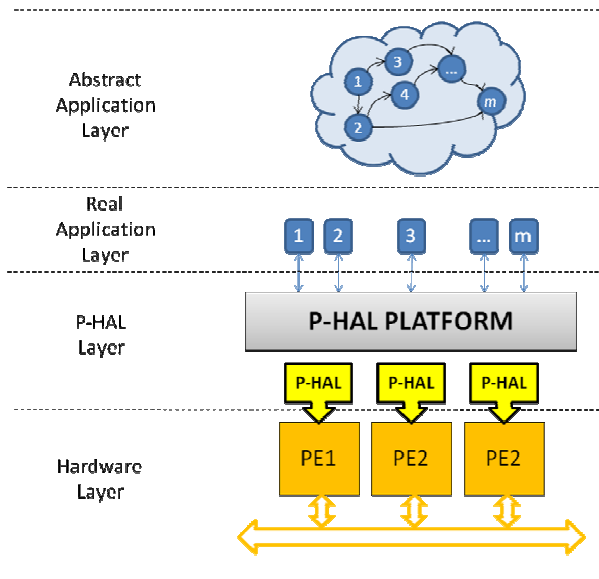


Figure 3-14: P-HAL Layers Schematic

3.3.4. Cognitive radio equipments management architecture: HDCRAM

3.3.4.1. Cognitive radio equipments

Cognitive radio relies on Mitola’s work in [Mit99] and [Mit00]. Mitola argues that radio will become more and more autonomous, and thanks to the support of flexible technology (namely SDR) will acquire some self-autonomy to dynamically modify its functionality. As explained in the schematic of Figure 3-15, this relies on a cognitive circle. Figure 3-15(a) is from [Mit00] and Figure 3-15(b) is a simplified view of the cycle summarized in three main steps:

- observe: gathers all the sensing means of a CR,
- decide: represents all that implies some intelligence including learning, planning decision taking,
- adapt: reconfigures the radio, designed with SDR principles in order to be as flexible as possible.

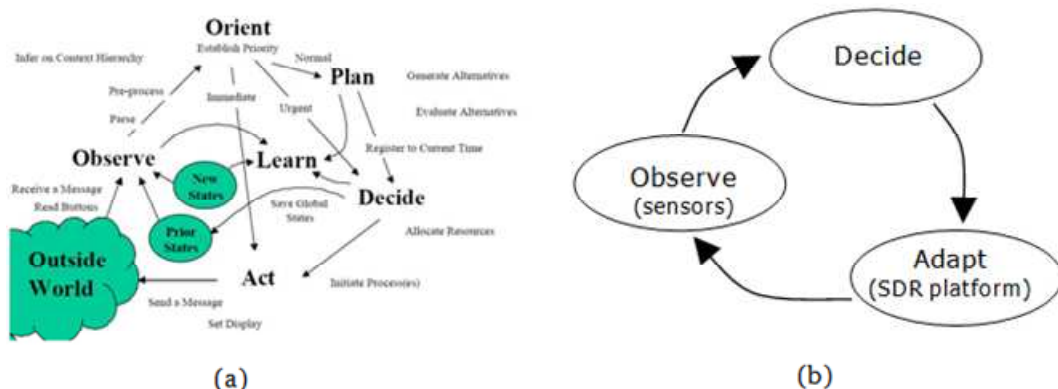


Figure 3-15: (a) Mitola’s cognitive cycle, (b) simplified version

Sensing is addressed in the following its broadest sense. All information that can help the radio to better adapt its functionality for a given service in a given environment, in other words under given constraints, is worth being taking into account. Then there is no restriction on the sensors nature, it is possible to draw the general approach exposed in Figure 3-16.

Sensors	Layer	Literature concepts
User profile (price, personal choices) Localization, sound, video, position, speed, security.	Application	Context Aware
Intra-network, and inter-network vertical handover, standards, load	Transport Network	Interoperability Ambiant networks
Access mode, power modulation, coding, Frequency, handover, Channel Estimation	Data link Physical	Link adaptation

Figure 3-16 – Simplified OSI model for cognitive radio context

However, due to the high financial pressure on spectrum issues, CR is often restricted in the research community to spectrum management aspects. Opportunistic spectrum access approaches are explored to increase the global use of the spectrum resources. FCC has already opened the door for unlicensed secondary usage of the TV broadcasting bands, and permits to secondary users (e.g. not licensed) to occupy primary users spectrum when available. Opportunistic radio has been deeply and extensively addressed by the IST-ORACLE project (www.ist-oracle.org). More futuristic CR scenarios may also be considered concerning the spectrum management. Relevant information on this aspect can be found in the IEEE SCC41 documentation (www.scc41.org). The most futuristic scenarios consider a fully deregulated spectrum access where all radio connections features would be defined on-the-fly: carrier frequency, modulation, data rate, coding scheme, etc. The main burden for this scenario to happen comes from the regulation aspect and the underlying business models, as current wireless systems stakeholders see such a scenario as a too profound change from the current model.

We can derive from the considerations of previous paragraph what should a CR equipment be made of [Moy05]. This is exposed in Figure 3-17 which schematically represents a CR equipment.

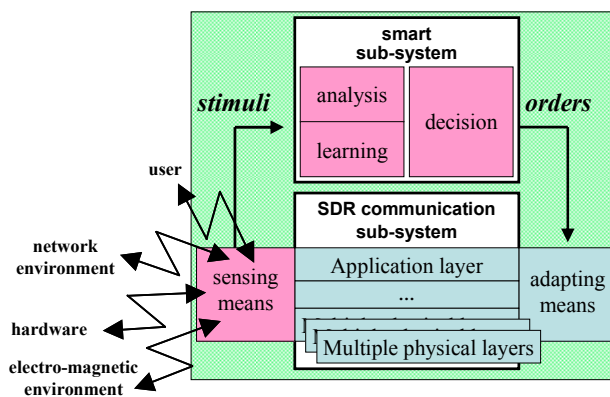


Figure 3-17 – Cognitive radio equipment functional block diagram

The SDR system sub-part is composed of multiple radio protocol stacks (from the physical to the application layer) executed in a flexible hardware (and corresponding necessary software) platform. As explained in Figure 3-15, to make an SDR become a CR, two main parts are to be added: sensing means and a smart sub-system. Sensors are the combination of electronic devices and algorithms that translate the signals into metrics of interest. For instance, one may consider a standard recognition sensor as being composed of an RF front-end and a set of processing functions that extract the metric of presence or absence of a set of standard radios. This information feeds the smart or cognitive engine of the CR equipment that takes decisions accordingly. This means that reconfiguration may be done to adapt the behavior of the equipment to the situation revealed by the sensors of the equipment, or sent by the network, or both of them.

There is an intense discussion in the SDR/CR community on whether most of the intelligence should be contained in the network or in the terminal. Defenders of a network-centric cognitive scheme argue that this permits to lower the terminal complexity, which is of major importance due to its restricted embedded capabilities. The network is also seen as the best place to centralize a very large set of information which permits global network optimization. Others think that the terminal itself is the best equipment to know what its operating conditions are. The mobility increase of equipments is also an argument for the terminal-centric approach as it will be harder for the network to follow the environment changes if they speed up. As it can also benefit from network information through communication means, a distributed cognitive approach is preferred. We do not want here to choose or decide for one or the other solution. We believe that a combination of both is worth. Depending on each situation, one orientation maybe privileged from time to time. Anyway, the equipment cognitive management scheme proposed in this document is completely compatible with both network-centric and terminal-centric cognitive radio management. Its complexity of course is lowered in the context of a mostly network-centric approach.

3.3.4.2. HDCRAM: a management architecture for cognitive radio equipment

As indicated in the cognitive cycle of Figure 3-15(b), a CR equipment management architecture has 3 duties: offering to the equipment the support for capabilities:

- to sense its environment,
- to take decision,
- to reconfigure.
- It is what HDCRAM does.

It is what HDCRAM does. HDCRAM stands for Hierarchical and Distributed Cognitive Radio Architecture Management. It has been published in [God08].

In order to make it understandable and re-usable by the community, a meta-model of the HDCRAM has been derived, which is presented in Figure 3-18. This meta-model represents a factorized view of the architecture that has to be deployed in a cognitive radio equipment accordingly to the functionality expected by the designer. In order to keep the figure readable, classes particularities such as attributes or operations have been hidden. The ReM and CRM classes are super classes from which the other classes are derived (except for class Operator). At the bottom of this figure, the Operator class is used as the parent class for any reconfigurable and/or metric's provider signal processing component.

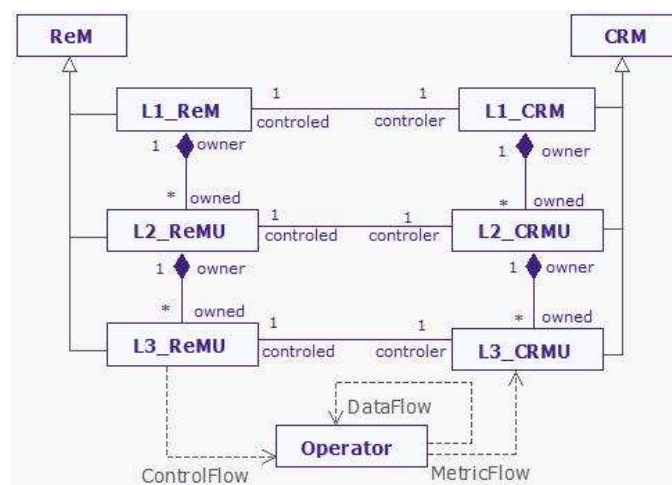


Figure 3-18 – Meta-model of the HDCRAM

A radio chain is made of interconnected operators which are operating signal processing activities so that the equipment may implement a radio communication standard. Another kind of operators, in a cognitive radio equipment, will be formed by new processing elements offering sensing capabilities. We can call them sensors. To each operator of the radio chain at the bottom of Figure 3-18 corresponds a couple of managers: L3_ReMU and L3_CRMU. ReM side is responsible for the reconfiguration of operators, CRM side for the decision taking. Several operators may advantageously

combine their metrics if they are sensors, or their reconfiguration procedures if they are operators, so that their L3 managers may converge their information at level 2 in a L2_ReMU on reconfiguration side, and L2_FRMU on the cognitive side. Finally, at top level, one global manager composed of a L1_ReM and a L1_CRM, is controlling the globality of the equipment. All this is detailed in [God09].

