



MORPHEUS

D8.4 Report on Course execution

CONTRACT NO MORPHEUS IST 027342
TYPE OF DOCUMENT Deliverable
DATE 11/05/2009
ABSTRACT This deliverable gives an overview about the courses that have been setup during the MORPHEUS Project. It shows how MORPHEUS affected the curricula for hardware and software engineers targeting heterogeneous and reconfigurable SoCs.
AUTHOR, COMPANY Michael Hübner (UK)
Matthias Kühnle (UK), Henning Sahlbach (TUBS);
Bernard Pottier (UBO); Davide Rossi (ARCES)
WORKPACKAGE WP8
CONFIDENTIALITY LEVEL PU
FILING CODE MORPHEUS-D8 4-UK-R1 5.doc

DOCUMENT HISTORY

<u>Release</u>	<u>Date</u>	<u>Reason of change</u>	<u>Status</u>	<u>Distribution</u>
R1.1	05/03/2009	Creation	On-going	website
R1.2	20/03/2009	Update		email
R1.3	27/04/2009	Review accounted	On going	email
R1.4	29/04/2009	Proof reading	Done	email
R1.5	11/05/2009	Final check	Done	EC-reviewers



Table of Contents

1. INTRODUCTION.....	1
2. EXECUTIVE SUMMARY.....	1
1. DESCRIPTION OF “MORPHEUS AFFECTED” COURSES.....	1
1.1. CUT.....	1
1.1.1 Lecture “Circuit Design”.....	1
1.1.2 Lectures “Systems Design” and “Rapid Prototyping”.....	2
1.1.3 Lecture “EDA-Tools”.....	2
1.2. TUD.....	2
1.3. TUBS.....	2
1.3.1 “Computer Architectures II”.....	2
1.4. UBO.....	3
1.4.1 UBO courses.....	3
1.4.2 Morpheus impact on UBO courses.....	4
1.5. ARCES.....	5
1.5.1 Electronic System Design (R. Guerrieri).....	5
1.5.2 Digital Architectures for Signal Processing (R. Guerrieri).....	5
1.5.3 Impact of Morpheus project on the courses.....	5
1.5.4 Seminars, trainings and labs.....	5
1.6. UK.....	7
1.6.1 “Hardware Software Co-design”.....	7
1.6.2 „SoC laboratory“.....	8
2. AUTUMN SCHOOL FEATURING MORPHEUS RESEARCH RESULTS.....	9
2.1. AMWAS 2008 - PROGRAMME.....	9
2.1.1 Autumn School.....	9
2.1.2 Workshop.....	10
2.2. SELECTED ABSTRACTS OF AMWAS 08.....	12
2.2.1 Spatial design backend: CDFG mapping on eFPGA and DREAM IPs.....	12
2.2.2 Heterogeneous Multi-Core Architecture: The Added value of Physical Design.....	12
2.2.3 Mapping of a Film Grain Removal Algorithm to a Heterogeneous Reconfigurable Architecture.....	14
3. MAPPING OF PROPOSED COURSES FROM DELIVERABLE 8.2 TO ACTUALLY DEVELOPED COURSES.....	14
4. CONCLUSION.....	16

1. Introduction

System design with traditional von-Neumann architectures is facing increasing problems in terms of design productivity, power consumption and more but students are currently not made aware of the potential of reconfigurable computing as a solution for the current and future problems to sustain Moore's Law with von-Neumann architectures and thus a "disruptive education reform" (R. Hartenstein) is required. 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 shall provide the necessary skills for a successful dissemination of reconfigurable computing in general and MORPHEUS in special. These courses were separated in 4 main areas:

- Training for HW-Designers/Engineers
- Training for SW-Designers/Engineers
- Training for both HW and SW Designers/Engineers
- Training for MORPHEUS end users

Content and training material for newly developed and adapted courses as well as a correlation of the achieved courses according to the 9 requested ones as outlined in deliverable 8.2 are described in this deliverable.

2. Executive summary

This deliverable describes how research results coming from the MORPHEUS work packages have been included in students curricula. As a summarizing example, some abstracts of a part of the courses of the AMWAS 08 summer school are given.

The aim of the deliverable is to show the impact of the research done within the MORPHEUS project on training issues. The chapters illustrate how these have affected students' curricula. It is also shown, which lectures have been updated with regard to motivate reconfigurable system with MORPHEUS as one or even the main case study. It is shown that a number of courses have changed with respect to bringing the MORPHEUS research in the curricula. The identified necessary courses/material which has been defined within WP8 (in D8.2) are compared with adopted courses towards these needs. It is shown that eight out of nine identified topics are covered in students' curricula up to now, which is a respectable result.

1. Description of "MORPHEUS affected" courses

1.1. CUT

The MORPHEUS topic is addressed in several lectures of the Chair Circuit and Systems Design.

1.1.1 Lecture "Circuit Design"

This lecture is attended by students of several study paths like electrical engineering, computer science and industrial engineering at the end of their Bachelor studies. It deals with the basics of digital circuits starting with simple PLD up to complex FPGA and Semi-Custom architectures. The MORPHEUS approach will be shown in the FPGA section of the lecture.

