DORADO
Dataflow ORiented Automated Design tOolchain

Consortium leader: Prof. Johan Lilius (Åbo Akademi University)
Sub-project leaders: Prof. Olli Silvén (University of Oulu)
Prof. Jarmo Takala (Tampere University of Technology)

1. Introduction

Embedded systems are evolving from one generation to the next in an increasing pace. For decades, single processor embedded systems have been dominant, but now it is clear that multiprocessing platforms will overcome traditional single-processing solutions in devices that require high throughput and/or low power. In addition, recent progress in low power reconfigurable technologies have shown that reconfigurable hardware is becoming a viable implementation platform also for low-energy mobile embedded devices.

Manually porting applications and algorithms to new multi-processing environments is an unmanageable task. Thus, highly automated, platform-independent software development tool chains that produce "performance portability" across different platform variations are compulsory. In this project we plan to address this problem by creating a tool chain for automatically generating embedded software/hardware solutions from high-level specifications that are platform independent and describe parallelism at all its main granularities (data level, instruction level, task level and memory level parallelism).

Special attention is paid to many-core systems that are expected to be a central approach in improving the computing throughput in the future due to reaching the limits of single core processors. As configurable platforms are increasing in popularity, the automation of the hardware design process is emphasized, ultimately aiming for the generation of efficient manycore application-specific processors.

2. Background

2.1 The Concurrency challenge

Current mainstream programming languages (e.g., C, C++, Java) are inherently sequential by their nature. Abstractions provided to deal with concurrency, e.g. threads, mutexes etc, do not scale well ([1]). The *Concurrency Revolution* as coined by Sutter [2], is currently the biggest single challenge for writing software.

In the spirit of [3] we believe that workloads can exhibit substantially different kinds of concurrency patterns, and that these patterns may require very different description techniques. This means that there is not one single solution to programming many-core machines, but such solutions should be built to match particular (large) classes of workloads. For embedded systems, typical applications like multimedia, sensor networks, software defined radio etc, are most naturally described by describing how the data flows in the application, rather than emphasizing the control structure (as one is forced to do when using C/C++).

Dataflow programming is an inherently parallel way of describing computer programs. Dataflow languages provide several advantages over traditional languages. A dataflow network consists of actors and channels or buffers that are used to transfer data between the actors. A dataflow network is inherently concurrent. The data dependencies describe the causal relationship between the computations. Ever since the pioneering work of Lee and

Messerschmitt [4], synchronous dataflow (SDF) has been the prevailing modeling technique in signal processing. More recently, Bhattacharya and Bhattacharyya have extended SDF modeling to be suitable for a larger variety of applications by introducing Parameterized SDF (PSDF) [5]. Several dataflow based languages have been proposed in the literature. For the current project most relevant are CAL, Canals, and OpenCL.

Within the dataflow paradigm, the CAL actor language offers expressiveness beyond PSDF and it has been chosen as the specification language of the recent Reconfigurable Video Coding standard [6]. The CAL language was originally developed at UC Berkeley, but recently the development activity has mostly shifted to INSA Rennes in France. Researchers of INSA Rennes have created a compiler named ORCC for processing CAL actor networks into various other languages to generate executable implementations.

The mapping of a dataflow program onto a particular computation platform requires the computation of a schedule, and implementation of such a schedule on the platform. Canals [7] is a research language that tries to address some of the current shortcomings of CAL by providing a more structured language for expressing networks, and by providing a framework for describing scheduling algorithms.

Another promising candidate, OpenCL, has recently been standardized and adopted by the major computing platform vendors [8]. In comparison to CAL, the programming model of OpenCL is closer to the traditional C language but still provides the capabilities to describe the parallelism in all its levels with a single application description. OpenCL programs can also be modeled as dataflow programs due to the possibility of event driven execution of kernels.

In this project, the Canals language is a research vehicle used to experiment with different language concepts. These concepts can then be provided to CAL for standardization in the next generation of the language. Finally, OpenCL provides a standardized interface that we hope will enable portable code generation for off-the-shelf many-core architectures.

2.2 From a single core to hundreds of cores

In the recent years, due to starting to hit the limits in improving the single thread program performance, there has risen significant interest towards exploiting multiple cores to improve application throughput. We have seen the emergence of *many-cores* where tens or even hundreds of independent processor cores execute the same throughput-oriented application.

The switch in terminology from *multi-core* to many-core can be considered to happen at the point when there are so many cores that a single shared global memory with consistent per-core local caches is not anymore feasible due to the increased traffic in the shared memory bus. Therefore, more complex memory hierarchies such as cores with fast addressable local memories (software caches or “scratchpad memories”) are exploited in order to keep the cores busy with data to process.