3.3.4.3. HDCRAM simulator: a simulator of CR scenarios to dimension CR equipments

Main characteristics are: (i) separation of data and control flow; (ii) separation of the control part of a CR equipment into two sub-parts reconfiguration and cognitive management; (iii) three levels of distribution with a hierarchical approach; (iiii) top-down approach for ReM part; (iiiii) bottom-up approach for CRM part. A common description language has also been chosen: the Object-Oriented Unified Modeling Language (UML) [OMG01]. UML provides the ability to model complex applications or huge systems through easy to understand annotated diagrams. The choice of an executable meta-modeling approach instead of a pure UML description permits to obtain an executable version of the architecture. It relies on the use of Kermeta language from the Triskell project of INRIA [Mul05]. This adds a new dimension to the design approach since it gives the opportunity to simulate the obtained architecture on actual CR scenarios. This is a great advantage in the research perspective that consists at defining and refining a CR management architecture. The HDCRAM design becomes indeed a real investigation tool, and not only a purely descriptive specification as just expected at the beginning. This is illustrated in Figure 3-19.

CR equipment specifications State of the art	HDCRAM	Functionalities Reality
Language (UML)	HDCRAM Meta-model	Symbolic representation of the reality
Design specific language (Kermeta)	HDCRAM Simulator	Verification/Refinement of the functionalities

Figure 3-19 – From an HDCRAM to an HDCRAM simulator

Figure 3-20 offers a more detailed view of the HDCRAM metamodel compared to Figure 3-18, highlighting the simulator facet [God08]. At the top of this representation, the HDCRAM_simulator class is used only for simulation purposes. It plays the role of a Host that launches the simulation and gives the opportunity to create a back up of played scenarios in XMI format (for exchange with other compliant XMI tools). It also instantiates the first elements of the HDCRAM: create the L1_ReM class and set its attributes. This figure shows new features in the meta-model. Those features are the StandardLibrary, FunctionLibrary and OperatorLibrary which store functionalities needed by CRM and ReM units deployed in the equipment. By functionalities is meant cognitive algorithms, executable files, table's metrics, functions needed to instantiate a standard, operators needed to instantiate a function, etc. For each couple CRM/ReM one library is created. Those libraries can be modified by self learning or under instructions of the network and then fulfill the need for a CR system to be evolvable.

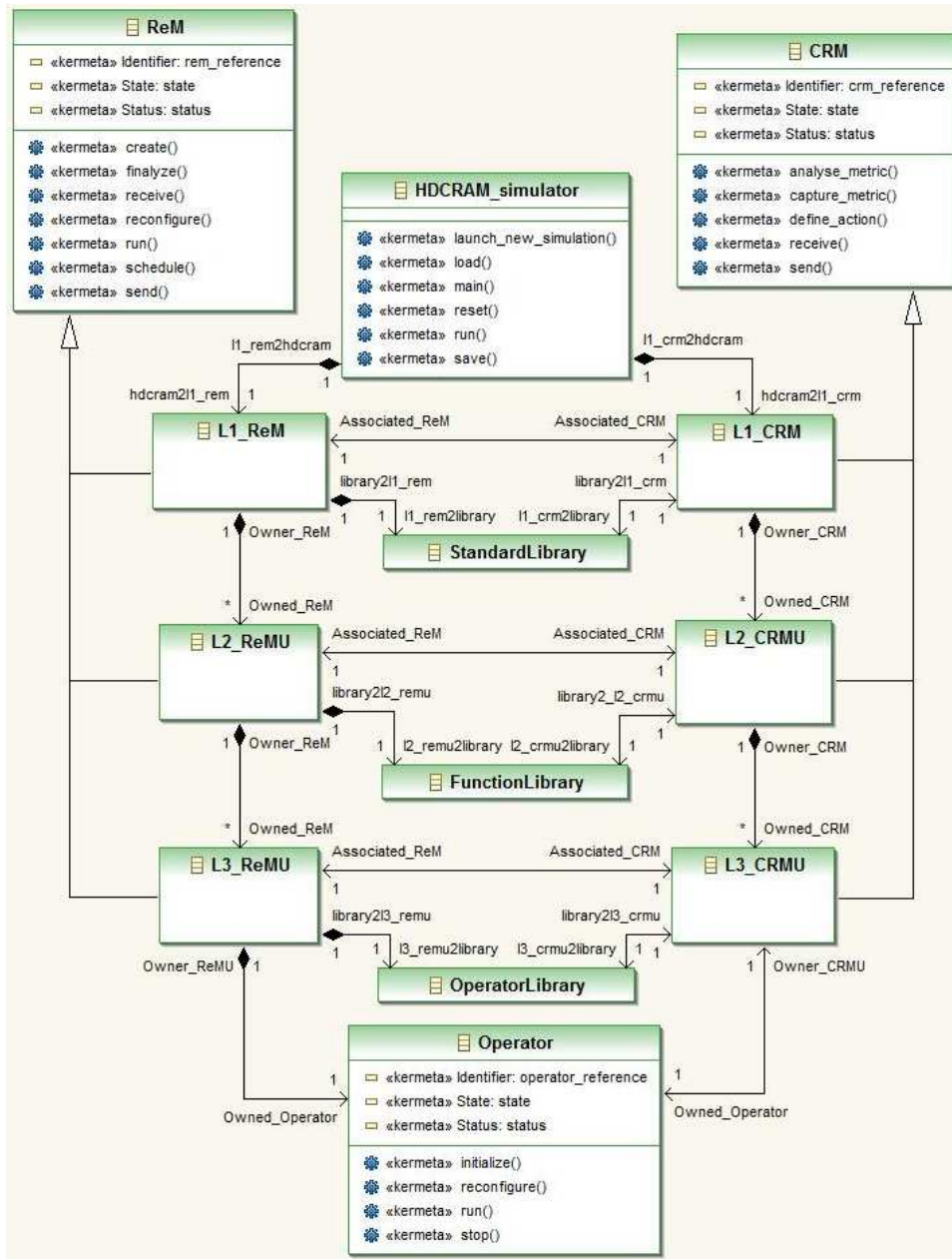


Figure 3-20: HDCRAM meta-model

The HDCRAM simulator can be used to design an HDCRAM architecture matched to a cognitive radio scenario. The goal is to define and dimension all the management facilities that have to be inserted in the equipment in order to support cognitive radio requirements associated to a given scenario. Designing a CR equipment consists then to define and play all the cognitive scenarios the equipment has to support. The proposed tool may help the designer to identify all the interactions and corresponding functionalities necessary to be added to the pure radio modulation and demodulation processing operators. This implies both pure managing facilities, but also the signal processing facilities that can extract metrics of any nature (not only spectrum-oriented), that are called sensors in this context.

Figure 3-21 presents a blind standard recognition scenario played by the simulator. This new concept of blind recognition [Rol03] aims at blindly identifying all the wireless radio standards in use in the vicinity of the terminal. The following step that is not described would consist in adapting the equipment so that it can use the most appropriate of the recognized standards according to service requirements and user needs. We propose to model and simulate the standard recognition sensor

[Hac07] in the HDCRAM simulator in order to give an example of the functionalities of our architecture. The Blind Standard Recognition sensor is integrated as follows: the control of the sensor is made by one couple L2_ReMU/L2_CRMU and for each operator of the sensor, one couple L3_ReMU/L3_CRMU is integrated. The five operators of the blind standard recognition sensor are gathered in two modules, one searching module and one analyzing module as shown in Figure 3-21.

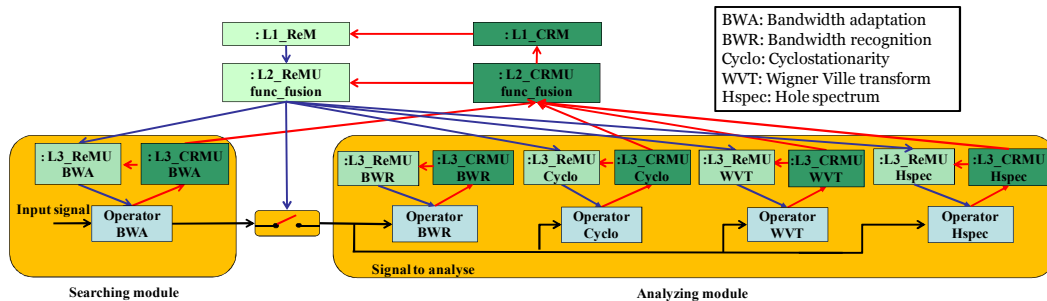


Figure 3-21: Blind standard recognition sensor

The bottom signal processing line is comparable to what usually exists in a conventional radio equipment. An SDR perspective is implemented here, and so each processing element (BWA: BandWidth Adaptation, BWR: BandWidth Recognition Cyclo: cyclostationarity, WVT: Wigner-Ville Transform, Hspec: Hole spectrum) is a programmable (in a processor) or reconfigurable (in a FPGA) operator. We see here that the cognitive management including both reconfiguration and “cognitive” facilities, requires a lot of added management units. This is the price to pay for a complete and efficient cognitive management. Of course, each of these units is not as demanding as signal processing operators in terms of computing power. Moreover, the management architecture may also be restricted to a set of particular operators in the CR equipment, which permits to minimize the management overhead in terms of complexity. As specified by the HDCRAM metamodel, the management infrastructure associates to each operator a couple made of one reconfiguration management unit L3_ReMU and one cognitive management unit L3_CRMU. As the set of operators presented in Figure 3-21 is having a global functionality of standard recognition, their global management is centralized at level 2 and they all share the same L2_ReMU and L2_CRMU. L2_CRMU for instance merges metrics from all four BWR, Cyclo, WVT and Hspec operators to perform a fusion, validates the set of metrics and extracts one abstract metric. This abstract metric is sent to the L1_CRM which deduces the presence or not of a standard radios within a pre-defined set of standards as explained in [Hac09].

4. FLEXIBLE HARDWARE ARCHITECTURE FOR COMPUTATIONALLY INTENSIVE PROCESSING

Support to a large variety of wireless communication standards and operation modes is today required in mobile devices, which must then incorporate an increasing level of flexibility. During the past years, great advances have been achieved in the implementation of multi-processor system-on-chip platforms (MP-SOC) efficiently supporting several base-band processing functions and MPSOC technology is the key enabling vehicle for Tier-2 Software Defined Radio. However, some of the main processing tasks in a wireless transceiver are computationally intensive and their implementation usually resorts to the design of dedicated hardware accelerators.