1.1.2 Lectures “Systems Design” and “Rapid Prototyping”

These two lectures deal with the introduction of hardware description languages (VHDL and Verilog). Several practice lessons are included where the Bachelor and Master students of electrical engineering and computer science design their own small components like watches or simple processors. The MORPHEUS reconfiguration approach of phase 1 is planned to be used for online practice where the students can reconfigure “their” FPGA Ethernet-based by web-interface.

1.1.3 Lecture “EDA-Tools”

In the EDA-Tools lecture several design tools are presented for specification and design of complex systems. The Master students deal with commercial and non-commercial tools and approaches like Matlab, SystemC or SystemVerilog. The specification handling is shown using the tool “SpecEdit” from MORPHEUS WP2. SpecEdit is also used to introduce model checking techniques into the auditory in the final year of the Masters study plans.

1.2. TUD

Faculty members of the CE lab at TUD are currently preparing a new course on Reconfigurable Computing Design, scheduled to be taught from academic year 2009/2010 on. This course introduces the students to the field of Reconfigurable Computing. The course will provide a balanced insight of both theoretical trends and practical hands-on experience with reconfigurable technology. In addition to the basic concepts, the students are taught advanced topics in reconfigurable technologies, organizations, architectures, programming paradigms, design tools and runtime systems. The particular topics to be covered are, but not limited to: fine and coarse grain reconfigurable technologies, partial and runtime reconfiguration, context switching, runtime algorithms for reconfigurable resource management, tools and methods for system-level and RTL synthesis, reconfigurable-specific compiler optimizations. Special attention is paid on application-specific acceleration in the context of reconfigurable technology. For example, the following application domains will be considered: image processing, streaming applications, encryption, compression, bioinformatics, supercomputing, etc.

Learning goals of the course:

- The student understands the basics of reconfigurable technology;
- The student is able to design efficiently for reconfigurable technology (in terms of performance and resource utilization);
- The student can identify computationally intensive parts of applications that are suitable for acceleration using reconfigurable computing;
- The student masters state-of-the-art EDA tools for reconfigurable computing;
- The student will be able to effectively use runtime and partial reconfiguration;
- The student will be able to approach the design problems using a top-down methodology (starting from an application, to an architecture, down to a (re)configuration bitstream);

The preparation of this course has been substantially influenced by the experience gathered within the MORPHEUS project and its achievements will be referred in the lecture notes.

1.3. TUBS

1.3.1 “Computer Architectures II”

The lecture “Rechnerstrukturen II” (in English: computer architectures II) is held by Prof. Rolf Ernst and deals with the design and architectures of embedded systems. It is offered for graduate students and approximately 20 students visit this lecture per semester. The lecture is divided into three major parts: specification and description of digital systems, architectures of embedded systems and implementation of embedded architectures.

Relations of the course “computer architectures II” to MORPHEUS

The key target of the lecture is to teach the basic techniques for understanding and designing embedded systems. First, this includes formal description methods for embedded systems such as synchronous data flow graphs (SDF), Petri nets, state charts or finite state machines. Second, various typical architectures such as micro controllers, DSPs or network processors are introduced to give an overview of existing embedded processor types and their characteristics. Usually these architectures are explained using example processors. Finally, design and synthesis techniques for embedded architectures are explained. This part includes compiler technology for embedded systems and scheduling strategies for real-time systems.

Reconfigurable architectures are included in this lecture as one specific type of processing platform. Together with its reference platform (FlexFilm), which is shown as an example for a FPGA-based architecture, the MORPHEUS platform is presented as an advanced reconfigurable architecture with heterogeneous processing engines. Due to its mixed granularity, the MORPHEUS platform is very suitable for teaching, as the different reconfigurable engines can be compared and their advantages and disadvantages can be evaluated, which leads to a better understanding of the various granularity levels for reconfigurable processors.

1.4. UBO

UBO communicates the knowledge about designing a middleware that implements a support for high-level compiler and offers synthesis capabilities to address reconfigurable IPs. This knowledge requires a cross expertise in software and electronic domains such as software development skills, compilation and code generation capabilities as well as synthesis and EDA tools interfacing.

1.4.1 UBO courses

UBO has set up a Master curriculum in Computer Science with a specialty in Software for Embedded Systems. During the first year, the students receive an “in depth” training in systems, concurrency, and advanced architecture, plus a course in software development:

- Distributed algorithms, development in concurrency based on CSP, fundamentals
- Abstract modeling of network activities (routing algorithms, naming, queue management, control flow, leaky bucket, etc.)
- Parallel architectures and practice (data parallelism, threads, systolic design, memory hierarchies)
- Compilation
- Object oriented development, version control, co-developments, tests, virtual machines
- Real-time operating systems
- Advanced algorithms and computation models

The second year is open for students having equivalent skills and is dedicated to software development for embedded systems. This year brings the Lab-STICC skills to the students providing them a deeper background in synthesis, reconfiguration, testing and system activities. The curriculum definition is as follow:

- Reconfigurable Architectures and Tools Development (Ra&TD): Physical modeling at logic level such as for FPGAs or nanotechnologies. Circuit organization, component models. Fundamental operations: placing, routing, editing, performance evaluation, library structures, binding to specifications.
- Synthesis for Reconfigurable Architectures (SRA): Synthesis in different aspects such as logic and architecture synthesis, role of constraints such as resource allocation and processing delays. Multiprocessing in hardware and synchronization implementations.
- Real time systems for SoC. Internal communications, DMAs. Embedded processors.
- Multimedia, stream coding, coding for error correction.

