



MORPHEUS

D8.2 Report on Course material (phase 1)

CONTRACT NO MORPHEUS IST 027342
TYPE OF DOCUMENT D8.2
DATE 21/12/2006
ABSTRACT This deliverable presents the contents of the planned courses for reconfigurable computing in general and MORPHEUS in detail. The courses are based on the results of deliverable D8.1.
AUTHOR, COMPANY Florian Thoma, UK
Philippe Bonnot, TRT
Eberhard Schüler, PACT
Bernard Pottier, UBO
Axel Schneider, Lucent
Antonio Deledda, ARCES
Stelios Perissakis, ICOM
WORKPACKAGE WP8
CONFIDENTIALITY LEVEL PU
FILING CODE MORPHEUS-D8.2-UK-R1.4-UK.doc

DOCUMENT HISTORY

<u>Release</u>	<u>Date</u>	<u>Reason of change</u>	<u>Status</u>	<u>Distribution</u>
R1.1	06/11/2006	Creation	On-going	WP8
R1.2	06/12/06	Draft	On-going	WP8 + Contributors
R1.3	06/12/06	Final Draft for internal Review	finished	reviewers
R1.4	22/12/06	Final version	Final	EC, website



Table of Contents

1. INTRODUCTION	2
2. EXECUTIVE SUMMARY	2
3. PATH TO COURSE SPECIFICATION	3
3.1. REQUIREMENTS	3
3.2. EXISTING TRAINING OFFERS	4
3.3. PROPOSED COURSES	6
4. COURSES FOR HW-DESIGNERS/ENGINEERS	8
4.1. COARSE-GRAINED RECONFIGURABLE ARCHITECTURES	8
4.1.1 (ARCES)	8
4.1.2 (UBO)	9
4.2. HW/SW INTERACTION AND DESIGN TESTABILITY	10
4.2.1 (TUD)	10
5. COURSES FOR SW-DESIGNERS/ENGINEERS	11
5.1. PARALLEL COMPUTING ON RECONFIGURABLE ARCHITECTURES	11
5.1.1 (ARCES)	11
5.1.2 (UK)	13
5.1.3 (UBO)	13
5.2. HW/SW-COMMUNICATION IN HETEROGENEOUS RECONFIGURABLE SOC	15
5.2.1 (UK)	15
5.2.2 (UBO)	16
6. COURSES FOR BOTH HW AND SW DESIGNERS/ENGINEERS	18
6.1. HARDWARE-SOFTWARE CO-DESIGN FOR RECONFIGURABLE SOCS (UK)	18
6.2. METHODOLOGY OF RECONFIGURATION IN MORPHEUS ARCHITECTURE – TRADE-OFF AND BENEFITS (CEA)	19
6.3. SIMULATION AND ANALYSIS OF HETEROGENEOUS SOC (ARCES)	19
6.4. EXPERIENCES AND LESSONS LEARNED WHILE IMPLEMENTING DEMONSTRATOR APPLICATIONS	19
6.4.1 <i>Telecommunication standards</i>	19
6.4.1.1 <i>Optical networking domain (LUCENT)</i>	19
6.4.1.2 <i>Wireless Communication (ICOM)</i>	20
6.4.2 <i>Video data processing</i>	20
6.4.2.1 <i>HD video (PACT)</i>	20
7. COURSES FOR MORPHEUS END USERS	21
7.1. USER GUIDE TO THE MORPHEUS RECONFIGURABLE SOC PLATFORM	21
7.1.1 (TRT)	21
7.1.2 (LUCENT)	23
8. CONCLUDING SECTION	24
8.1. CONCLUSION	24
8.2. ACRONYMS	24

1. Introduction

Current System design include both software and hardware components to fulfil the requirements of real-time applications. In the MORPHEUS project the focus is on reconfigurable computing and extends traditional SoCs with heterogeneous reconfigurable hardware. The combination of control flow oriented architectures, like microprocessors and reconfigurable parallel processing architectures like FPGA, XPP and PiCoGA, enlarges the design space for system designers and is a challenging task for the provided design flow. To ensure the optimised exploitation of the targeted heterogeneous architecture in MORPHEUS, an adequate training must be provided from the academic partners focused to the requirements of the industrial partners. In the document D8.1 "Industrial needs and course specification", the needs of the industry and current offers of the academic partners were assessed. This assessment led to the definition of nine courses, which will provide the necessary skills for a successful dissemination of reconfigurable computing in general and MORPHEUS in special. The content and training material for each course will be detailed in this document.

2. Executive summary

The MORPHEUS project aims to improve traditional SoCs with heterogeneous reconfigurable hardware to achieve reconfigurable computing. The new hardware architecture will be supported by a rich set of design-tools. As the ideas and concept of reconfigurable computing are not well known in the community it is necessary for the project to provide training on generic and MORPHEUS specific training.

This document bases on the previous work presented in deliverable D8.1 "Industrial needs and course specification". The specification of the content of the proposed training courses for the MORPHEUS project is the content of this document. Each course has a description of its intended content in an appropriate level of detail for this stage of the project.

3. Path to course specification

This section retraces the steps to the course specification of Deliverable 8.1 "Industrial needs and course specification". In some cases there had to be made some minor adjustments in the specification of the courses.

3.1. Requirements

All training requirements from the industry partners of the project have been analysed and the main topics were collected to a list in order to identify a unified training and to prepare the course material for the MORPHEUS project.

The list is separated in two columns. The first column serves as an index for later processing (see section 3.3). The second column notes the idea (or requirement) for training.