4.1. FFT operators for OFDM

In the following sections we present first a survey on the state-of-the-art FFT architectures targeting high throughput, optimizing cost and power dissipation. Then, the approaches for designing flexible FFT architectures developed so far within the NEWCOM⁺⁺ project are described.

4.1.1. FFT architectures for telecommunication requirements

Since the introduction of the FFT algorithm, a plethora of architectures has been proposed to further improve the performance of the initial algorithm. The performance required by the FFT computation on N points demands either a single processor driven to a very high clock frequency [Lee02] or alternatively the implementation of an Application Specific Integrated Circuit (ASIC) solution utilizing parallel processing and bit-pipelining techniques [He98]. Speeding up the FFT algorithm becomes an important issue as applications become more demanding. In addition to the speed requirements of the past decades, modern applications impose to the FFT designs a small Area/Memory footprint requirement, power consumption efficiency, a low data-frame processing latency and large FFT data-frame sizes. This situation typically arises in telecommunication applications in which the FFT performance affects the modems in the following three (3) areas:

- 1) Low latency is required for minimizing the round trip delay,
- 2) Low power is required for portable transceivers and for minimizing the chip temperature,
- 3) FFT computations -with the input set of points of large size- are required for high bit-rate broadband systems.

Several techniques can be applied in order to improve the performance of the FFT algorithm with respect to the above parameters. These techniques include parallelizing the FFT processing pipeline, increasing the radix of the algorithm, decreasing the clock frequency in order to minimize the transitions and reduce the power consumption, as well as combinations of the above. The vast majority of relevant architectures utilize FFT processors with either Radix-2, or Radix-4. Radix-2 calculations allow a straightforward implementation while the use of Radix-4 reduces the number of multiplications and also the number of stages in the FFT computation.

FFT organizations that have been proposed in the literature [Tho83, Wol84, He96, OH06, Wu05, Cha05, Lee06] vary with respect to the level of parallelism, the sustained throughput rate, the memory size, the utilization of the hardware resources and the power dissipation. Fully unfolded FFT designs sustain the maximum throughput at lower clock rates but they occupy more VLSI area and use larger size memory between successive stages than any other parallel FFT organization. Cascade FFT topologies [Tho83] reduce memory requirements but lack efficiency for higher than radix-2 structures.

A straightforward strategy to achieve low power consumption is to increase parallelism and to reduce the operating frequency. An alternative approach utilizes asynchronous FFT designs [Sut98] with fully unrolled FFT circuits but occupies more VLSI area. Also, a different way to keep power consumption to a minimum is by performing FFT computations with significantly scalable power dissipation across FFT sizes [Zho06, Tak06].

Addressing the problem of high throughput for real-time FFT computations, architectures utilizing higher radix techniques have been developed [Bab06, Man07]. Higher radix techniques reduce the number of stages of the FFT at an increased cost in terms of VLSI area for each stage. These organizations are deeply pipelined to maximize the operating frequency and sustain high throughput rates. A different approach in order to tackle the task of high throughput wireless channels is by using a multiple-input multiple output orthogonal frequency division multiplexing (MIMO OFDM) [Che06] presents a block scaling FFT/IFFT processor for WiMAX applications, utilizing a ping-pong cache-memory architecture, reducing as a result both power consumption and hardware. Also, parallelizing the memory accesses can improve the performance and the efficiency of the FFT computations. [Rei06] presents a technique which allows the parallelization of the memory accesses in hardware implementations of the FFT algorithm and leads to increasing the speed up by a factor equal to the number of parallel accesses.

Finally, during the last few years, with the convergence of multiple radio standards into a single terminal, the need for a single processor able to perform FFT/IFFT computations of variable length has risen. As a result, a number of different realizations have been proposed. These architectures try to cope with the problem of different sizes FFT computations, maintaining at the same time low power consumption, efficient hardware utilization and the best possible throughput rate [Pan07, Wan07, Kuo03].

4.1.2. Approaches for Designing FFT Architectures within the NEWCOM⁺⁺ t

This section presents the approaches for designing flexible FFT architectures as these have been developed within the NEWCOM⁺⁺.

High Throughput Approach: Since the main concept of the NEWCOM⁺⁺ project is flexibility, the Digital Systems Team (IASA) is keen on exploring the idea of a highly flexible, reconfigurable FFT/IFFT architecture, which will be able to accommodate different standards. This flexible architecture will be able to fit in various existing OFDM-based communications systems. This design will be able to perform Fast Fourier Transform computations of different sizes, varying from 128 to 2048 complex points. The processor will utilize multiple data streams. The design will target either achieving high throughput rates or minimize power dissipation.

There are several different scenarios and approaches that have to be studied, in order to obtain the best possible organization. The proposed design should consist of a different number of FFT engines, interconnected in such a way as to provide variable length FFT computations. These engines should also have the ability to operate as standalone FFT processors (for small size FFT computations), thus offering better hardware utilization. Moreover, implementation of a cached FFT organization reduces the memory accesses, thus offering improved power consumption. On the other hand, parallelizing the memory accesses can significantly speed-up the FFT computations. Proper scheduling of the input data streams is required, in order to avoid stalls and to achieve efficient hardware utilization. As a result of complex control, the proposed design may suffer from additional hardware cost, compared to a fixed length FFT processor. Another challenging task consists in being able to operate the various FFT engines on different frequencies and/or throughput, depending on the application specifications. Such an achievement will be critical for the flexibility of the entire system. Finally, since power consumption is critical for portable systems, this processor should be power efficient. The proposed organization should offer power scalability, due to the reconfigurable architecture. Depending on the application, this architecture offers the option of “shutting down” the unnecessary modules, thus improving the power consumption.

The Digital Systems Team (IASA) is considering the use of the IEEE.802.16e WiMAX standard, as a reference scenario for the proposed architecture. In such a case, several other considerations have to be made, regarding the modulation types and the interval guard modes, which have to be supported.

Air Interface focused approach: In the following, the design of a high throughput architecture for multi-point Fast Fourier Transform (FFT) is presented. This implementation is part of flexible platform being developed at Institute Eurecom, an N⁺⁺ partner through CNRS. The design is based on identifying the basic common processing blocks, and then putting those in different processing units to achieve a hierarchical, flexible yet efficient hardware / software co-design. Front End Processor (FEP) is one such identified block in our design, catering the air-interface requirement for a flexible radio.

Though FFT makes the core of frequency domain receivers for different air-interfaces as mentioned above, there are some other blocks required to execute all other the tasks. As an example, Carrier Phase Offset (CPO) estimation in OFDM system can be performed using a dot product operation over pilot symbol position in each OFDM symbol. Dot-Product is also useful in WCDMA/Single-Carrier Systems not only for CPO but also for synchronization. Similarly, initial synchronization in 802.11x, 3G and 802.16 can be achieved using the FFT, component-wise-product, and peak detection (a max calculation over sub-bands is sufficient). Functionalities like signal-to-noise ratio estimation and automatic gain control require the knowledge of the received signal energy, and hence an energy calculation block can be added to the functional requirements of the Front End Processor (FEP). Table I lists some use cases for the macro-operations required by the different air-interfaces supported by our FEP block.

Table 4-1: Air Interface Operations and relevant Macro Processing blocks

Operations	Air Interfaces		
	OFDM/A	SC-FDMA	WCDMA
<i>Channel Estimation</i>	DFT + Component-wise Product	DFT + Component-wise Product	DFT
<i>Energy Calculations</i>	Energy Calculations	Energy Calculations	Energy Calculations
<i>Data Detection</i>	DFT + Component-wise Product	DFT + Component-wise Product	DFT + Component-wise Product / Time domain dot product
<i>Synchronization</i>	Max Calculations, Filtering via DFT or Dot-Product	Filtering via DFT or Dot-Product	Dot-Product
<i>CPO Estimation</i>	Dot-Product over sub-bands	Dot-Product over sub-bands	Dot-Product
<i>MIMO Signal Processing</i>	Matrix /Vector Processing (Dot- Product + Accumulation)	Matrix /Vector Processing (Dot- Product + Accumulation)	Matrix /Vector Processing (Dot-Product + Accumulation)

Once the macro-processing blocks are defined, the next step is to look for proper hardware components. A thorough analysis of computation intensive block of FEP, i.e. DFT/IDFT, was carried out to select the DSP slices for its mathematical operations. Candidate standards suggest that the number of input samples be limited to powers of two between 8 and 4096. This allows calculating the DFT using simpler and efficient algorithms such as radix-4 FFT and split-radix FFT. The selection of number of bits representation of each sample is based on target technology, hardware resources, maximum achievable frequency of end product, and dynamic range of AD converters. Therefore, all input samples to FEP block are represented in 32-bits, with real and imaginary part each of 16-bits.

To calculate an N-point FFT, Radix-4 [Lao05] and Split-Radix [Yav68] are the most widely used and efficient schemes given the aforementioned processing block. Radix-4 computes FFT for vector sizes that are powers of 4 in M stages (where $M = \log_4 N$), while split-radix algorithm also supports vector sizes that are powers of 2. Radix-4 is more computation efficient and is preferred over Split-Radix as long as the number of input samples (vector size) is power of 4, otherwise Split-radix is used in our design. Split-Radix performs all radix-4 stages (operations) except the first or last stage, which is a Radix-2 stage (operation). The choice of first or last stage is dependent on the decimation scheme used: for Decimation in Frequency (DIF) the first stage is of type Radix-2, while for Decimation in Time (DIT) the last stage is of Radix-2 type. Radix-4 FFT algorithm implementation is based on N/4

butterfly operations [Yav68] per stage, each requiring four complex additions and then one complex multiplication.

The aim of the presented processing block and architecture is to come up with a research based prototyping experimental platform, and is not meant for any mass scale production; therefore FPGA is considered as the target technology FPGAs rather than ASICs. Among the latest technologies available Virtex-V FPGA by Xilinx has DSP48E slices which have '25*18' bit multipliers and 48-bit adders, quite close to the requirements of FFT processing unit of FEP. The most likely input data samples to FFT operation in the FEP block are 16-QAM signal, and Zero-Mean White-Noise. Simulation analysis of these input data samples reveals that the FFT operations never exceed the limit of 25-bits with any input samples range. This makes the Virtex-V the best choice with no extra operation of saturation and/or re-scaling, and also provides a good resolution of 25 bits in the intermediate stages of FFT operation.