- Test and validation, principles, digital and analog.
- Advanced compilation and high performance computing.

The students follow a 4 to 6 months internship in industry, or research laboratories. They can spend this time in foreign countries under professor supervision.

1.4.2 MORPHEUS impact on UBO courses

The MORPHEUS project is used as a practical case study for software development course. This course covers two separate fields: the first one addresses project management and the second one focuses on engineering techniques for agile development and tools integration.

The figure 1 illustrates the flow of the UBO contribution within the Morpheus spatial design task. On the right side, the CDFG design-time makes use of meta case tools and modeling techniques. On the left side, the CDFG use-time goes through code compilation (C, CSP, ST80) or Cascade/SPEAR front-end to instantiate CDFG.

The bottom part of the figure illustrates CDFG synthesis to address a reconfigurable target. It is the place where advanced concepts from MORPHEUS will be given, possibly in relation with international partners.

Two levels are under consideration: synthesis, and target modeling. Synthesis makes use of several schemes depending on the nature of the control to operate. For non predictable synchronization (e.g. DMA hand shakes), finite state machines are used. For internal computation schedule, structural controllers are considered. Also inter-process synchronization goes through channel communication that allows addressing dedicated compilation schemes.

A module focuses on reconfigurable architecture modeling. This module also includes result from MORPHEUS such as coarse grain and memory management.

Besides, recent published work addressing automatic wrapper generation for testing of post-synthesis CDFG netlist is going to be injected in the SRA course.

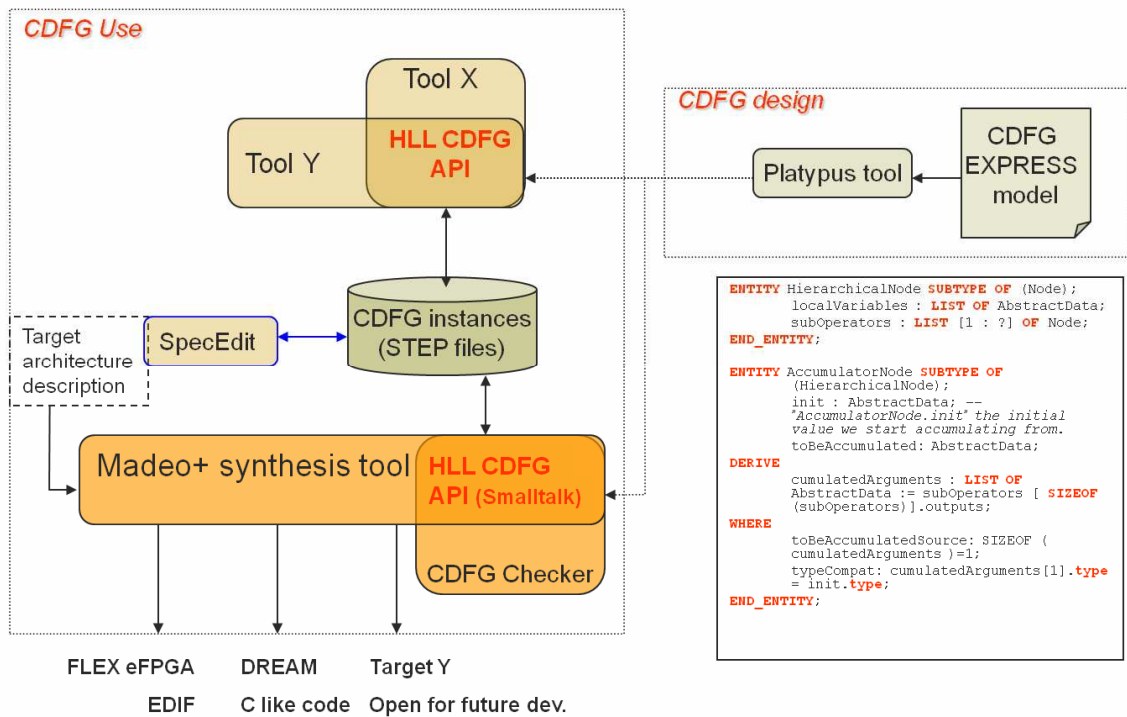


Figure 1: spatial design at UBO

1.5. ARCES

1.5.1 Electronic System Design (R. Guerrieri)

This course covers the most common design issues that have to be considered when a system on board has to be designed. We cover the main integration issues related to glue logic implemented through FPGA and electric design of power, clock and signal lines. In addition, package and materials are discussed in terms of their impact on overall performances. The main points are:

- Logic design based on VHDL
- Hardware design based on FPGA
- Physical implementation of digital systems: heat removal
- On board signal management: derivation of models to account for distributed electrical parameters
- Design of termination of lines
- Design of clock networks
- Design of power distribution networks
- Electrical design of buses and overview of metastability

1.5.2 Digital Architectures for Signal Processing (R. Guerrieri)

This course covers the most common digital architectures for signal processing. Starting from the study of a few algorithms required for consumer applications in video and speech processing, we derive the main specifications for an actual hardware and software implementation. This approach allows us to explain why existing machines are organized in this way and understand the ongoing trends. The main points of this course are:

- Analysis of algorithms for image and video processing with a derivation of the computational costs;
- Architectures of the available processors with examples of ASIC and DSP machines taken from existing products;
- Models of parallel algorithms: pipeline oriented computation;
- Parallel algorithms derived from Data Flow Graph description;
- Mapping of algorithms on parallel hardware;
- Impact of word length on a few simple algorithms;

1.5.3 Impact of Morpheus project on the courses

The Morpheus project is used as a practical case study for both Digital System Design and Digital Architectures for signal processing. Seminars, trainings and labs about spatial computation, run-time configurability has been proposed, together with problematic regarding integration of different flavours of reconfigurable devices in a system, with particular reference to the Morpheus system. Trainings and labs about DREAM architecture, programming model and run-time with mapping of real applications has also been proposed as illustrated in the following sections.

1.5.4 Seminars, trainings and labs

Parallel computing on reconfigurable architectures

A general introduction to parallel oriented design is presented. Different kinds of parallelization approach are discussed starting from Instruction level parallelism, to data level parallelism and finally to thread level parallelism.

The basic programming environment of the PiCoGA reconfigurable engine is presented. Both the syntax and the programming environment of the Griffy-C programming language is described:

A functional simulation environment based on the Eclipse 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.

The DREAM Instruction set extension mechanism is explained. All functionalities are implemented in C-code, while low-level control over PiCoGA is performed by memory-mapped control/data registers. The compiling environment is described as well.

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

System level integration of reconfigurable architectures

An introduction about different flavours of run-time programmable engine is presented including FPGAs, mid grain and coarse grain reconfigurable engines.

A description of possible integration strategies for run-time reconfigurable machines into complex systems together with advantage/disadvantage is presented:

- Instruction level acceleration: reconfigurable engine coupled with GP processor, the processor control all the accelerator data-flow.
- Intensive computational engine (DREAM computational model): data is stored in a tightly coupled multi-bank memory, and massively parallel computation is performed with many iteration of data flow through the accelerator to exploit data locality.
- Streaming computation (M2000, XPP computational model): accelerator is coupled with I/O FIFO with asynchronous communication with the rest of the system.

Differences/Affinities between DMA/Address generators approach to feed accelerators and suitability with shown models are presented:

The DMA approach description is shown including standard DMA functionalities with particular reference to DMA controllers used in the Morpheus system. Advantages and disadvantages of coupling DMAs with FIFO-based accelerator are explained. Example of application accelerated by coupling DMA and FIFO based accelerator are shown with particular reference to M2000 and PACT XPP integration on the Morpheus system.

The Address generator approach is described in details. Address generator functionalities are shown in general with specific reference to the DREAM address generators API. Coupling Address generator model with intensive computational engine and multi bank memories (DREAM architecture).

Network-on-Chip architectures

A general overview of the system level issues are presented starting from data communication infrastructure, moving to configuration level, and ending with synchronization issues in multi-core systems.

- From the communication infrastructure point of view, possible ways to handle communication in multi-core systems are presented:
 - Simple Bus concept description
 - Multilayer Bus concept description
 - NoC concept description
 - Detailed description of Spidergoon STNoC with specific references to the 8-node ST Spidergoon NoC implemented in the Morpheus system.
- Then synchronization issues in multi core systems are presented:
 - General description of synchronization issues in multi-master system

- Exchange register approach to handle synchronization with specific reference to Morpheus system
- Configuration management using run-time programmable machines:
 - General description of issues regarding run-time reconfigurability
 - Description of techniques to hide re-configuration latencies
 - Example of application which requires dynamic reconfiguration with specific reference to DREAM architecture

Training and Lab on DREAM architecture and tool-set

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

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

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.

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.

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 of state-of the art applications on the DREAM architecture has finally been proposed.

1.6. UK

1.6.1 "Hardware Software Co-design"

The lecture about Hardware Software Co-design and reconfigurable systems has been updated and is held by Michael Hübner since autumn 2008. The lecture communicates the knowledge about possible strategies with regard to the design of embedded hardware/software systems. Since autumn 2008, special emphasis is put on the benefit of reconfigurable systems. The MORPHEUS platform is taken as exemplary and motivating vehicle. About 40 Students are visiting this course.

Relations of the course "Hardware Software Co-design" to MORPHEUS

Hardware/Software Co-design is the denomination of the concurrent and influenced design of hardware and software components within one system. In most modern embedded systems (for example mobile phones, automotive and industrial controller devices, game consoles, home cinema systems, network routers) one main problem is to find the right balance between hardware and software components. Enabled by the rapid progress in microelectronics, embedded systems are becoming increasingly more complex with manifold application specific criteria. The deployment of computer aided design tools is not only necessary for handling the increasing complexity, but also for reducing the design costs and time-to-market. The lecture Hardware/Software Co-design discusses the needed criteria, methods and possible hardware/software target architectures on the topics given below. From the lecture hold in autumn 2008, MORPHEUS started playing a major role, since it not only combines pure software and hardware structures, but also reconfigurable IPs, which can be looked at as something in between these two fields. It is explained, how

reconfigurable Engines can be programmed similar to a processor, however architecture dependent modelling is mostly beneficial for efficient application development. The accompanying exercises intend to consolidate the knowledge from the lectures. Selected topics will be repeated and, by working on theoretical and practical examples, the students will learn to apply the methods in modern system design. Topics are the following, wherein most of those MORPHEUS can be taken as an illustrating example:

- Target architectures of HW/SW-systems
DSP, microcontrollers, ASIPs, reconfigurable Systems, reconfigurable SoCs, FPGAs, ASICs, System-on-Chip
- Estimation of design quality
Hardware- and software-performance, performance trade-offs between coarse and fine grained reconfigurable engines
- Methods for hardware/software partitioning
- Interface and communication structures
Abstraction of different communication principles to meet a homogeneous view from the programmers side
- Co-simulation and rapid prototyping
Testing using simulation tools or FPGA prototyping.
- System design and specification languages (SystemC); E.g. the MORPHEUS simulator is demonstrated.
- Theoretical and practical exercises, case studies (such as MORPHEUS)

1.6.2 „SoC laboratory“

A hardware system is designed in that course that is able to decode OGG Vorbis coded audio streams. Many similarities exist within the development process and the system structure between that course and MORPHEUS. This will be outlined in the subsequent section. The course has been set up in the end of 2006 and is currently already running 2 times a year with 10 to 20 participants.

Relations of the course “SoC laboratory” to MORPHEUS

Acquiring knowledge about system modelling, simulation and verification by using state of the art simulation, synthesis and layout tools are the key components of the laboratory. The laboratory has already been described in deliverable 8.2. Because of that only a brief summary is given here emphasising the similarities with MORPHEUS.

The development tools within the laboratory are the same as in the MORPHEUS modelling approach. (E.g. Modelsim or GDB debugger) Also the same state of the art bus protocol is used in the SoC laboratory, namely AMBA 2.0. The system on chip covers even further development topics as such analogue design using TSMC standard cells. The digital design is realized in a top down approach, first by modelling a simulator for design space exploration and later on by implementing RTL code using the Modelsim environment. The system is verified on the basis of software test benches running on the Simulator, which is again the same approach as in MORPHEUS. In contrary to MORPHEUS no chip will be produced here but a first test of the system is done on an FPGA basis. Also specification constraints such as area and power are taken into account and lead to several redesign steps. Summarizing, it can be seen that state-of-the-art SoC development has been adopted in that laboratory; to give the students a very close idea to design processes within actual research and development projects, in particular MORPHEUS.

2. Autumn school featuring MORPHEUS research results

At the AETHER-MORPHEUS workshop and school in autumn 2008, actual projects results have been presented. These have been the basis for several novel courses as already described in the previous chapter. Around 20 students have been following these lectures. In the following, the program of that event is outlined. Also some exemplary abstracts are illustrated to communicate the broadness of the addressed topics within MORPHEUS to the students. The presented topics are a homogeneous mixture of the main work packages of MORPHEUS: hence about the developed toolchain, the architecture and the applications.



Figure 2: AETHER – MORPHEUS Workshop and school, October 7-9, 2008, Lugano, Switzerland; Web: <http://www.alari.ch/AMWAS08/>

The Second AETHER - MORPHEUS Workshop- Autumn School From Reconfigurable to Self - Adaptive Computing (AMWAS'08) was organized in Lugano, Switzerland in October 7-9, 2008. The innovative "Workshop-Autumn School" format has been chosen to provide participants with both grounding on a new, challenging scientific area and with exposure to research results and proposals. The AMWAS'08 will constitute a meeting-point for researchers and graduate students, interested in innovative next-generation computing architectures.

2.1. AMWAS 2008 - Programme

2.1.1 Autumn School

Software aspects: Toolchain and System Software

AETHER: μ threaded run-time environment for adaptive Programming
Chris Jesshope

MORPHEUS: MORPHEUS Global Toolset overview
Philippe Millet, Thales, France

AETHER: The S-NET Environment
Clemens Grelck, University of Hertfordshire

Exploitation of reconfigurability and Chip Design: The hardware aspects

MORPHEUS: Implementation of heterogeneous multi-core processors
Fabio Campi, ST, Italy

AETHER SP1 Design of adaptive hardware using run-time reconfigurable devices
Jean-Marc Philippe, CEA LIST, Martin Danek, UTIA

MORPHEUS: Application Scenarios in MORPHEUS
Sean Whitty, Henning Sahlbach, TU Braunschweig, Germany
Cyrille Batarriere, Thales, France

2.1.2 Workshop

Session1: **MODEL ENVIRONMENT FOR SELF-ADAPTIVE SYSTEMS**

Spatial design backend: CDFG mapping on eFPGA and DREAM IPs
Loic Lagadec and Damien Picard (University of Brest, France)

On Simulating Operating Environment Decisions in a SANE Network
Milad El Khodary, Jean-Philippe Diguët, Guy Gogniat, LabSTICC, Université de Bretagne-Sud, UEB - CNRS, Lorient, France
Fabrice Muller and Michel Auguin, LEAT, Université de Nice-Sophia Antipolis - CNRS, Valbonne, France