Nr.	Idea / Training requirement
1	FPGA designers: Computation power of coarse-grain modules; simplification brought by the memory hierarchy; efficiency brought by the on-chip network; flexibility offered by the dynamic reconfiguration management; design productivity made possible thanks to tool modules offered to implement functions on reconfigurable units; skill corresponding to the possibilities offered by MORPHEUS and not offered by FPGA
2	Software designers: Speed-up brought by the Molen paradigm making use of the reconfigurable units offered by the architecture; easiness of implementing functions on reconfigurable units; another vision than the classical von Neumann paradigm; Software has to be aware, which reconfiguration part currently is active e.g. in order to cope with configuration dependent register maps
3	Training with different technologies within MORPHEUS: PACT XPP, ARCES PicoGA, M2000 FlexOS architectures and tools, Molen compilation, Dynamic reconfiguration RTOS, RC-SPEAR and MADEO design tools
4	For engineers in general: Aware of both software and hardware issues, convinced of the interest of the good flexibility/performance compromise solution that is offered by reconfigurable computing
5	Software designers must be trained in hardware understanding on system level and step by step mapping to the architecture
6	Software designers must be able and willing to leave high abstraction levels. Training need in profiling and native programming
7	Software designers must be trained in Multi tasking and synchronisation of tasks and data transfers
8	Software designers must be trained in how to write efficient drivers and how to specify the hardware for it
9	Software designers should know how to specify external IP. Training need in structuring of specifications and level of details and acceptance procedures for outsourcing tasks
10	Project handling: how to avoid iteration "loops" because of misunderstandings or unclear specifications, how to check the plausibility of time estimations and how to steer complex projects
11	Hardware designers must be trained in translation of software demands into hardware requirements and calculate the "costs" of those requirements
12	Hardware designers must be trained in modular and testable hardware design and specification of test suites
13	Hardware designers must be trained in mapping of algorithms to parallel arrays of processing elements (coarse and fine grained)

14	Hardware designers must cooperate with software designers in order to estimate how efficient driver software is designed and which hardware prerequisites are required. Training need: Driver software hardware parameterization
15	Extension of existing tutorials: Methods for partitioning of algorithms, HW / SW Co-Design, Graph Algorithm, parallelism on data, task and instruction level
16	Extension of existing tutorials: Methods for optimal memory usage in a heterogeneous architecture
17	Extension of existing tutorials: Examples for DMA setup and buffer handling
18	Extension of existing tutorials: Optimisation methodologies with respect to power, area and performance
19	Awareness of the cost of reconfiguration to take the best advantage of this mechanism
20	Methodology of conception and high level modelling of complex SOC architecture
21	Analysis of the significant impact on the efficiency and flexibility of the final implementation caused by communication mechanisms and physical interconnection
22	Training on platform analysis tools that results may be interpreted correctly and bottlenecks can be identified (performance analysis)
23	Basic concept of FPGA architecture and the functionalities provided to designers. Methodology and the mode of use, for both hardware and software
24	Concept and exploitation of dynamic reconfiguration. Benefits, trade-off and methodology
25	Methodology training for MORPHEUS as a whole, with a focus on exploiting reconfigurability, Practical use of the MORPHEUS platform
26	Basic knowledge of HDL coding, Bus-standards and Network-on-Chip paradigm
27	Understanding of telecommunication standards
28	Handling and programming of design tools like simulators, simulation environment, synthesis and analysis tools
29	Software development for reconfigurable systems
30	Video Data Processing

Table 1: Summary of training requirements from industry

3.2. Existing training offers

All academic partners of WP8 and some of the industrial partners of the project presented their existing and future training offers in the actual state. The goal was to identify existing training offers to satisfy the training requirements listed in section 3.1. This led to a collection of 36 courses, which cover topics identified in the previous section. The list consists of two columns. In the first column the partner is listed who offers the training. In the mean time, this list could be extended to 41 courses which are listed below.

Section	Training Offer
PACT	Tutorials and examples for the tools and the XPP-III architecture
PACT	Examples and documentation on all complexity levels for PACT XPP-III (also product documentation)
CEA-LIST	Configuration manager for dynamic reconfiguration mechanisms as a case study for dynamic management of reconfigurable resources in SOC
ACE	Compiler development system CoSy training material
ST	Training course on XiRISC-PiCoGA-GriffyC including practical examples. Maybe

	extension to XPP and M2000
Suggested by INTRACOM	“System Level Design of Reconfigurable Systems-on-Chip” Springer Book, containing details on how future systems could benefit from the use of reconfigurable technologies
LUCENT	Ethernet Basics
ARCES	Introduction to the configurable technology, showing the advantages of spatial design with respect to temporal computation
ARCES	Parallel computing concepts
ARCES	Training on XiRisc processor
ARCES	Standard structure of an FPGA
ARCES	PiCoGA reconfigurable device
ARCES	Integration of the XiRisc processor in a SoC
ARCES	Griffy programming environment
ARCES	Network-on-Chip architectures
ARCES	GALS Systems
ARCES	Multiprocessor SoCs
ARCES	Next generation digital computing architectures
TUD	Profiling of applications in terms of real time constraints, memory and power consumption (identification of kernels that are candidates for hardware acceleration)
TUD	Modification of kernels by transformation
TUD	Compilation and Test of generated code
TUD	VHDL for students
TUD	SOC and NOC design
TUD	Hardware Design Lab
TUD	Modern Computer Architectures
UBO	Constraints of reconfigurable platforms: Local memory capacity, possible concurrency in a data path, general streaming of application data
UBO	Usage of tools helping in partitioning, building the communication structure and the stream oriented control
UBO	Adaptation of a computation graph (CDFG) to a specific platform with the numeric aspects (loop unrolling, software pipelining, resource allocation, memory use)
UK	Basic HDL course
UK	HW / SW CoDesign
UK	Bus Systems
UK	SoC and Network-on-Chip paradigms
UK	Target Architectures for HW/SW-Systems
UK	Design flow for Reconfigurable Architectures / State of the Art Design Flow for Dynamic and partial reconfigurable Architecture
UK	Introduction of communication primitives for reconfigurable architectures (FPGA)
UK	One dimensional and two dimensional dynamic and partial reconfiguration

UK	Logic-Synthesis
UK	Technology Mapping / Technology mapping for FPGAs
UK	Dynamic and partial reconfiguration of Spartan-III / Virtex-4 FPGA
UK	SoC-Design Lab

Table 2: Summary of training offers

3.3. Proposed courses

The analysis of the training needs performed the segmentation of the training offer into four main classes:

1. Training for HW-Designers/Engineers
2. Training for SW-Designers/Engineers
3. Training for both HW and SW Designers/Engineers
4. Training for MORPHEUS end users