The overall throughput of the most computation intensive operation (the in DFT) is set to one sample per cycle. 4096-point-DFT is the maximum value supported with radix-4 algorithm, and it would take six stages while each stage would require $N/4$ butterfly operations. To meet the functional specification of throughput, two butterfly operations per cycle need to be performed which in turn implies that eight samples and six twiddle factors are accessed (read from memory) in each cycle. Thanks to sophisticated memory organization and memory access algorithm, only $N/8$ twiddles are stored in memory and just two twiddle factors are accessed per cycle instead of six. To match the input samples access requirements, memory banks for samples are divided into sub-memory systems to have enough memory reads/writes every cycle. Each butterfly operation requires three complex multiplications, which in turn means utilization of twelve DSP48E multipliers. Thus DFT processing block utilizes 24 DSP48E slices. Initial synthesis results reveal that the critical operation in FEP, i.e., butterfly operation of the DFT module, can achieve 230 MHz frequency (minimum). This, in turn, implies for a 2048-point-FFT operation a (minimum) throughput of 150 M-samples/second with the current design, to be further improved with design/code optimizations.

4.2. FEC co-decoders

Current implementations of decoders for modern Forward Error Correction (FEC) are in the form of accelerators optimized for specific standards, but do not take into account flexibility and scalability issues. Particularly, the current design approach implies the allocation of multiple separated HW accelerators to realize multi-standard systems, which often result in poor HW efficiency and long design times. So a key research challenge is the design of HW accelerators for channel decoding incorporating some elements of flexibility.

A first objective is the search for efficient FEC architectures handling several codes of the same family: a significant case study is represented by LDPC (Low Density Parity Check) codes. Codes of this family are usually decoded by means of a "message passing" algorithm that implements the iterative exchange of messages generated by processing nodes. The communication structure to support this message exchange is critical: fully parallel architectures result in excessive cost and serial architectures are too slow, while partially parallel architectures allow for proper cost-performance trade-offs. Unfortunately, parallelism introduces the collision problem in memory access, a well-known problem already addressed in parallel turbo decoders. Another challenging task is the design of LDPC decoders that are flexible in terms of supported block sizes and code rates. The need for decoders able to dynamically adapt to different transmission conditions is well known in practical communication systems

An even more ambitious objective is the exploration of FEC architectures dedicated to both LDPC and turbo codes. Results on this side have been published only very recently and they are all based on the MP-SOC approach. An alternative solution can be searched in the direction of flexible hardware accelerators, which exploit the commonalities between the LDPC and turbo-decoding algorithms in order to come to a versatile architecture, covering both code families with the same hardware support.

In both cases, exploiting the inherent parallelism of the iterative decoding algorithm and enabling the efficient partitioning of the decoding task among several communicating processing elements, the throughput can be efficiently multiplied. A communication network has to support the communication demands of the different processing elements without degrading the throughput of the overall system.

4.3. LDPC operators

In this subsection, issues related to the design and the implementation of a high-performance LDPC decoders are discussed. Starting from this overview some possible approaches aimed to provide high-throughput decoders with different flexibility levels are presented. The flexibility cost, presenting some objective metrics that could be used as a way to assess such cost is also tackled.

4.3.1. *Partially parallel LDPC codes and their implementation*

Given the importance of LDPC codes in modern telecommunication standards, an increasing number of research works have been published in the past years addressing the problem of achieving high decoding throughputs with moderate hardware complexity [Bla02, Bra06, Man03, Kie03, Dar05, Kan06, Mas07]. The most typical decoding procedure is the *Message Passing Algorithm* [Kay99], that implements the iterative exchange of node-generated messages along the edges of the graph. The problem of high throughput LDPC decoding can be further divided in two tasks:

- Finding a suitable structure for the node elements (Variable Nodes and Check nodes) able to provide the required computational capabilities;
- Finding a proper interconnection structure able to sustain the heavy point-to-point traffic among VNs and CNs.

Given very low adjacency of associated parity check matrices, the latter problem is particularly significant when high throughputs are required.

Several realizations have been proposed to tackle the problem of efficient message delivery between the two types of processing units. Fully parallel architectures [Bla02] usually result in excessive implementation cost, while serial architectures are too slow for most applications. Partially Parallel architectures, with multiple processing units serving VNs and CNs allow for proper cost-performance trade-offs: a complete description of partially parallel decoders' design space is available in [Bra06].

Unfortunately parallelism introduces the collision problem in memory access, a well-known problem already addressed in parallel turbo decoders implementations. A solution to the collision problem can be found in the scheme proposed in [Tar04]; it is important to observe how the idea presented in this work can be applied both to turbo and to LDPC decoders. In [Qua06] some complexity and throughput estimation for an LDPC decoder based upon the aforementioned approach are presented.

From a literature review it is possible to observe two distinct approaches to the collision problem:

- To design collision free codes [Man03, Hoc03].
- To design a decoder architecture able to avoid or mitigate collision effects [Tar04, Kie03, Qua06].

Even if the first approach has proven to be effective, it significantly limits the supported code classes, spoiling the flexibility of the obtained solution. The second approach, on the other hand, is well suited for flexible and general architectures.

Another challenging task is the design of LDPC decoders that are flexible in terms of supported block sizes and code rates. The need for decoders able to dynamically adapt to different transmission conditions is well known in practical communication systems. This problem has been faced in the past again introducing suitable constraints in the code design and exploiting regularities in the structure of the parity check matrix [Vil04]. Such approach has proved to be very efficient when the required flexibility is limited to a set of pre-defined codes, but it tends to be inadequate when the support to generally defined codes is required. With the approach introduced in [Mas07] it is possible to design a fully flexible decoder able to support any LDPC code. The underlying architecture is based on a

scalable, partially parallel topology and a reorganized message-passing algorithm. independently proposed by [Mas05a, Mas05b, Dar05, Kan06].

Error! Reference source not found. shows the block scheme of the implemented decoder. It is interesting to observe how this architecture can be essentially split into three separate parts:

- The Processing Elements (PEs) standing on the leftmost part of the figure, devoted to implement the node functionalities;
- Each PE is connected to a local memory: generated messages that need to be delivered to different PE are stored in this memory;
- An $N \times N$ non-blocking interconnection network is used to mitigate collisions. According to simulation it has been shown [Mas07] that performance degradation caused by collisions can be mitigated using graph partitioning techniques.

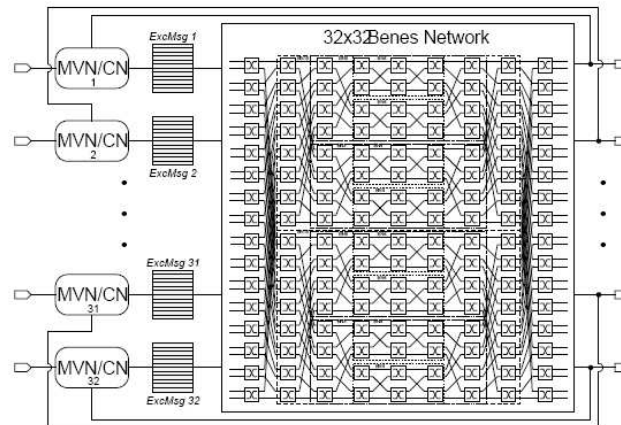


Figure 4-1: decoder network

Each PE implements the so-called LT-BPA (Low Traffic Belief Propagation Algorithm). In **Error! Reference source not found.** the internal architecture of such a node is shown. More details on this algorithm can be found in [Qua05] and [Mas07].

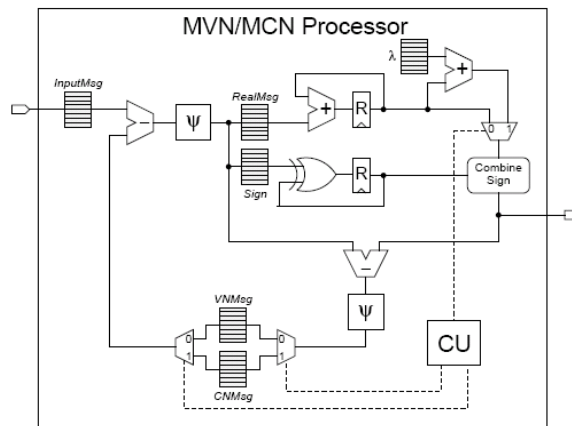


Figure 4-2: PE architecture for LT-BPA

Table 4-2: LDPC implementation report

	<i>B-H</i>	<i>K-P</i>	<i>this work (code 4)</i>
Max. Clock Freq. [MHz]	64	200	245
Gate Count for Logic	1750 K	457 K	170 K
Area for Memory [mm ²]	—	—	2.63
Total Area [mm ²]	52.5	6.29	2.94
Throughput (8 iter.) (Mbps)	8000	582	31.2
block size	1024	1024	1024
number of edges	3328	3078	3815
TAR	230.83	177.39	10.63

It is also important to observe how the number of PEs in the architecture can be tuned according to the system specification and requirements. This means that in this flexible approach it is possible to tune the system's parallelism to the worst case required decoding throughput. Assuming that each PE is able to process a single incoming message every clock cycle, conversely producing a single output message, it is possible to determine the minimum number of PE operating in parallel in order to reach a given performance objective. As introduced in [Gui07], it is possible to define the Processing Power (P_c) as:

$$P_c = \frac{D}{f_{clk} \cdot R_c} \cdot \frac{\epsilon \cdot i_{max}}{N} = \frac{\epsilon \cdot i_{max} \cdot D}{K \cdot f_{clk}} \left[\frac{\text{edges}}{\text{cycle}} \right]$$

where D is the required information throughput, R_c is the code rate, K is the number of information bits per codeword, N is the block size i.e. $N=K/R_c$, f_{clk} is the operating frequency, i_{max} the number of decoding iteration (actually multiplied by two since a TP-BPA is considered) and ϵ is the total number of edges to be processed.