S-Net Type System and Operational Semantics
Haoxuan Cai, Susan Eisenbach, Dept of Computing, Imperial College London, UK
Clemens Grelck, Informatics Institute, University of Amsterdam, Netherlands; Dept of Computer Science, University of Hertfordshire, UK, Frank Penczek, Sven-Bodo Scholz and Alex Shafarenko, Dept of Computer Science, University of Hertfordshire, UK

Session 2: **ARCHITECTURE ISSUES FOR SELF-ADAPTATION**

Heterogeneous Multi-Core Architecture: The Added value of Physical Design
Fabio Campi, STMicroelectronics, Italy

LOW-LEVEL DESIGN OF A SELF-ADAPTIVE SYSTEM
Javier Soto, J. Manuel Moreno, Jordi Madrenas, Joan Cabestany, UPC, Spain

Evaluating flexible CGRA cells
Giovanni Ansaloni, Paolo Bonzini, Laura Pozzi, University of Lugano, Faculty of Informatics, Lugano, Switzerland

Round Table / Panel : **Preparing AETHER and MORPHEUS aftermath**

Topic: what research and project shall we propose?

Session 4: **PRACTICAL HW AND APPLICATION ISSUES**

Self-Adaptive Computing Networked Entities for DSP: Evaluation of One Possible Implementation

Jiri Kadlec, Martin Danek, Roman Bartosinski, Lukas Kohout
UTIA AV CR, v.v.i., Pod Vodarenskou vezi 4, Praha 8, Czech Republic

Enabling Self-adaptivity at Application Level

Onur Derin, Alberto Ferrante, ALaRI, Faculty of Informatics, University of Lugano, Lugano, Switzerland

Mapping of a Film Grain Removal Algorithm to a Heterogeneous Reconfigurable Architecture

Sean Whitty, Henning Sahlbach, Rolf Ernst, Technical University of Braunschweig, Germany
Wolfram Putzke-Röming, Deutsche Thomson OHG, Germany

Session 5: SYSTEMS CONCEPTS

Embedding Self-Adaptivity in Future Computing Devices using the SANE Concept: Model, Implementation on Reconfigurable Hardware and Example Application
Jean-Marc Philippe, Benoît Tain, Christian Gamrat, CEA, LIST, France

Dynamic Reconfiguration of Nano Architectures using Application Independent Fault Detection

Mahtab Niknahad, Christian Schuck, Michael Hübner, Jürgen Becker, Universität Karlsruhe (TH), Germany

A Guarantee of Service Protocol for Pervasive Distributed Systems

Alberto Ferrante, Roberto Pompei, Anastasia Stulova, Antonio Vincenzo Taddeo, ALaRI, Faculty of Informatics, University of Lugano, Lugano, Switzerland

Session 6: Run-Time Reconfiguration

Running S-Nets on Shared Memory Multicore Systems

Clemens Grelck, University of Hertfordshire, Department of Computer Science, United Kingdom; University of Amsterdam, Institute of Informatics, Amsterdam, Netherlands
Frank Penczek, University of Hertfordshire, Department of Computer Science, United Kingdom

Towards Reconfiguration and Self-Adaptivity in S-Net

Frank Penczek, Sven-Bodo Scholz, University of Hertfordshire, Department of Computer Science, United Kingdom

Clemens Grelck University of Hertfordshire, Department of Computer Science, United Kingdom; University of Amsterdam, Institute of Informatics, Amsterdam, Netherlands

Graph Walker: Implementing S-Net on the Self-adaptive Virtual Processor

K. Bousias, C. R. Jesshope, Institute for Informatics, University of Amsterdam, The Netherlands

J. Thiyagalingam, S. Scholz, A. Shafarenko, Department of Computer Science, University of Hertfordshire, United Kingdom

Final remarks: what's next (Brainstorming and discussion of the coordinators of AETHER and MORPHEUS)

2.2. Selected abstracts of AMWAS 08

2.2.1 Spatial design backend: CDFG mapping on eFPGA and DREAM IPs

The spatial design task of the Morpheus's WP2 aims at providing a unified implementation flow, starting from a set of C functions seen as processes to be composed, and allowing to address two reconfigurable IP targets: fine grain M2000 eFPGA and medium grain DREAM from STMicroelectronics/ARCES.

In this context, UBO provides an open middleware layer, allowing describing the applications as Control/Data Flow Graphs (CDFG). CDFGs support standard textual representation and the CDFG's API is provided for several programming environments (Java, C++, ...) so that any tools such as SPEAR or CASCADE can build/read/write and exchange CDFGs. CDFGs are then used to feed the Madeo+ synthesizer.

Madeo+ owns capabilities to output scheduled FlexEOS compliant RTL netlists (EDIF). This output is processed by the FlexEOS tool chain for both reporting (timing, allocation, verilog rewriting...) and bit stream generation.

Madeo+ also supports code production to output C and Griffy-C code. The DREAM platform is composed of two execution engines; the embedded processor is programmed by classical C code (linked to a library that includes the API for controlling the interconnection matrix configuration, the address generators, loading and executing accelerated functions on the PicoGA) while the accelerated functions are described in a subset of C (focusing on data flow computations description) called Griffy-C. Both codes are taken as input of the DREAM compiler and validated in the DREAM simulator.