These main classes were now sub-segmented into training courses with the required content related to section 3.1. In the next table the training requirements were bundled and bounded to specific courses. In the first column a possible course title is noted. The second column references to the topic of section 3.1 in order to satisfy all identified requirement. The last column describes roughly the content of the course. The content of these courses will be detailed in the following sections. This table was updated against D8.1

Training for HW-Designers/Engineers: Course title	Topic of section 3.1	Content of course
Coarse-grained reconfigurable architectures	1, 13	<ul style="list-style-type: none"> • State of the art coarse-grained reconfigurable architectures • Structure of the different processing elements • Coarse-grained reconfigurable architectures used in MORPHEUS (detailed) • Algorithm mapping to parallel architecture • Exploitation of dynamic reconfiguration
HW/SW interaction and design testability	11, 12, 14	<ul style="list-style-type: none"> • Requirement analysis for software demands • Cost estimation for hardware requirements after analysis for software demands • Hardware requirements and parameterization in relation to driver software • Modular and testable design methods • Design of tests suites in correlation to software requirements
Training for SW-Designers/Engineers: Course title	Topic of table 1	Content of course

Parallel computing on reconfigurable architectures	2, 5, 6, 23	<ul style="list-style-type: none"> • Basic concept of FPGA architecture • Mode of use: Software and Hardware partitioning • Abstraction levels for reconfigurable system in MORPHEUS: Top down and bottom up approach • Profiling and native programming • Mapping of algorithms to reconfigurable architecture
HW/SW-Communication in heterogeneous reconfigurable SoC	7, 8, 9	<ul style="list-style-type: none"> • Data transfer in reconfigurable MORPHEUS SoC • Specification of external IP • Driver design for data transfer in reconfigurable architecture • Specification of hardware in relation to software-drivers • Multi-tasking and synchronisation of HW/SW partitioned systems
Training for both HW and SW Designers/Engineers: Course Title	Topic of table 1	Content of course
Hardware-Software Co-design for reconfigurable SoCs	4, 15, 20, 26	<ul style="list-style-type: none"> • Methods and algorithms for HW/SW partitioning • Parallelisation of tasks • Basic knowledge of HDL coding • Bus-standards and Network-on-Chip • High level modelling of complex SoC architectures
Methodology of reconfiguration in MORPHEUS architecture – trade-off and benefits	3,18,19,21,24,29	<ul style="list-style-type: none"> • Training with MORPHEUS architectures • Power, area, performance trade off in heterogeneous reconfigurable SoCs • Cost awareness by exploiting reconfiguration (in terms of power, performance and area) • Concepts and methodology of reconfiguration
Simulation and analysis of heterogeneous SoC	16, 17, 22, 28	<ul style="list-style-type: none"> • Memory usage in heterogeneous SoC architecture • Example: DMA setup and buffer handling • Platform synthesis and analysis of MORPHEUS architecture
Experiences and lessons learned while implementing demonstrator applications	10, 27, 30	<ul style="list-style-type: none"> • Efficient use of the MORPHEUS Toolchain • Telecommunication standards

Training for MORPHEUS end users: Course title	Topic of table 1	Content of course
User guide to the MORPHEUS reconfigurable SoC platform	25	<ul style="list-style-type: none"> • Introduction to application scenarios • Introduction of toolchain for MORPHEUS • Description of flexibility through reconfiguration • Reconfiguration paradigm • Exemplarily: Integration of one example application on MORPHEUS platform (from WP5)

Table 3: MORPHEUS course and training planning

4. Courses for HW-Designers/Engineers

4.1. Coarse-grained reconfigurable architectures

4.1.1 (ARCES)

A brief introduction of the PiCoGA reconfigurable engine is presented. The PiCoGA architecture is shown together with the data flow oriented computation model. Some examples are presented in order to support these concepts.

- Introduction of PiCoGA (Pipelined Configurable Gate Array)
 - PiCoGA Architecture
 - Computation on PiCoGA
 - Multi-context exploitation
 - Examples

The HDL simulation environment with Modelsim Mentor tools is introduced, in order to enable the attendees to verify their designs that integrate the PiCoGA device.

- Introduction to HDL simulation with Modelsim

The DREAM architecture integrating the PiCoGA device is introduced. The computation pattern is presented, explaining the advantages of a large bandwidth memory access to fully exploit the computational power provided by the PiCoGA. The use of programmable address generators is explained in detail, and an example of a memory access pattern is shown.

- Introduction to the DREAM architecture
 - DREAM Dynamically adaptive DSP
 - DREAM architecture
 - Computation pattern
 - High Throughput Data Transfer
 - Address Generators functionality
 - Detailed API description
 - Example of memory access pattern
 - Circular buffering

4.1.2 (UBO)

Basic training on fundamentals of reconfigurable technology

- Contents: how to provide a structured description of a reconfigurable architecture and to bring up a set of tools for this architecture. Fine grain examples. How to build operators, how to produce sequential machines from behavioural specifications, hard and soft 'macros'.

Conceptual presentation of physical device and basic tools

Reconfigurable architectures are flat or hierarchical replications of tiles, ranging from wide to very wide. Tiles group information processing devices, storage, and communication primitives: look-up tables, logic cells, arithmetic operators, or memories. This physical organization is described and programmed by basic tools that will configure its elements to implement logic or arithmetic functions, state machines, internal memories. The input of basic tools is generally a hierarchical composition of elementary computations known to fit the architecture resources.

An abstract model of the reconfigurable 'physical layer' functionalities is introduced based on the following elements:

- a reconfigurable architecture model acceptable from the point of view of physical tools,
- an application prepared for a particular architecture or a family of architectures,
- mapping functionalities in terms of resource allocation, given a particular architecture and a prepared application.