It is central to observe how P_c represents only a sort of performance lower bound since the actual number of parallel operating processors depends on the efficiency of the interconnection structure. As an example the LDPC code from the IEEE 802.16e standard can be considered, i.e. the rate $\frac{1}{2}$ (1728, 864) one. In that case, assuming a clock frequency of 100 MHz, 10 iterations and a target decoding throughput of 20 Mbps, the minimum required P_c is 13.5. This means that observing only the P_c it would be possible to conclude that a partially parallel decoder with 16 PEs operating in parallel would be sufficient. In reality, simulations in [Mas07] show that message delivery cycles impacts negatively on the performance, making this number only a theoretical one. Resorting to multicast message delivery and graph partitioning techniques it has been shown how these performance can be achieved resorting to 32 PEs operating in parallel.

As far as the flexibility is concerned, the proposed scheme is able to support virtually any kind of code. Since no assumptions are made on the code structure, the previous statement holds not only for partially structured codes but also for random irregular LDPC. Finally, it is interesting to assess the cost of such flexibility: since it is commonly perceived that *flexibility* is difficult to quantify, one can try to measure the *efficiency* of the proposed solution. To do this in [Mas07] the concept of Throughput to Area Ratio (TAR) is introduced. More specifically TAR is defined as the ratio between normalized throughput (i.e. net throughput multiplied the number of decoding iterations) and the total equivalent gate count. In **Error! Reference source not found.** some comparisons are reported: *B-H* is a fully parallel decoder presented in [Bla02], while *K-P* is a decoder described in [Kan06] and it is based on very similar algorithm simplification. However in [Kan06] only a particular class of regular LDPC codes are dealt with. To enable fair comparisons, a code with a similar number of VN's and CN's, i.e. an (1024, 506) was selected. It was taken from Prof. Mackay's website [McKay]. As it can be observed, the proposed solution is outperformed under the TAR perspective by almost an order of magnitude from *K-P*. Anyway *K-P* is build around a specific class of codes, while the astonishing performance of *B-H* are relative to a single code (not even a class).

4.3.2. Algorithms and Preliminary Architectures For Turbo-LDPC Decoders

Flexibility and hardware reuse in the decoder architecture are based on an adequate and unified algorithmic framework for the decoding of both Turbo and LDPC codes. The identification of a common set of the operators involved in the decoding operations is the crucial step toward the design of a flexible decoder architecture and further of this road, employing the same algorithm for the decoding of the two classes of codes would allow a great advantage. Unfortunately, the common praxis today is to use the BCJR algorithm for the turbo decoding (TD), and message-passing (MP) algorithms for LDPC decoders. Although both belonging to the class of maximum a posteriori (MAP) algorithms, TD and MP result in different – although intrinsically similar – operations in the decoding process. So, the effort is to approach the decoding problem in a different way and define a single algorithm for both turbo and LDPC decoding.

Two approaches may be followed to this aim: one is known as Turbo-Gallager decoding (TG) and is based on the translation of a turbo code into an equivalent parity check matrix; the other is called Turbo-Decoding Message-Passing (TDMP) and dually, looks at an LDPC code as a particular case of a turbo-code and employs the BCJR algorithm for its decoding.

4.3.2.1. Turbo-Gallager (TG) Decoding

Turbo-Gallager codes [Col04] paved the road to the design of a LDPC decoder that could be used for the decoding of both, turbo and LDPC codes. To this aim, a turbo code must be viewed as an LDPC code, and the Trellis representation of Recursive Systematic Convolutional (RSC) codes must be changed into parity-check constraints and a resulting parity check matrix.

The key idea about TG decoding originates from the consideration that an RSC encoder, with its linear feedback shift register structure and its connections or taps, is actually imposing a parity-check constraint on transmitted coded bits. More in detail, let us consider the generator vector of an ordinary RSC code which can be expressed in its general form through the generating polynomial as:

$$\left\{ \begin{array}{l} g(D) = \left[1 + \frac{a(D)}{b(D)} \right] \\ a(D) = \sum_{j=1}^{J_a} D^{\alpha_j} \\ b(D) = \sum_{j=1}^{J_b} D^{\beta_j} \end{array} \right.$$

with J_a and J_b the number of addends of $a(D)$ and $b(D)$, respectively. Let us also refer to u_k and p_k as the bit at the input (information bit) and coming out (parity) the linear feedback shift register at time k ; then the following constraint holds:

$$p_k = \sum_{j=2}^{J_b} p_{k-\beta_j} + \sum_{j=1}^{J_a} u_{k-\alpha_j} \quad (1)$$

where the sum is defined over GF(2). The last equation represents a parity check constraint, so it can be turned in a row of the corresponding parity-check matrix and the final matrix can be easily achieved by applying it to all the K information bits in the codeword (for a total of K rows). There is no restriction to apply this procedure to any RSC code, and in other words, it is always possible to build a parity check matrix starting from an RSC code.

However, good error correction performance of the so-derived LDPC code are not always guaranteed. As an example, if the resulting parity check matrix has cycles of length four, the iterative decoding algorithm could fail to convergence to the correct transmitted codeword. A closer inspection to the equation (1) and considering that a cycle of length four corresponds to a pattern of 4 ones on the

vertices of a rectangle in the parity check matrix, it is possible to derive some guidelines to attain good LDPC codes:

when $J_a = 1$ and $J_b = 2$ ($J_a = 2$ and $J_b = 1$), the parity check matrix has no cycles;

when $J_a = 2$ and $J_b = 2$, the parity check matrix has:

girth four when $\alpha_2 - \alpha_1 = \beta_2 - \beta_1$;

girth six when either $\alpha_2 - \alpha_1 = k(\beta_2 - \beta_1)$ with $k = \{2, 1/2\}$;

girth eight in all the other cases;

when $J_a \geq 2$ and $J_b \geq 2$ the parity check matrix can have cycles of at most length six if all the differences $\alpha_j - \alpha_p$ with $j, p \in [1, J_a]$ and $\beta_n - \beta_m$ with $n, m \in [1, J_b]$ are distinct.

Once an RSC code has been successfully mapped onto an LDPC one, it is straightforward to build the corresponding parity check matrix for a parallel turbo code (PCCC). First of all, the parity check matrix of the second RSC code is exactly the same of the first one but with the column shuffled according to the interleaving law; then the overall matrix of a PCCC merely turns out in the concatenation (row-wise) of the parity check matrices related to the two RSC codes. Also with PCCCs it is crucial to avoid cycles of length four in the resultant parity check matrix, so that the interleaving law must be properly designed and, specifically, a good design choice is represented by S-random interleavers [Div95, Fra99] with $spread\ S = \max_{i,j} \{\alpha_j - \alpha_i\}$ with $i, j \in [1, J_a]$.

4.3.2.2. Turbo- Decoding Message_Passing (TDMP)

Turbo-Decoding Message-Passing (TDMP) was originally proposed by Mansour and Shanbahg in [Man03]. The underlying idea in TDMP is that a codeword of a given LDPC code has to satisfy a particular set of parity-check constraints, specified by the parity-check matrix, so that the sum over GF(2) or the OR of the transmitted bits involved in a particular check node (those corresponding to ones in the parity-check matrix), must be 0 (even parity). Actually it is possible to describe this constraint by means of a 2-states Trellis, in which the single state at the generic stage k , represents the parity after the k -th bit belonging to that constraint. Such a trellis, shown in Fig. 1, has a length equal to the check node degree and the first and last state must be the even parity state (S_E).

Under this hypothesis, particular forward (α) and backward (β) metrics can be defined to calculate the updated check-to-variable messages through the BCJR algorithm. More in detail, let us define:

$$\begin{cases} \alpha_{k+1}(S_E) = \log\left(e^{\alpha_k(S_E)} + e^{\alpha_k(S_O)+\lambda}\right) \\ \alpha_{k+1}(S_O) = \log\left(e^{\alpha_k(S_E)+\lambda} + e^{\alpha_k(S_O)}\right) \\ \beta_k(S_E) = \log\left(e^{\beta_{k+1}(S_E)} + e^{\beta_{k+1}(S_O)+\lambda}\right) \\ \beta_k(S_O) = \log\left(e^{\beta_{k+1}(S_E)+\lambda} + e^{\beta_{k+1}(S_O)}\right) \\ \Lambda = \log\left(e^{\alpha_k(S_E)+\beta_k(S_O)} + e^{\alpha_k(S_O)+\beta_k(S_E)}\right) - \log\left(e^{\alpha_k(S_E)+\beta_k(S_E)} + e^{\alpha_k(S_O)+\beta_k(S_O)}\right) \end{cases}$$

where λ represents the variable to check message related to the bit transmitted bit k ($\lambda = \lambda_{b_k, c_m}$) and Λ the corresponding check variable message ($\Lambda = \lambda_{c_m, b_k}$).

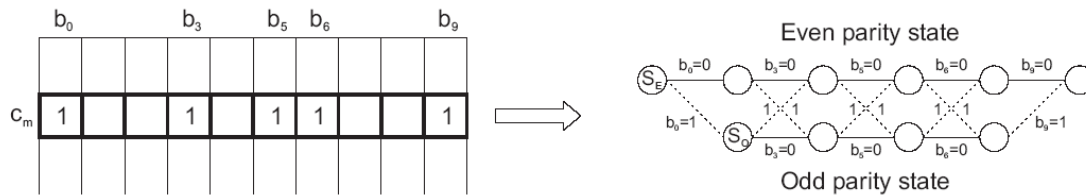


Figure 4-3: 2-state equivalent trellis of a parity-check constraint.

The equations above can be used within the LDPC decoding algorithm to compute the check node operations instead of resorting to the commonly used *sum-product* formulation [Che05]. In such a way, this mathematical approach to the LDPC decoding makes it possible in principle to design one single BCJR SISO decoder to allow the decoding of an ordinary Turbo code and the computation of the check node phase of the LDPC algorithm at the same time. Of course, this must be supported by an appropriate RTL architecture of the SISO engine in order to make it flexible enough to support different codes with different number of states and number of inputs and outputs. However, increased flexibility in the design of a configurable advanced error-correcting codes platform is expected in application of the TDMP approach.