In addition to synthesis and code generation, some features such as high-level simulation have been developed as programming and debugging facilities. Such a simulation relies on a multi-level CDFG simulator, integrated within a system simulator enabling to take into account activities from outside of the IPs such as bus transfers and contention, DMA synchronizations or memory activities.

This talk highlights the UBO middleware layer functionalities through some examples, with a specific focus on synthesis issues, data transfer management (DMA-handshake local controllers) and synthesis' outputs validation with mainstream tools (ModelSim and DREAM simulator).

2.2.2 Heterogeneous Multi-Core Architecture: The Added value of Physical Design

Reconfigurable computing holds the promise of delivering ASIC-like performance while preserving run-time flexibility of digital signal processors. On the other hand, fine grained reconfigurable devices such as FPGAs often suffer area, power, and timing overheads, while Coarse-Grained Reconfigurable Architectures offer higher computation density, but at the price of being rather domain specific. A promising approach towards a general purpose yet highly performing signal processing platform is to build a Multi-core SoC architecture where different computing devices (processors, DSPs, ASIC accelerators, fine grained fabrics, coarse grained fabrics) are merged together to build a single computing engine. The main challenge in this appealing domain is three-fold: it is necessary to build a solid toolset, a robust and user friendly system-level control, synchronization and interconnect platform, and to choose an implementation strategy that is able to cope with the intrinsic complexity of the target architecture. In particular, the last two points should be challenged as a single task, as it is not conceivable to design a multi-core, million-gates architecture without taking into account from the very beginning implementation issues. On the contrary, abolishing the old "Front-End Versus Back-end" barrier is a new and exciting opportunity for the design of a heterogeneous multi-core system. It can represent an innovative added value and a powerful enabling factor rather than a mere implementation phase where the only target is not to deteriorate too much Front-End specifications. In this paper, a few opportunities emerged in the research related to Morpheus project are described. In particular, research related to the implementation activity focused on the opportunities offered by a NoC-centric approach, and the consequent partition of the design in independent logic blocks. Hierarchical design, definition of separate frequency and Voltage islands to allow Dynamic frequency and voltage scaling (DFVS),

asynchronous communication and system level synchronization are the most significant implementation strategies investigated during the project. In the case of such a complex design, the definition from the start of a computation model that is strictly related to the implementation choices allow for significant advantages in implementation cost, design time and effort, performance and power results.

The Morpheus architectural concept is based on a set of computational nodes: its most peculiar feature, as compared to other MPSoCs, is the relevant computation density and granularity of the nodes, both in terms of chip area and of computation throughput. Morpheus is a scalable platform, but the number of nodes it can support is typically in the order of 2/3 to 10. Also, the nodes typically iterate the same computation for "long" times over different sets of data to mitigate the cost of reconfiguration that is larger than that of a standard processor thus leading to a regular traffic. Example of Computation nodes are the ARM926E RISC core, the XPP processor, the DREAM processor and the M2K eFPGA. Other IPs can be embedded DSP cores, ASIC accelerators or other kind of signal processing engines.

The objective of the research related to the physical design is to merge together in a single System-on-chip instance the IP-blocks mentioned above, based on different specifications, programming styles, and hardware features. In order to build a consistent silicon platform it appeared necessary to implement a loose coupling strategy, composed of independent logic entities that would be implemented on independent physical entities.

On the one hand, the clock speed of the different engines can be related to the different IP design options, and the different area/speed trade-off chosen by the respective vendors. Also, in the case of hardware reconfigurable engines often the final clock speed is strongly dependent on the application mapping. On the other hand, also when the clock speed is independent of the application, it appears not useful to utilize a given IP at top speed when such the maximum computation power is not required. A similar line of reasoning can be followed when evaluating the feed voltage: considering the shmoo plot related to each IP, a given IP will surely need nominal (or higher) power supply, regardless of the related leakage and dynamic consumption, when it is computing the critical kernel of a given application. But it is likely that not all IPs will be critical at the same time, so that significant power saving can be obtained down clocking a given IP and feeding it with the minimum voltage that can guarantee functionality at that speed.

This kind of dynamic control, defined in literature DFVS (Dynamic Frequency and Voltage Scaling) can be a very powerful enabling factor provided by physical design. On the other hand, the exploitation of DVFS is possible only orienting from the very beginning design choices in that specific direction. Indeed, an IP-based heterogeneous architecture may greatly ease this option due to its block-oriented structure. The critical design features in this context are the chosen interconnect and memory hierarchy strategy. In particular, in the Morpheus platform data interconnect is made intrinsically scalable by the utilization of network-on-chip. Moreover, each IP is logically structured as a single block, whose only communication with the rest of the system is via dual-clock buffers for data, configuration and control. Each IP (ARM, DREAM, XPP, M2K or any other) can be then physically implemented as a separate layout and then reused as a library in the top level design. Also, storage nodes such as on-chip RAM and off-chip memory controllers (DDRAM, Flash, SRAM, SDRAM) can be considered as additional independent nodes. The network-on-chip router itself can be designed as a separate layout block and than instantiated repetitively in the design to build the desired network topology.