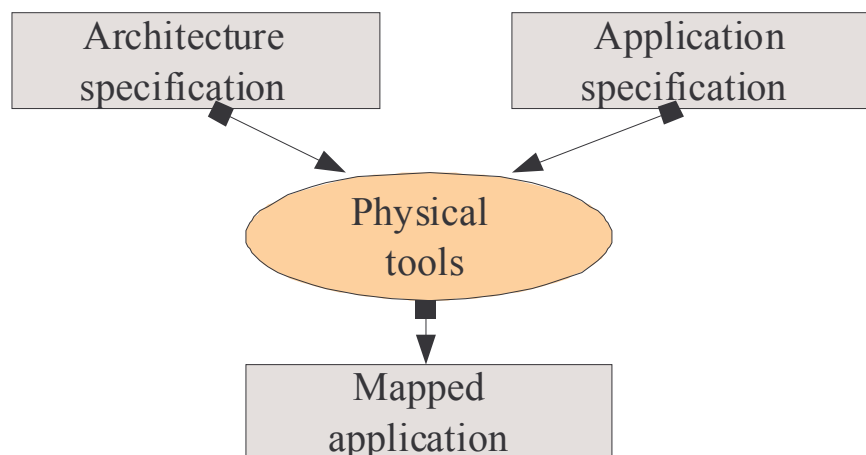


Figure 1: Specification workflow

The elements are illustrated with the cases of simplified FPGA architectures and coarse grain architectures with different application inputs, and mapped applications.

The physical tools are introduced in terms of functionalities: definition of a resource mapper (placing) for logic, arithmetics, memories, definition of both point-to-point and global routers, definition of a floor planner, definition of an editor, regular resources allocator, configuration generator.

Architecture specification

The architecture specification highlights the potential reuse that comes from similarities between several targets, in terms of hardware elements. The specification uses a grammatical definition

combining hierarchically patterns of elements. Both routing devices and compute nodes are characterized to favour modularity and genericity.

Training consists in defining some fine grain and coarse grain architectures from existing examples. Online video support is provided to demonstrate an architecture specification and bringing up an interactive editor on the Madeo environment.

Physical layer application specification

Application specification is a random or regular hierarchical composition of modules (known as soft macros in the industry tools). The modules can refer directly to hardware primitives existing in a target architecture, or logic/arithmetic networks to be remapped on logic cells or arithmetic slices.

Training consists both in analysing a variety of examples, defining regular composition of modules on a given simple syntax, and finally testing the portability of structured applications.

Mapped applications

Mapped applications are the results of applying physical tools to the two previous inputs resulting in a configuration: resource allocation and element characterization implementing the application.

Principles of measures on the mapping are presented. Low level (mapped application) reusability is discussed through hard macro.

Physical fine grain tools and algorithms

Basic algorithms are introduced for placement (deterministic, annealing based), routing (point-to-point, global, two phased), floorplanning (transitive closure graph based), visiting with policy for hierarchical application graph, module building and archiving.

Training on Versatile Place and Route (VPR) and Madeo platforms.

Logic and arithmetic synthesis layer

The interest of this layer is to prepare logic network to produce complex circuit elements for the physical layer, mostly for fine grain. This is based on high level strictly behavioural programs that are represented as networks of look-up tables. Principles of the most critical optimisations are described. Other techniques are shown: "resynthesis" rather than synthesis, nesting in CDFG from the MORPHEUS spatial design, simulation and test, use of SIS, FSM implementations.

Demonstration on a critical application for error correction.

Benefits

- Developing both soft macros and hard macros, storing macros
- Developing a toolset targeting a given architecture
- Adding, customizing and replacing algorithms
- Benchmarking / Hardware prospection

4.2. HW/SW interaction and design testability

4.2.1 (TUD)

- HW/SW co-design
 - Objectives
 - Pareto curves

- Design Space Exploration
 - Simulation
 - Ahmdahl's law
- Profiling and Performance Estimation
 - Dynamic-static profiling
 - Statistical profiling
- Task level partitioning and transformation
 - Graph collapsing
 - Application specific instruction
- Program code instrumentation
 - Pragma annotation
- Compiler and OS scheduling
 - Intra and interprocedural compiler scheduling
 - RTOS scheduling
- Reconfigurable specific compiler optimisations
 - Loop parallelisation
 - Loop optimisations
- Mapping on HW platforms
- Benchmarking
 - Choosing benchmarks
 - Using benchmarks
- Design Verification, Validation, Testing
 - Are we building the system right?
 - Are we building the right system?
 - Testing patterns

5. Courses for SW-Designers/Engineers

5.1. Parallel computing on reconfigurable architectures

5.1.1 (ARCES)

A general introduction to parallel oriented design is presented. Different types and grains of parallelism are shown

- Parallelism analysis and Design guidelines
 - Algorithm exploration
 - Parallelism analysis

The basic programming environment of the PiCoGA reconfigurable engine is presented. A novel programming language, the Griffy-C, is taught giving a detailed description of its syntax. The compiling environment is described too.

- How to program PiCoGA with Griffy-C

- PiCoGA programming approach
- PiCoGA configuration flow
- PiCoGA Operation Description (Basic)
- Griffy-C syntax
- Single assignment form & parallel management
- How to compile Griffy-C
- Bit-stream generation and format

A functional simulation environment based on the GDB debugger is shown together with an example of application. This environment is used in the development phase in order to validate only the mapped application at functional level.

- Functional validation of PiCoGA operations
 - Introduction of PiCoGA simulation environment
 - Debugging of PiCoGA operations
- Example of application (ie. Motion estimation)

The advanced programming environment of the PiCoGA reconfigurable engine is presented. The use of Built-in Hard-Macros and user defined Soft-Macros is introduced and advanced techniques for pipeline balancing are discussed. Internal state management for the support of static variables is also explained.

- Advanced Griffy-C programming
 - Direct LUT utilization
 - Built-in Hard-Macros (ie. Galois field multiplication)
 - Pipeline balancing
 - Internal state managements
 - Delay line
 - Shift register
 - Circular buffer
 - User defined Soft-Macro

An approach to the programming methodologies capable of exploiting the PiCoGA features is presented. In particular it is shown how standard sw techniques such as software pipelining can be adopted in writing Griffy-C in order to expose exploitable parallelism. Furthermore it is explained how to handle large DFGs that don't fit the PiCoGA resources by suitably splitting the original DFG or identifying common sub-graphs.