4.3.2.3. VLSI Architectures for Turbo-LDPC Decoders

As remarked in the last two sections, it is possible to either decode an LDPC code as a Turbo code with the TDMP algorithm, or dually, to decode a Turbo code as an LDPC code with the TG algorithm. However, this statement holds with some constraints, since while TDMP can always represent an effective way to perform the check node updates in an LDPC code, on the contrary some issues could arise for the TG algorithm, and guidelines have been provided in Section 4.2.2.1 to control the girth of the corresponding parity check matrix (that must be higher than 4 at least) not to spoil the decoder error correction performance.

Thus, whenever the code is not designed jointly with the decoder, but is given a-priori, TDMP seems to be preferable to TG, whose applicability must be always verified. For this reason the focus is put on TDMP for the design of a flexible Turbo/LDPC decoder.

As a result of this approach, a flexible VLSI architecture for Turbo and LDPC decoding will be composed of a plurality of SISO BCJR decoders, properly connected in order to implement both the TDMP of LDPC codes and the customary Turbo-Decoding. Programmability and flexibility will be achieved through the proper control of configuration multiplexers, which are the means for hardware re-use.

In doing this, the real challenge is represented by the fact that while a regular turbo code employs constituent convolutional codes with 8 or 16 states for good error-correction performance, on the other hand in a TDMP there are only two states in the Trellis, meaning even and odd parity as shown in Section 4.2.2.2.

Thus, the VLSI architecture must be capable of reconfigure the HW resources as to perform 2- or 8-states BCJR decoding, and this is achieved with the proper management of the decoding data flow and the control of the data switches. It may be seen that the HW resources required to implement a 8-state BCJR SISO decoding also allow 4 2-state BJCR updates to be performed in parallel. In other words, it is possible to reach a very high degree of re-use of the HW resources. How deeply this HW sharing can be exploited is still a topic of investigation: from the first explorations it seems that only some parts of the SISO decoder can be re-used (mainly the forward/backward processors). Even if these considerations would be confirmed, it seems that the resulting overhead in the whole architecture can be accepted considering the flexibility of the TDMP-based LDPC decoder.

Although flexibility has been only discussed for the HW resources in terms of operators, multiplexers etc., similar considerations involve memories. Again, the underlying principle is to reuse the memory required by a 2-state or 8-state SISO decoder, in such a way that the memory needed by 4 2-state decoders will be packed into and reused by one 8-state decoder with apparently no overhead.

This preliminary architectural report will be carefully assessed during the next phases of the project.

4.3.2.4. ASIP based TURBO-LDPC decoders

The same approach can be targeted toward ASIP systems instead of dedicated RTL architectures. This ASIP based approach has been described in section 2.1.1.

Although the ASIP design flow allows for a higher degree of flexibility due to the programmable approach of the processor, and in principle very different algorithms could fit in the same processor, this implementation activity can exploits the benefits of the unified algorithmic approach reported in Sections 4.2.2.1 and 4.2.2.2, and particularly of the TDMP. In this way, the ASIP processor can only encompass those operators and resources required by the BCJR algorithm, eventually parametrized for different constraint lengths of the constituents RSC codes. Then, it is only a matter of properly programming the processor in order to attain LDPC or Turbo decoding.

The University of Pisa has also reached an agreement with the RWTH University of Aachen for the implementation of a mixed LDPC/Turbo decoder ASIP. The design will be carried on in the University of Pisa by exploiting proprietary CAD and simulation tools in order to assess the algorithmic performance of the decoder and dimension the internal data-paths of its finite-precision arithmetic architecture.

Also, a reference scenario, IEEE.802.16e seems to be very attractive candidate, due to the large number of FEC solutions foreseen by the standard, which include convolutional codes, turbo codes, block turbo codes and LDPC codes.

5. CONCLUSIONS

This document discussed key research aspects of the digital hardware design for forefront wireless technology implementation. Due to the stringent requirements on computational power and flexibility, NEWCOM⁺⁺/WPRC intends to tackle parallel architectures like MP-SoS and NoC and dedicated processors for digital communication such as ASIPs.

The design of efficient architecture needs an efficient design space exploration methodology, and efficient tools for specifying and designing advanced architectures was discussed too, with an emphasis on ASIP design.

Besides, due to the need for flexibility, and to the inefficiency of pure software radio approaches, a mixed SW / HW based flexible radio design concept was addressed. This approach, which consists in identifying common operators that can be reused intensively in several transceiver instantiations was discussed and will be further investigated.

Finally, extremely computationally intensive operators such as LDPC/Turbo decoders and FFTs that are required in a large portion of advanced communication system were addressed. To tackle these blocks, parallel architecture are considered and benefited from.

This state of the art document has clearly showed that many open issues need to be tackled in terms of flexible design for advanced communication system. This will be the core activity of WPRC in the future.

6. REFERENCE

- [3GPP05] Technical Specification Group Radio Access Network (3GPP), “TS 25.212 V6.4.0 – multiplexing and channel coding (FDD),” March 2005, www.3gpp.org
- [Agu07] C. Aguayo, F. Portelinho, J. Reed, “Design and Implementation of an SCA core framework for a DSP platform”, *Military Embedded Systems*, March 2007
- [Ala08a] Laurent Alaus, Dominique Noguét, Jacques Palicot, « A Reconfigurable Linear Feedback Shift Register for Software Defined Radio Terminal », ISWPC2008, Santorin, Greece, 6-9 May 2008.
- [Ala08b] Laurent Alaus, Dominique Noguét, Jacques Palicot, « A Extended Reconfigurable Linear Feedback Shift Register for Software Defined Radio Terminal », ISSSTA2008, Bologna, Italia, 25-28 August 2008.
- [ARC] ARC Configurable Cores Homepage, <http://www.arc.com/configurablecores/>.
- [Bas01] S. Basu, “High Level Synthesis from Sim-nML Processor Specifications,” Master’s thesis, Indian Institute of Technology, Kanapur, August 2001.
- [Bei06] E. Beigne, P. Vivet, Design of On-chip and Off-chip Interfaces for a GALS NoC Architecture, Proceedings of 12th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC’06), Grenoble, France, pp. 172-181, March 2006.
- [Bei08] E. Beigné, F. Clermidy, S. Miermont, A. Valentian, P. Vivet, S. Barasinski, F. Blisson, N. Kohli, S. Kumar, « A Fully Integrated Power Supply Unit for Fine Grain DVFS and Leakage Control Validated on Low-Voltage SRAMs”, ESSCIRC’2008, Edinburg, UK, Sept 2008
- [Ben01] L. Benini and G. Micheli, “Networks on Chips: A New SoC Paradigm,” *IEEE Computer*, vol. 35, no. 1, Jan. 2001, pp. 70-78.
- [Ben06] L. Benini “Application specific NoC design”, In Proceedings of the Conference on Design, Automation and Test in Europe: Proceedings Munich, Germany, March 06 - 10, 2006
- [Ber02] J. Bertrand, J. W. Cruz, B. Majkrzak, T. Rossano, “CORBA Delays in a Software-Defined Radio”, *IEEE Communications Magazine*, Feb. 2002
- [Ber04] David Bertozzi and Luca Benini, Xpipes: A Network-on-Chip Architecture for GigascaleSystems-on-Chip, *IEEE circuits and systems magazine*, pp 18-31, second quarter 2004
- [Ber05] P. Bernardi, G. Masera, F. Quaglio, and M. Sonza Reorda, “Testing logic cores using a BIST P1500 compliant approach: a case of study,” in Proc. Design, Automation and Test in Europe Conference and Exhibition (DATE 2005), vol. 3, Mar. 2005, pp. 228-233.
- [Bje05] T. Bjerregaard, S. Mahadevan, R. Grøndahl Olsen and J. Sparsø, An OCP Compliant Network Adapter for GALS-based SoC Design Using the MANGO Network-on-Chip, Proceedings of the International Symposium on System-on-Chip (SoC’05), pp. 171-174, 2005.
- [Bla02] A. J. Blanksby and C. J. Howland, “A 690-mW 1-Gb/s 1024-b, Rate ½ Low-Density Parity-Check Code Decoder,” *IEEE J. Solid-State Circuits*, vol. 37, no. 3, pp. 404–412, Mar. 2002.
- [Bra06] T. Brack, F. Kienle, and N. Wehn, “Disclosing the LDPC code decoder design space,” in *Proc. DATE 2006*, vol. 1, Mar. 2006, pp. 200–205.
- [Bru46] N.G. de Bruijn, “A combinatorial problem,” *Koninklijke Nederlandse Akademie v. Wetenschappen*, vol. 49, pp. 758–764, 1946.
- [Cha03] A. Chakraborty, M. Greenstreet, Efficient Self-Timed Interfaces for Crossing Clock Domains, Proceedings of 9th International Symposium on Asynchronous Circuits and Systems (ASYNC’2003), pp. 78-88, Vancouver, Canada, 2003.
- [Cha06] Wei-Hsin Chang, and Truong Nguyen, “An OFDM-Specified Lossless FFT Architecture”, *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS: REGULAR PAPERS*, VOL. 53, NO. 6, JUNE 2006.
- [Che00] T. Chelcea, S. Nowick, Low-latency asynchronous FIFO’s using token rings, Proceedings of International Symposium on Advanced Research in Asynchronous Circuits and Systems, pp. 210-220, April 2000.
- [Che05] Jinghu Chen, Ajay Dholakia, Evangelos Eleftheriou, Marc P. C. Fossorier, Xiao-Yu Hu, “Reduced-Complexity Decoding of LDPC Codes”, *IEEE Transactions on Communications* 53(8): 1288-1299 (2005)
- [Col04] G. Colavolpe, “Design and Performance of Turbo Gallager Codes,” *IEEE Trans. Commun.*, vol. 52, pp. 901–1908, Nov 2004.
- [Coo65] Cooley, J., and Tukey, J., “An algorithm for the machine computation of the complex Fourier series”, *Math. Comput.*, 1965, 19, pp. 297-301.