This block-based hierarchical design organization eases the definition of separate clock and voltage islands. Moreover, it significantly relieves the design effort breaking the complexity of a single flat multi-million gates place and route session (with all its CPU requirements and intrinsic complexity) in several smaller designs that can be handled in parallel. Of course, limiting the complexity of the design one can obtain much higher performance results. In particular, the definition of independent clock domains leads to smaller and more distributed clock trees in place of a very deep and complex one, thus mitigating one of the most critical causes of power consumption and IR-drop in SoC design.

In the reminder of this paper, the techniques investigated for the deployment of DFVS and hierarchical design to the implementation of heterogeneous MPSoCs are described, and related Design options and implementation choices will be discussed.

2.2.3 Mapping of a Film Grain Removal Algorithm to a Heterogeneous Reconfigurable Architecture

Despite recent advances in FPGA, GPU, and general purpose processor technologies, the challenges posed by real-time digital image processing at high resolutions cannot be fully overcome due to insufficient processing capability, inadequate data transport and control mechanisms, and often prohibitively high costs. To address these issues, we proposed a two-phase solution for a real-time film grain noise reduction application. The first phase is based on a state-of-the-art FPGA platform used as a reference design. The second phase is based on a novel heterogeneous reconfigurable computing platform that offers flexibility not available from other computing paradigms. This paper introduces the heterogeneous platform and briefly reviews our previous work with the application in question, as well as its implementation on the FPGA demonstration board during the first phase. Then we present a decomposition of the application, which allows an efficient mapping to the new heterogeneous computing platform through the use of its diverse reconfigurable computing units and run-time reconfiguration.

3. Mapping of proposed Courses from Deliverable 8.2 to actually developed courses

The courses described above are mapped to requirements and courses as specified in deliverable 8.2. The following table summarizes that mapping. The first column shows the courses as defined in Deliverable 8.2. If an adequate mapping exists to these proposals, the corresponding developed and adapted courses are mentioned in the second column and the title of the training course is marked green. It can be seen that all nine topics are already covered in students' curricula by the MORPHEUS university consortium, which is a very good result.

Training for HW-Designers/Engineers: Course title	Courses, mapped to the contents	Content of course
Coarse-grained reconfigurable architectures	<ul style="list-style-type: none"> • RA&TD (UBO) • SRA (UBO) • Circuit Design (CUT) • Computer architectures II (TUBS) • Training and Lab on DREAM architecture and tool-set(ARCES) 	<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	<ul style="list-style-type: none"> • HW/SW Co-design (UK) • SoC laboratory (UK) • EDA-Tools (CUT) • System level integration of reconfigurable architecture (ARCES) 	<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		Content of course
Parallel computing on reconfigurable architectures	<ul style="list-style-type: none"> • SoC laboratory (UK) • EDA-Tools, Systems Design, Rapid Prototyping (CUT) • Computer architectures II (TUBS) • Parallel computing on reconfigurable architectures (ARCES) 	<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	<ul style="list-style-type: none"> • Network-on-Chip architectures (ARCES) 	<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		Content of course
Hardware-Software Co-design for reconfigurable SoCs	<ul style="list-style-type: none"> • EDA-Tools, Systems Design (CUT) • Parallel computing on reconfigurable architectures (ARCES) • Training and Lab on DREAM architecture and tool-set (ARCES) • Network-on-Chip architectures (ARCES) 	<ul style="list-style-type: none"> • Methods and algorithms for HW/SW partitioning • Graph Algorithms • 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 the MORPHEUS architecture – trade-off and benefits	<ul style="list-style-type: none"> • HW/SW Co-design (UK) • RCD (TUD) • Training and Lab on DREAM architecture and tool-set (ARCES) 	<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	<ul style="list-style-type: none"> • SoC laboratory (UK) 	<ul style="list-style-type: none"> • Memory usage in heterogeneous SoC architecture • Example: DMA setup and buffer

		handling <ul style="list-style-type: none"> Platform synthesis and analysis of MORPHEUS architecture
Applications for MORPHEUS architecture and project planning	<ul style="list-style-type: none"> Rapid Prototyping (CUT) RCD (TUD) Training and Lab on DREAM architecture and tool-set (ARCES) 	<ul style="list-style-type: none"> Telecommunication standards Video data processing Project handling: How to steer complex projects
Training for MORPHEUS end users: Course title		Content of course
Application notes with MORPHEUS reconfigurable SoC	<ul style="list-style-type: none"> Simulators user guide in Deliverable 3.5 and 3.6 Tool chain users guide in Deliverable 2.5.1 (toolset integration report) 	<ul style="list-style-type: none"> Introduction to application scenarios Introduction of tool chain for MORPHEUS Description of flexibility through reconfiguration Reconfiguration paradigm Exemplarily: Integration of one example application on MORPHEUS platform (from WP5)

Table 1: Mapping table: Requirements vs. Achieved Courses

4. Conclusion

It can be seen from chapter 1 that a number of courses have changed with respect to MORPHEUS. In most of these MORPHEUS achieved to bring curricula topics closer to research. This is due to the fact that students are now able to link a name (MORPHEUS) to certain design methodologies, to a defined architecture and to well known problems that come along with embedded system development. Chapter 4 finally summarizes the achievements within WP8. It compares identified necessary courses/material that has been defined within WP8 (in D8.2) with adopted courses towards these needs. Table 1 shows that all of the nine identified topics are covered in students' curricula up to now, which is a respectable result.