- Mapping methodologies for DFG implementation on PiCoGa
 - Software pipelining on DAG
 - How to handle large DFGs
 - Splitting
 - Common Sub-graph identification
 - Folding common Sub-graph
 - Internal state utilization

The DREAM Instruction set extension mechanism is explained. All functionalities are implemented in standard C-code, while low-level control over PiCoGA is performed by memory-mapped control/data registers. Details of configuration (I/O matrix and PiCoGA) and execute instructions is given. The compiling environment is described as well.

- DREAM Run-Time library
 - Configuration
 - Data feed configuration
 - Computation trigger
- How to compile for the DREAM architecture

The Application Program Interface (API) for the integration of the PiCoGA device in a simulation environment is shown.

- Application Program Interface
- DREAM Simulation strategy
 - Functional simulation vs. cycle-accurate
 - Cycle-accurate simulation
 - Stall analysis
 - Performance evaluation

5.1.2 (UK)

- Basic concept of FPGA architecture
 - FPGA based Embedded Systems and the required tools
- Mode of use: Software and Hardware partitioning
 - Development of a microprocessor based system on FPGA
- Abstraction levels for reconfigurable system in MORPHEUS: Top down and bottom up approach
 - Adding user defined IP-Cores to an existing embedded system on FPGA
- Profiling and native programming
- Mapping of algorithms to reconfigurable architecture
 - Heuristics
 - Partitioning
 - Decomposition
 - Covering
 - Dynamic programming

5.1.3 (UBO)

Tools developer workbench

- Contents: recalls on system level aspects (Course 5.2.2). CDFG presentation, structured control flow, hierarchy aspects, mapping, low level processes, memory allocation and swapping. The course has theoretical contents, as well as practical ones, allowing handling different tools from the MORPHEUS project showing the migration from high level behavioural specification down to hardware execution. The form of UBO contribution will include conventional material such as slides and written documents, and tools written in an object oriented development platform in order to allow describing visually the interactions in the architecture (communications, buffer transitions, circuit behaviour performance analysis).

Overview

This course covers the compilation chain from the generation of communications and accelerated circuit specification, down to the execution circuit mapped for a particular reconfigurable unit. Therefore, it deals with portability and heterogeneity on the following aspects:

- variety of front-end languages and methodologies allowing to address the platform
- different system semantics for these front-ends
- different grain and flexibility at the back-end
- memories and communication issues
- technical problem of the different development languages used for the tools
- library portability

MORPHEUS carries a set of tools of different levels, with fixed solutions such as SPEAR and CASCADE for the front-end, addressed from a C program compiled by Molen, or the reconfigurable unit specific tools from PACT, STMicroelectronics, or M2000.

Here the purpose is to explain the architecture of WP2 Spatial Design workbench, and to provide information allowing extending this workbench for further languages and reconfigurable units.

This overview is an introduction that will be provided as an online resource embedding videos that show particular aspects of the workbench. The course can be followed by engineers, scientists, and technical managers willing to use the workbench. This part of the course will also serve as an introduction to the full course.

Software flow: partitioning

Starting from an external language, it is necessary to produce a behavioural specification of the computation. This behaviour can be divided into two pieces of software likely to be generated by compilers or methodologies:

- a program producing a communication definition that will be executed by a communication engine (DMA, reconfigurable unit wrapper, main CPU)
- a program representing the accelerated computations working on local memories, to be mapped on the reconfigurable units.

This part of the software flow is explained in UBO course 2; it will be summarized to fix the context in which the accelerator is working. A small technical training is provided for the API handling communication queues writing and reading.

Elements of this course 2 will be reused to support the explanations.

Software flow: addressing the CDFG high level

Coarse grain architectures are assembly of operators, busses, control devices or logic blocks for FSM, and memories. Fine grain offers much flexibility, but these computation elements exist as libraries. The generic assembly language inside the tools allows presenting the computation as a set of processes that interacts using memories or communication primitives. Inside the project, this system structure is based on SPEAR processes communicating through memory accesses according to a static schedule (connectors), the case of external communications being supported by handshakes with the communication engine.

The course shows a typical CDFG-HL with a (1) process structure and synchronisation primitives depending on the system paradigm, (2) 'sequential' computation, possibly pipelined, grouping operators, array accesses, and high level control structure, (3) type information on dependency edges. An API is demonstrated that allows producing, exchanging and visiting a CDFG-HL data structure.

Contents of the training are general explanations, and practical exercises centred on the use of the API.

Workbench internals: translation to the CDFG low level

This part of the course deals with several different sets of information:

- CDFG-HL for the application behaviour
- Repository of classified components corresponding to hardware primitives or libraries, representing high level elementary computations such as simple or complex operators associated to programming symbols and types
- Type systems
- Architecture definition, as explained in course 1
- Translated CDFG, called CDFG-LL that also embeds memory mapping and accesses.

This part of the course shows how to extract information concerning the target architecture resources, how to develop CDFG-HL nodes into component subnetworks according to the types and the repository, and how to produce a CDFG-LL from these developments.

Principles and practical example for mapping a language subset.

Advanced issues

Specific control for a set of processes. Returning information to the higher level tools. Debugging support. Global control.

5.2. HW/SW-Communication in heterogeneous reconfigurable SoC

5.2.1 (UK)

The hardware software interface is one of the most important parts in system design when talking about global understanding of modern SoC platforms in education. In a heterogeneous reconfigurable system, which is expected to be the one of the main branches of upcoming SoC development, hardware software co-design and therefore its interfacing becomes highly relevant. So, engineers and students need to have knowledge about HW/SW modelling techniques. To satisfy this demand, the University of Karlsruhe was setting up a new course that exactly covers this topic.

The course is called "System on Chip Laboratory", where a SoC is going to be designed that contains a microcontroller and a reconfigurable coprocessor module. The complete SoC is implemented on a FPGA and finally a multimedia application will demonstrate that software as well as hardware is working. The main focus is lying on the hardware software interaction and synchronisation, such as how commands written in software deliver subtasks such as highly computational algorithms to dedicated hardware modules. This laboratory gives the students the ability to get a detailed insight into each single task of a system developer as well as to gain a complete overall understanding of nowadays SoC architectures and interfaces. In the end, the students handled each step of the development process of a state-of-the-art system design. Figure 2 shows the system as it will be integrated on an FPGA.