- [CoWare] CoWare Processor Designer Homepage, <http://www.coware.com/products/processor designer.php>.
- [Dae05] Daeho Seo, Akif Ali, Won-Taek Lim, Nauman Rafique, Mithuna Thottethodi, "Near-Optimal Worst-case Throughput Routing for Two-Dimensional Mesh Networks", Proceedings of the 32nd International Symposium on Computer Architecture (ISCA'05), 2005
- [Dar05] A. Darabiha, A. C. Carusone, and F. R. Kschischang, "Multi-Gbit/sec low density parity check decoders with reduced interconnect complexity," in *Proc. IEEE ISCAS 2005*, vol. 5, May 2005, pp. 5194–5197.
- [Din06] L. Dinoi, R. Martini, G. Masera, F. Quaglio, and F. Vacca, "ASIP design for partially structured LDPC codes," *Electronics Letters*, vol. 42, no. 18, pp. 1048–1049, 2006.
- [Div95] D. Divsalar and F. Pollara, "Turbo codes for PCS applications," in *Proc. IEEE Int. Conf. Communications*, Seattle, WA, Jun. 1995, vol. 1, pp. 54–59.
- [Dob04] R. Dobkin, R. Ginosar, C. Sotiriou, Data Synchronization Issues in GALS SoCs, Proceedings of the 10th International Symposium on Asynchronous Circuits and Systems, pp. 170-179, Crete, Greece, 19 - 23 April 2004.
- [Dob07] Rostislav (Reuven) Dobkin, Ran Ginosar and Israel Cidon QNoC Asynchronous Router with Dynamic Virtual Channel Allocation, First NoC Symposium, 2007
- [Fab99] T. Faber, M. Schönle, "DSP-platform target report," SLATS Consortium, Project no. 27016, Deliverable D23, Dec. 1999.
- [Fau95] A. Fauth, J. V. Praet, and M. Freericks, "Describing Instruction Set Processors Using nML," Proceedings of the European Design and Test Conference (EDTC), pp. 503–507, 1995.
- [Fra99] C. Fragouli and R. D. Wesel, "Semi-random interleaver design criteria," in *Proc. Communications Theory Symp. Globecom*, Rio de Janeiro, Brazil, Dec. 1999, vol. 5, pp. 2352–2356.
- [Fra06] C. Francalanci, and P. Giacomazzi, "High-performance self-routing algorithm for multiprocessor systems with shuffle interconnections," *IEEE Transactions on Parallel and Distributed Systems*, pp. 38-50, January 2006.
- [Gil03] F. Gilbert, M. Thul and N. Wehn "Communication Centric Architectures for Turbo-Decoding on Embedded Multiprocessors", in *Proc. of Design, Automation and Test in Europe (DATE) Conference 2003*, pages 356-361, Munich, March 2003.
- [God08] L. Godard, C. Moy, J. Palicot, "A simulator for the design of the management architecture of cognitive radio equipments", 5th Karlsruhe Workshop on Software Radios, WSR'08, Karlsruhe, Germany, March 2008
- [God09] L. Godard, C. Moy, J. Palicot, "An Executable Meta-Model of a Hierarchical and Distributed Architecture Management for the Design of Cognitive Radio Equipments", *Annales des Télécommunications*, Special issue on Cognitive Radio, to be published in 2009
- [Goo05] K. Goossens, J. Dielissen, A. Radulescu, *ETHERAL architectures, concepts, architectures and implementations*, Special NoC session, IEEE Design&Test of Computers, pp. 414 - 421, Sept-Oct 2005.
- [Gue00] P. Guerrier and A. Greiner, A generic architecture for on-chip packet switched interconnections, *Design, Automation and Testing in Europe DATE00*, pp. 250–256, Mar. 2000.
- [Gui07] [F. Guilloud, E. Boutillon, J. Tusch, and J. Danger, "Generic description and synthesis of LDPC decoders." IEEE Trans. Commun., accepted for publication. \[Online\]. Available: http://web.univ-ubs.fr/lester/boutillon/ldpc/ldpc.htm, 2007](http://web.univ-ubs.fr/lester/boutillon/ldpc/ldpc.htm)
- [GUO06] Yuanbin Guo, Jianzhong(Charlie) Zhang, Dennis McCain, and Joseph R. Cavallaro, "An Efficient Circulant MIMO Equalizer for CDMA Downlink: Algorithm and VLSI Architecture", *EURASIP Journal on Applied Signal Processing* Volume 2006, Article ID 57134.
- [Hac07] R. Hachemani, J. Palicot, C. Moy, "A New Standard Recognition Sensor for Cognitive Radio Terminal", *EUSIPCO'07*, Poznan, Poland, 3-7 Sept. 2007
- [Had02] G. Hadjiyiannis and S. Devadas, "Techniques for Accurate Performance Evaluation in Architecture Exploration," *IEEE Transactions on VLSI Systems*, 2002.
- [Had97] G. Hadjiyiannis, S. Hanono, and S. Devadas, "ISDL: An Instruction Set Description Language for Retargetability," in *Design Automation Conference*, pp. 299–302, 1997.
- [Hal99] A. Halambi and P. Grun, "Expression: A language for architecture exploration through compiler/simulator retargetability," *Proceedings of the European Conference on Design, Automation and Test (DATE) 1999*, March 1999.
- [Hel01] M. Helard, R. Le Gouable, J.F. Helard, J.Y. Baudais, "Multicarrier CDMA Techniques for Future Wideband Wireless Networks", *Annals of Telecom*, 56, n°5-6, May and June 2001.

- [Hoch03] B. M. Hochwald and S. ten Brink, "Achieving near-capacity on a multiple-antenna channel," *IEEE Trans. Commun.*, vol. 51, no. 3, pp. 389–399, Mar. 2003
- [Hoc03] D. E. Hocevar, "LDPC code construction with flexible hardware implementation," in *Proc. IEEE ICC 2003*, vol. 4, May 2003, pp. 2708–2712.
- [Hof02] A. Hoffmann, H. Meyr, and R. Leupers, "Architecture Exploration for Embedded Processors with LISA", Springer, 1st edition November 2002.
- [ITRS] The International Technology Roadmap for Semiconductors, annual reports available on www.itrs.net
- [Jon02] F. Jondral, "Parameterization-a technique for SDR Implementation" Chapter 8 of "Software Defined Radio Enabling Technologies" edited by W. Tuttlebee, Wiley, 2002.
- [Kan06] S.-H. Kang and I.-C. Park, "Loosely Coupled Memory-Based Decoding Architecture for Low Density Parity Check Codes," *IEEE Trans. Circuits Syst. I*, vol. 53, no. 5, pp. 1045–1056, May 2006.
- [Kar] [George Karypis, "METIS" http://glaros.dtc.umn.edu/gkhome/views/metis](http://glaros.dtc.umn.edu/gkhome/views/metis)
- [Kar05] K. Karuri, M. A. Al Faruque, S. Kraemer, R. Leupers, G. Ascheid, and H. Meyr, "Fine-grained Application Source Code Profiling for ASIP Design", in *Proc. of the Design Automation Conference (DAC)*, (Anaheim), June 2005.
- [Kar08] K. Karuri, A. Chattopadhyay, X. Chen, D. Kammler, L. Hao, R., Leupers, H. Meyr, and G. Ascheid "A Design Flow for Architecture Exploration and Implementation of Partially Reconfigurable Processors", *IEEE Transactions on Very Large Scale Integrated Systems*, Volume: 16, Issue: 10, Oct. 2008.
- [Kay99] D. J. C. MacKay, "Good Error-Correcting Codes Based on Very Sparse Matrices," *IEEE Trans. Inform. Theory*, vol. 45, no. 2, pp. 399–431, Mar. 1999.
- [Kie03] F. Kienle, M. J. Thul, and N. Wehn, "Implementation Issue of Scalable LDPC-Decoders," in *Proc. 3rd Int. Symp. on Turbo Codes & Related Topics*, Brest, France, Sept. 2003, pp. 291–294.
- [Kie03] F. Kienle, M. J. Thul, and N. Wehn, "Implementation issues of scalable LDPC-decoders," in *Proc. 3rd International Symposium on Turbo Codes and Related Topics*, Brest, France, September 2003.
- [Kie03] S. Kim, J. Masse, S. Hong, "[Dynamic Deployment of Software Defined Radio Components for Mobile Wireless Internet Applications](#)", *Proceedings of International Human, Society and Internet*, 2003
- [Lao05] Christophe Laot, Raphael Le Bidan, "Low-Complexity MMSE Turbo Equalization: A Possible Solution for EDGE", *IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS*, VOL. 4, NO. 3, MAY 2005.
- [Lat07] D. Lattard, et al. "A Telecom Baseband Circuit-Based on an Asynchronous Network-on-Chip", *International Solid State Circuit Conference, ISSCC'2007*, San Francisco, USA, Feb 2007.
- [Lat08] D. Lattard, E. Beigne, F. Clermidy, Y. Durand, R. Lemaire, P. Vivet, F. Berens, "A Reconfigurable Baseband Platform Based on an Asynchronous Network-on-Chip", *IEEE Journal Of Solid State Circuits*, Vol. 43, Issue 1, pp. 223-235, Jan. 2008.
- [Lee06] J. Lee, S. Kim, S. Hong, "Q-SCA: QoS Enabled JTRS Software Communications Architecture for SDR-based Wireless Handsets", *Real-Time Systems*, Volume 34, Issue 1, pg. 19-35, Sept. 2006
- [Leu06a] R. Leupers, "From ASIP to MPSoC - Architectures and Design Tools for Communication and Multimedia Systems". In *Computer Engineering Colloquium at TU Delft*, Dec 2006.
- [Leu06b] R. Leupers, K. Karuri, S. Kraemer, and M. Pandey, "A Design Flow for Configurable Embedded Processors based on Optimized Instruction Set Extension Synthesis", in *Proc. of the Design, Automation and Test in Europe Conference and Exhibition (DATE)*, (Munich), March 2006.
- [Lin07] Yu-Wei Lin and Chen-Yi Lee, "Design of an FFT/IFFT Processor for MIMO OFDM Systems", *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS: REGULAR PAPERS*, VOL. 54, NO. 4, APRIL 2007.
- [Iac05a] Daniele Lo Iacono, Julien Zory, Ettore Messina, and Nicolo' Piazzese, "Block processing engine for high-throughput wireless communications", *2nd International Symposium on Wireless Communication Systems*, 2005.
- [Iac05b] Daniele Lo Iacono, Ettore Messina, Costantino Volpe, and Arnaldo Spalvieri, "Serial Block Processing for Multi-Code WCDMA Frequency Domain Equalization", *IEEE Wireless Communications and Networking Conference(WCNC)*, 2005.
- [Man03] M. M. Mansour and N. R. Shanbhag, "High Throughput LDPC Decoders," *IEEE Trans. VLSI Syst.*, vol. 11, no. 6, pp. 976–996, Dec. 2003.
- [Mas05a] G. Masera, "Flexible hardware for wireless communications: a case of study," in *Proc. URSI XI National Symp. of Radio Science*, Apr. 2005.