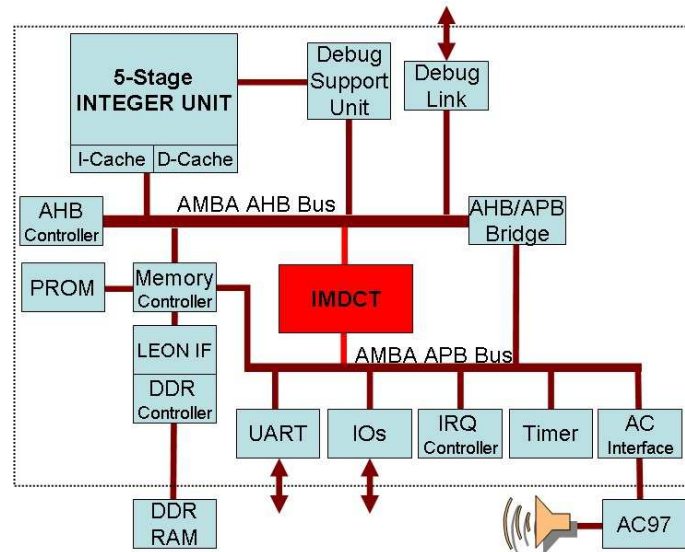


Figure 2: System of the SoC laboratory

An overview about the students tasks are given in the following:

Starting from a software reference design, where the complete multimedia application is executed on the microcontroller, the traditional von Neumann idea can be illustrated. The main focus is then lying on the new hardware software partitioning and its interfacing. Therefore the most computation intensive algorithm, which is the IMDCT, will be extracted from the software's c code and implemented in hardware (VHDL). This newly developed hardware module has further to be integrated, hence the interface of the IMDCT module has to be developed and the busses of the SoC have to be adapted and connected.

After hardware simulation, that is always essential for verification purposes, an important task of the students is to set up the framework for communication of the hardware module with the software. Software adaptation and driver development are the keywords that are made accessible to the students. Surely, debugging and verification will play a major role during the whole system design in terms of time consumption. Finally, to top the SoC development process off, FPGA synthesis and a system analysis with respect to performance constraints will be accomplished.

As it can be seen, this laboratory tries to cover all aspects that are relevant in a SoC design flow. The focus and intention of the laboratory was thereby not lying in the particular understanding of each line of c-code and VHDL, but to gain the point of view of a system designer.

5.2.2 (UBO)

System level behaviour, a software oriented design flow

- Contents: relations between the software control core and accelerated processes, memory access and data structure abstraction, address transformation mechanisms, distributed and centralized control. Models for performance analysis.

Motivation

The general behaviour of applications running on MORPHEUS is very different from software applications, where 'everything is feasible', and SoC design oriented to IP reuse. The introduction of the course provides a detailed analysis of a memory oriented, configured application working in a SoC explaining:

- how the execution is achieved in terms of data movements memory to memory, and inside the accelerator, in terms of processing and temporary storage

- what are the critical devices involved in the execution: processors, DMAs, configured operations
- typical schedules of operations, pipeline and streaming aspects
- possible specification and methods, available tools and interchange principles

The motivation will be available as a one hour overview organized around a simulator showing how the execution is achieved, and what is a typical software design flow. This overview will be available as a video file with some interactive tools to demonstrate properties and behaviours at system level: dynamic reconfiguration, typical HRE choices for applications, role of programming language and methods, role of translation tools. Motivation course could be distributed widely by the internet with early versions available under agreement of the project partnership. It is also an introduction to the rest of this course.

Data movements and programming

High level languages support data definitions in terms of simple types and structures such as arrays, records, various kinds of streams, etc ... The physical data organization in a processor memory is decided by the language compiler, but programmers or tool writers have most of the time resources allowing controlling data size and addresses, or sparse and contiguous characteristics of data structure.

Feeding an accelerator by program is a problem because the useful data can be sparse in memory while special arrangements are required in the accelerator physical memory. The key point to solve this issue is a hardware address computation and transfer mechanism that can operate concurrently with the processor, in coherence with the accelerator. This mechanism can be a DMA.

Data movements are explained as a memory to memory loop involving sparse or contiguous data transfers from main memory to local memory, local memory to local memory, and local memory back to main memory.

- Static definition of the transfers is explained using a library allowing to define elementary transfers and queued transfers grouped by execution step coherent with the execution on the accelerator (packet oriented communications).
- A programming training targets a simple simulator that closes the read-write loop on local memory simulation with observable data moves in main memory, and without data processing. A practical example with sources is provided with a tool that eases data access descriptions.

Document, tools and video showing how to handle the tools

Model for an accelerated execution

Once global data are remapped to local memories, it is time to examine how they are processed and what are the possible issues in process structuration.

- The previous training continues with a motivation for overlapping communications and executions.
- An execution process is inserted in the main memory to main memory loop that represents the accelerator data transformations, and an extended version of the previous simulator executes this process concurrently with memory transfers. This process is programmed in a high level language, and compiled for the simulator. The engineer now sees a pipeline with one stage on a dedicated accelerator, the other stages grouping buffer transfers.
- This model is extended to the partitioning of the acceleration as two cascaded processes. Discussion on the local use of memories, internal and system synchronizations.

Document, tools and videos showing how to proceed

Mapping the accelerated function

Using spatial design tools to map accelerated processes to circuits or proprietary IP tools. Getting performance numbers.

Document, tools and videos showing how to proceed. Probably only a partial development with code generation down to a mapped CDFG to get estimations on circuit characteristics.

Performance analysis

How to get a global figure of the execution. This is done on the top of a system model grouping analysis of the transfers, capability of the network on chip, accelerator throughput and latency. A practical case is given and discussed.

More background and conclusion

Explanations on the practical MORPHEUS framework: nesting of the accelerated function in a program, relation with the operating system, practical HRE targets, dynamic reconfiguration, multiprocessing, token streaming and flow regulation problems, etc...

6. Courses for both HW and SW Designers/Engineers

6.1. Hardware-Software Co-design for reconfigurable SoCs (UK)

The University of Karlsruhe is going to provide a course in Hardware-Software Co-design for reconfigurable SoCs. It will cover 7 topics starting with “estimation of design quality” which, after an introductory lesson on abstraction levels and system synthesis, covers the fields of graph models for data / control flow and estimation techniques. The students will learn the parameters and metrics of estimation techniques. They will get to know different techniques for the estimation of hardware performance, software performance and communication performance.

An important base for partitioning is graph theory. The students will learn about graph-classes. They will learn several graph problems, like Dijkstra’s algorithm for shortest path and Kruskal’s algorithm and Prim’s algorithm for the Minimum Spanning Tree problem. Another important graph problem is the maximum clique problem.

In the third chapter the students will hear about two forms of parallelism, temporal parallelism and spatial parallelism.

The fourth topic is “Methods and algorithms for HW/SW partitioning”. After a generic definition of the partitioning problem including cost functions and abstraction levels, it will introduce different partitioning approaches. The students will learn the exact partitioning methods enumeration and integer linear programming (ILP). As these methods are only feasible with small problems, the heuristical partitioning methods will be introduced. Heuristical methods can be grouped in constructive methods and in iterative methods. The concept of constructive partitioning methods will be explained on the example of hierarchical clustering. The iterative partitioning methods will be explained on the examples of the Kernighan Lin algorithm, the Fiduccia Mattheyses algorithm, Tabu-Search and Simulated Annealing. The students will also learn about the use of genetic algorithms for partitioning problems.

The students will learn basic knowledge of HDL coding resulting in the knowledge to do hardware modelling with VHDL and verify these models with hardware simulation with VHDL.

This leads to the high level modelling of complex SoC architectures and the introduction to the System on Chip paradigm.

The connectivity between different blocks on a System on Chip can be provided in two different ways. The students will learn the use of bus systems on the example of the AMBA AHB. They will also learn about Networks on Chip.

6.2. Methodology of reconfiguration in MORPHEUS architecture – trade-off and benefits (CEA)

Configuration controllers are common in reconfigurable SoCs as an interface to load bitstreams in the reconfigurable devices. We first review the work performed on FPGAs for the dynamic management of tasks on homogeneous reconfigurable slots. Then a predictive configuration manager for heterogeneous multi reconfigurable cores is presented. In the trend of giving to the architectures higher levels of autonomy to solve the complexity management for the future generations of SoCs, we show the benefits of advanced reconfiguration services. In the case study of the MORPHEUS project, the sizing and service selection is completed to illustrate the flexibility/surface trade-off. We finally demonstrate a complex application scenario to show how the prediction performs and integrates with the OS.

6.3. Simulation and analysis of heterogeneous SoC (ARCES)

Some general aspects about the usage of the simulator environment are described. Attention is given in the usage of the memory model in order to analyze bandwidth available with different configuration of the hierarchy memory. Some examples describe in detail how to program the DMA engine and how to speed-up the executed code with the support of the TCM memories in the ARM.

- The memory model
- Local buffer configuration
- Introduction to the DMA C driver
- How to program the DMA in C
 - Single Block transfer
 - Multi Block transfer
 - Interrupt management

6.4. Experiences and lessons learned while implementing demonstrator applications

This course aides future users of the MORPHEUS platform by condensing the experiences and lessons learned by the partners of WP5 and IP providers during the implementation of their applications. It is split in two coarse parts corresponding to the general fields of the intended applications.

6.4.1 Telecommunication standards

6.4.1.1 Optical networking domain (LUCENT)

For telecommunication standards in the optical networking domain (network routing systems etc.), there is a high number of courses and training programs available on the market, e.g.

- The basic communication protocols like Ethernet, TCP/IP, SDH/SONET etc. are subject to lectures on many universities in several technical programs
- Equipment manufacturers like Lucent offer trainings for their customers, which cover the general telecommunication standards and protocols as well as the product specific knowledge. Some of these vendors (e.g. Cisco) offer a broad range of trainings, courses, tutorials incl. certifications not only for customers, but also for other interested persons.
- Measurement equipment vendors offer similar trainings
- Almost each education & training provider in the technical domain offers courses, tutorials etc. about telecommunication standards and protocols

- There is a number of online resources to get familiar with the basic telecommunication standards, from education & training providers as well as from equipment vendors (e.g. <http://apex.vtc.com/cisco-ccna.php>)
- Free online resources are available as well, e.g. via Wikipedia (<http://www.wikipedia.org/>), but the main portion of courses and training programs are commercial

As most of these offers are either restricted to a specific group of persons (e.g. students, customers) and/or are associated with costs, Lucent developed an own introductory course for the basics of Ethernet. The purpose of this course is to provide a sufficient level of Ethernet knowledge to be able to understand the demonstration application of Lucent within MORPHEUS. Furthermore internal course material is available for the real target applications (which are too demanding for a demonstrator). This material will be condensed and compiled in an additional section of the demonstrator course to underline the need for the reconfiguration technology as demonstrated in MORPHEUS. The course material will be completed, once the demonstrator in phase 1 has been finalized and its results are presented (in the time frame of M5.3 evaluation of phase 1, M21).

6.4.1.2 Wireless Communication (ICOM)

- Background
 - WiMAX/Mobile WiMAX overview
 - OFDMA PHY layer description
 - MIMO algorithms review
- Reconfigurability scenario
- Implementation
 - Executable specification
 - Partitioning based on task granularity
 - Communication between kernels
 - Verification
- Discussion
 - Performance evaluation & bottlenecks
 - Tool flow issues
 - Comparison to other implementation platforms

6.4.2 Video data processing

6.4.2.1 HD video (PACT)

- HD Video Standards (SMPTE)
- Video codecs
- H.264
 - Algorithms
 - System requirements
 - Partitioning: Control-flow and data flow
 - Control flow – example with XPP
 - Data-flow – example with XPP
 - Dynamic reconfiguration

- Bandwidth and Memory requirements
- Audio
 - Audio standards and interfaces
 - Audio processing resp. reconfigurability
 - From reference code to implementation
 - Audio interfacing

7. Courses for MORPHEUS end users

These courses focus on the MORPHEUS specific aspects of reconfigurable computing. They will enable application designer an efficient use of the MORPHEUS tools and architecture.