- [Mas05b] G. Masera, F. Quaglio, and F. Vacca, "Finite precision implementation of LDPC decoders," *IEE Proc. Communications*, vol. 152, no. 6, pp. 1098–1102, Dec. 2005.
- [Mas07] Masera, G.; Quaglio, F.; Vacca, F.; Implementation of a Flexible LDPC Decoder, *IEEE Transactions on Circuits and Systems II*, Volume 54, Issue 6, June 2007 Page(s):542 546
- [Mas07] G. Masera, F. Quaglio, and F. Vacca, "Implementation of a flexible LDPC decoder," *IEEE Transactions on Circuits and Systems*, pp. 542-546, June 2007.
- [McKay] [D. J. C. MacKay. Encyclopedia of sparse graph codes. \[Online\]. Available: http://www.interference.phy.cam.ac.uk/mackay/codes/data.html](http://www.interference.phy.cam.ac.uk/mackay/codes/data.html)
- [Mei05] B. Mei, A. Lambrechts, D. Verkest, J. Mignolet and R. Lauwereins, "Architecture Exploration for a Reconfigurable Architecture Template", *IEEE Design and Test*, vol. 22, no. 2, pp. 90 – 101, 2005.
- [Mil04] Mikael Millberg, Erland Nilsson, Rikard Thid, Shashi Kumar, and Axel Jantsch, The Nostrum Backbone - a Communication Protocol Stack for Networks on Chip, *Proceedings of the 17th International Conference on VLSI Design (VLSID'04)*, 2004.
- [MIPS] MIPS CorExtend Homepage: <http://www.mips.com/products/processors/32-64-bit-cores/pro-series-family/>.
- [Mit00] J. Mitola, "Cognitive Radio: An Integrated Agent Architecture for Software Defined Radio", Ph.D. dissertation, Royal Inst. of Tech., Sweden, May 2000
- [Mit95] J. Mitola, "The software Radio Architecture", *IEEE Communications Magazine*, May 95, pp. 26-38.
- [Mit99] J. Mitola, G. Maguire, "Cognitive radio: making software radios more personal, *Personal Communications*", *IEEE [see also IEEE Wireless Communications]*, Vol. 6, No. 4. (1999), pp. 13-18
- [Mou07] Hazem Moussa, Olivier Muller, Amer Baghdadi, Michel Jézéquel, Butterfly and Beneš-Based on-Chip Communication Networks for Multiprocessor Turbo Decoding, *DATE Conference*, 2007
- [Mou08] H. Moussa, A. Baghdadi, M. Jézéquel, "Binary de Bruijn interconnection network for a flexible LDPC/turbo decoder", in *proceedings of the IEEE International Symposium on Circuits and Systems*, May 18-21, Seattle, USA, 2008.
- [Moy05] C. Moy, A. Bisiaux, S. Paquelet, "An Ultra-Wide Band Umbilical Cord for Cognitive Radio Systems", *PIMRC'05*, Berlin, September 2005, vol 2, pp. 775-779
- [Mul05] P.A. Muller, F. Fleurey, J.M. Jezequel, "Weaving executability into object-oriented metalanguages", *LNCS, Montego Bay, Jamaica, MODELS/UML'2005*, Springer (2005)
- [Mul06a] O. Muller, A. Baghdadi, M. Jézéquel, "ASIP-Based Multiprocessor SoC Design for Simple and Double Binary Turbo Decoding", *DATE'06*, March 2006.
- [Mul06b] O. Muller, A. Baghdadi, M. Jézéquel, "Exploring Parallel Processing Levels for Convolutional Turbo Decoding", *ICTTA'06*, April 2006.
- [Mul06c] O. Muller, A. Baghdadi, and M. Jezequel, "On the Parallelism of Convolutional Turbo Decoding and Interleaving Interference", *GLOBECOM'06*, November 2006.
- [Mut00] J. Muttersbach, T. Villiger, W. Fichtner: "Practical Design of Globally-Asynchronous Locally-Synchronous Systems", *Proceedings of the Sixth International Symposium on Advanced Research in Asynchronous Circuits and Systems, ASYNC'2000*, Eilat, Israel, pp. 52-59, April 2-6, 2000.
- [Nae08] Naessens, Frederik; Bougard, Bruno; Bressinck, Siebert; Hollevoet, Lieven; Raghavan, Praveen; Van der Perre, Liesbet; Catthoor, Francky, "A unified instruction set programmable architecture for multi-standard advanced forward error correction", *IEEE Workshop on Signal Processing Systems*, 2008. SiPS 2008. 8-10 Oct. 2008 Page(s):31 – 36
- [Nog08] D. Noguet et al., "An MC-SS Platform for Short-Range Communications in the Personal Network Context" *EURASIP Journal on Wireless Communications and Networking*, Volume 2008, Article ID 830273, 2008.
- [Noll] T. Noll, EECS, RWTH Aachen, <http://www.eecs.rwth-aachen.de/>.
- [OMG01] OMG Unified Modeling Language Infrastructure Specification, version 2.0, September 2001. Document ptc/03-09-15, available at <http://www.omg.org/>.
- [OSSIE] OSSIE, MPRG, Wireless group at Virginia Tech, <http://ossie.wireless.vt.edu/trac>
- [Ora3.3] ORACLE deliverable D3.3 "Description of the transceiver architecture and performance results". Public document available on www.ist-oracle.org
- [Ora3.4] ORACLE deliverable D3.4 "Overall complexity guidelines of an OR Terminal". Public document available on www.ist-oracle.org.

- [Pal03] J. Palicot, C. Roland, "FFT: a Basic Function for a Reconfigurable Receiver", ICT' 2003, February 2003, Papeete, Tahiti.
- [Pee99] S. Pees, A. Hoffmann, V. Zivojnovic, and H. Meyr, "LISA-Machine Description Language for Cycle-Accurate Models of Programmable DSP Architectures," in Proc. of the Design Automation Conference (DAC), (New Orleans), June 1999.
- [Qua05] F. Quaglio, F. Vacca, and G. Masera, "Low Complexity, Flexible LDPC Decoders," in *Proc. 14th IST Mobile Summit 2005*, June 2005.
- [Qua06] F. Quaglio, F. Vacca, C. Castellano, A. Tarable, and G. Masera, "Interconnection Framework for High-Throughput, Flexible LDPC Decoders" in Proc. Design, Automation and Test in Europe Conference and Exhibition, 2006.
- [Raj99] V. Rajesh and R. Moona, "Processor modeling for hardware software codesign," International Conference on VLSI Design, January 1999.
- [Ree05] X. Reves, V. Marojevic, R. Ferrus, A. Gelonch, "FPGA's Middleware for Software Defined Radio Applications", *FPL 2005*
- [Rhi02] Arnd-Ragnar Rhiemeier, "Benefits and Limits of Parameterized Channel Coding for Software Radio", 2nd Karlsruhe Workshop on Software Radios, Germany, March 2002.
- [Rol03] C. Roland, J. Palicot, "A new Concept of Wireless Reconfigurable Receiver", IEEE Com. Mag. July 2003
- [Saa03] I. Saastamoinen, M. Alho, and J. Nurmi, "Buffer implementation for Proteo network-on-chip," ISCAS, 2003, pp. 113–116.
- [Sic02] C. Siclet, P. Siohan, D. Pinchon, "Oversampled orthogonal and biorthogonal multicarrier modulations with perfect reconstruction", DSP '00, Santorini, July 2002.
- [Str] Stretch Software-Configurable Processors Homepage, <http://www.stretchinc.com/technology/>.
- [Tan04] R. Tanner, J. Woodard, WCDMA – Requirements and Practical Design. John Wiley & Sons Ltd, 2004.
- [Tar04] A. Tarable, S. Benedetto, and G. Montorsi, "Mapping interleaver laws to parallel turbo and LDPC decoders architectures," *IEEE Trans. Inform.Theory*, vol. 50, no. 9, pp. 2002–2009, Sept. 2004.
- [Target] Target IP Designer Homepage, <http://www.retarget.com/resources.php>.
- [Tens] Tensilica Xtensa 7 Homepage, http://www.tensilica.com/products/x7_processor_generator.htm.
- [The05] T. Theocharides, G. Link, N. Vijaykrishnan, and M. J. Irwin, "Implementing LDPC decoding on a network-on-chip," in Proc. of the International Conference on VLSI Design, pp. 134-137, January 2005.
- [TI] [Texas Instruments \(TI\) Website. www.ti.com](http://www.ti.com)
- [Tra07] X.-T. Tran, J. Durupt, Y. Thonnart, F. Bertrand, V. Beroulle, C. Robach, "Implementation of a Design-for-Test Architecture for Asynchronous Networks-on-Chip", In Proc. of International Conference on Network-on-Chip, NoCS'07, Princeton, USA, May 2007.
- [Tut95] W. Tuttlebee, "Evolution of radio systems into the 21st century", Proc. IEE Int. Conf. on 'Radio receivers and associated systems', Bath, United-Kingdom, 25-27 September 1995.
- [Val07] Alexandre Valentian, "Gate Bias Circuit for an SCCMOS Power Switch achieving maximum leakage reduction", ESSCIRC 2007, 11-13 September, Munich, Germany.
- [Vil04] A. I. Vila Casado, W.-Y. Weng, and R. D. Wesel, "Multiple Rate Low-Density Parity-Check Codes with Constant Blocklength," in *Proc. 38th Asilomar Conf. on Signals, Systems and Computers*, vol. 2, Monterey, USA, Nov. 2004, pp. 2010–2014.
- [YAV68] R. Yavne, "An economical method for calculating the discrete Fourier transform," in Proc. AFIPS Fall Joint Computer Conf. 33, 1968, pp. 115-125.
- [Yun96] K. Yun, R. Donohue, Pausible Clocking: A first step toward heterogeneous systems, Proceedings of International Conference on Computer Design (ICCD), October 1996.
- [Ziv96] V. Zivojnovic, S. Pees, and H. Meyr, "LISA – machine description language and generic machine model for HW/SW codesign," in Proc. of the IEEE Workshop on VLSI Signal Processing, (San Francisco), Oct. 1996.