7.1. User guide to the MORPHEUS reconfigurable SoC platform

7.1.1 (TRT)

In this course devoted to future end users of the complete HW/SW platform, it is proposed to start with the description of the applications scenarios, then to present the benefits of dynamic reconfiguration from an user point of view, to explain the different components of the SoC architecture as well as the tool chain, to conclude by an example of an implementation of a selected test case on the platform by using the tool chain.

TRT proposes here a list of slides to be prepared for the implementation of the course. It is composed of four sessions, with the objective of providing a sufficient knowledge of the MORPHEUS platform to start using it.

TRT proposes accordingly this contend, with a list of corresponding slides:

Session 1: Introduction to application scenarios

For the four application test cases, the purpose is to appreciate the benefits resulting of the MORPHEUS developments. This starts with a description of these scenarios continued by the analysis of different key features offered when using the dynamic reconfiguration capabilities and an insight of the potential impact. This result in the following slides:

- application domains addressed: the range of potential applications:
- broadband wireless telecom systems description
- exploitation of reconfiguration in a telecom system
- optical networking system description
- reconfiguration under system availability constraints
- intelligent cameras system description
- dynamic reconfiguration for camera performance density
- HD professional video system description
- dynamic reconfiguration for video system performance density
- resulting sizing considerations

Session 2: Rationale for flexibility through reconfiguration

This session will cover the different benefits resulting from using dynamic reconfiguration from a system perspective: new capabilities, new performances. This includes the following slides:

- system level dynamic reconfiguration

- example of multi mode utilisation
- algorithm level dynamic reconfiguration
- example of data dependent run-time configuration
- function level dynamic reconfiguration
- example of reconfigured functions along a processing chain
- reconfiguration motivations on performance density

Session 3: Introduction to MORPHEUS SOC architecture

This session is intended to detail the different parts of the MORPHEUS architecture to understand its mechanisms so as to be able to do the best exploitation of the platform. It consists of:

- global view of MORPHEUS SOC architecture
- dataflow efficiency within the architecture
- control mechanisms
- dynamic reconfiguration mechanisms
- configuration manager
- busses and NOC
- DMA and Direct NoC Access features
- memory hierarchy
- HRE standard interface
- XPP IP within HRE definition
- PiCoGA IP within heterogeneous reconfigurable engine
- M2000 IP within heterogeneous reconfigurable engine

Session 4: Introduction to MORPHEUS toolchain

This session is intended to introduce the different parts of the MORPHEUS Integrated tool chain so as to understand the design flow, the functionalities and the key details and being able to do the best use of this design and programming environment. It consists of:

- global view of MORPHEUS toolset approach
- the Molen paradigm
- the Molen intermediate instructions
- reconfiguration scheduling at compile-time
- dynamic reconfiguration OS with static allocation
- dynamic reconfiguration OS with dynamic allocation
- formal specification
- optimising datastream throughput efficiency
- graphical datastream function capture
- multidimensional arrays
- loop kernels and elementary transformations
- loop kernel fusions and pipeline

- coarse synchronisation
- DMA parameters generation
- processing chain within heterogeneous reconfigurable engine
- inter IP data reorganisation and bufferisation
- fine synchronisation and buffer optimisation
- retargetable synthesis

An existing Web VIDEO of SPEAR tool could be used to illustrate this part of the course.

Session 5: Application implementation example on MORPHEUS platform

The goal of this part is to explain on a concrete example (from the real application test cases) how to use the platform with an illustration of the evaluated (measured or simulated) benefits. It is composed of:

- application description reminder (possibly Camera or Professional video)
- application algorithm details
- application global description through Molen C description
- Molen compilation results (code abstracts)
- Molen graphs for OS and configuration manager
- OS scheduling and allocation examples
- accelerated function C code example (within Molen program)
- accelerated function graphical capture example
- accelerated function mapping example on SOC level
- accelerated function fine synchronisation optimisation example
- accelerated function implementation result example

7.1.2 (LUCENT)

Introduction to application scenarios

Lucent will develop an introductory course to explain its application scenario for the MORPHEUS demonstrator. The course will consist of:

- Telecommunication standards course material as described in section 6.4.1.1
- Motivation for reconfiguration in the Lucent application domain
- Description of the reconfiguration approach in a telecommunication network (i.e. how to reconfigure a large number of SoCs in a real telecommunication network with minimal impact on the network services)
- Introduction to the MORPHEUS SoC (might be re-used from other courses)
- Technical overview about the mapping of the Lucent application to the MORPHEUS SoC
- Description of the application specific reconfiguration mechanism for a single SoC
- Evaluation of the demonstration results, outlook to future exploitation

The first version of the course will be available after phase 1 of the demonstration and will be updated after phase 2 with the additional results gained from the demonstration with the real MORPHEUS SoC.

8. Concluding section

8.1. Conclusion

This document provided an overview about the content of the courses which have been proposed in deliverable 8.1 "Industrial needs and course specification". It shows that WP8 is progressing according to plan towards education offers fundamental to the necessary cultural changes for the widespread adoption of reconfigurable computing. Next step is the realisation and execution of the training courses.

8.2. Acronyms

<u>CDFG</u> :	Control Data Flow Graph
<u>DFG</u> :	Directed Flow Graph
<u>DMA</u> :	Direct Memory Access
<u>FPGA</u> :	Field Programmable Gate Array
<u>HDL</u> :	Hardware Description Language
<u>HRE</u> :	Heterogeneous Reconfigurable Engine
<u>NoC</u> :	Network On Chip
<u>PiCoGA</u> :	Pipelined Configurable Gate Array
<u>RTOS</u> :	Real-Time Operating System
<u>SoC</u> :	System On Chip
<u>VHDL</u> :	Very High Speed Integrated Circuit Hardware Description Language
XPP:	eXtreme Processing Platform