Efficient exploitation of local memories adds to complexity in programming and compiling for many-cores. The common traditional thread programming models (such as the *pthreads* C API) where only a single shared address space can be assumed is unfit when programming devices with multiple cores with their own private memories. In addition to the explicit programming models, the research in automated exploitation of the local memories is important in the context of performance portability. The application should be able to be written in such a way that it can be easily ported to various styles of memory architectures. In this job the compiler plays a crucial role.

2.3 Existing automated design flows

Metropolis [9] is a somewhat aged environment for simulation, analysis and synthesis of applications based on a computational meta-model that allows describing components with different languages. *Ptolemy* [10] is very similar to Metropolis and concentrates mostly on simulation of systems that are compositions of *domains* described with different semantics. Compared to the proposed project, Metropolis and Ptolemy stay on a more abstract level and do not consider the actual generation of efficient implementations.

PeaCE [11] is a codesign environment for generating hardware and software from a single specification. One of the main differences to the DORADO project is that in PeaCE the generated software is C code. The pitfall of generating C code is that it creates low-performance programs when targeted to parallel processing platforms due to its sequential nature. This is avoided in DORADO by the use of higher level data flow languages and the use of compiler internal representations that maintain the information of all parallelism in the described program down to the implementation level.

2.4 Background of consortium parties

Prof. Olli Silvén's group at the University of Oulu has been working on dataflow scheduling for several years. The research started as cooperation with Prof. Bhattacharyya's group and has evolved into the scheduling of CAL networks together with researchers of the École Polytechnique Fédérale de Lausanne (EPFL) and INSA Rennes. Recently, the group has been working on automatic analysis of CAL dataflow networks for the purpose of improving their run-time performance.

The research group of Prof. Johan Lilius in Åbo Akademi University has been developing domain-specific languages for over 10 years. The emphasis has been on the one hand using meta-modeling as a tool for fast generation of domain-specific languages, and on the other hand on several aspects of formal languages intended at expressing computations in several models of computation and concurrency (MoCCs). Rialto [12], a formal language for heterogeneous computations, as well as Canals [7], a streaming language allowing explicit specification of communication patterns, data-flows and in particular scheduling strategies at different levels of granularity, are currently developed by Prof. Johan Lilius' group. Part of the research was carried out in cooperation with Prof. Bhattacharyya's group from the University of Maryland, Dr. Mickaël Raulet from INSA Rennes and Dr. Lionel Morel from INSA Lyon. The group is an academic member of the Multicore Association (MCA).

Prof. Jarmo Takala's research group at TUT has been working on Transport Triggered Architecture processors for almost 10 years now. The Transport Triggering concept allows creating application specific processors (ASP) that have energy efficiency close to a fixed function ASIC implementation, but that still maintain programmability. Due to the architecture being static, this research has mostly concentrated on compiler code generation issues. In the course of this research, Prof. Takala's group has developed a processor design toolset named TTA-based Codesign Environment (TCE) to allow engineers to design processors, compile software efficiently and synthesize the created processors on an FPGA or ASIC. The toolset has been published as open source attracting interest from both the academy and the industry. The compiler for TTA processors is based on the LLVM compiler infrastructure, so it is directly targetable by the ORCC compiler, thus makes a perfect fit as an FPGA or ASP backend for the planned toolchain.

The technological concepts and ideas presented in this proposal have preliminarily been investigated in TEKES funded projects NECST (2006-2007, partners TUT, OU, ÅAU, and Helsinki Univ. Tech.), ECUUS (2008, partners: TUT, OU, ÅAU) and the Academy Funded project PARSEC (2009-2011, partners: TUT, OU, ÅAU).

3. Objectives

The high-level objective of this project is to produce a **toolchain for synthesizing executable system implementations from high-level specifications**. The high-level specification approach is based on dataflow and the produced implementations cover a variety of systems, while paying special attention to many-core and reconfigurable solutions. The approach will not be limited to any particular target nor input language but proposes an internal representation that can be easily mapped to *graphics-processing units (GPU)*, *symmetric multiprocessors (SMP),* and automatically generated *application-specific manycores (ASMC)* based on the TCE technology. Several input languages will be considered and their advantages compared.

We claim that, based on our expertise, it is possible to automatically generate implementations from high-level specifications whose performance (energy-efficiency) is in the same magnitude as that of a *manually designed system.* It is clear that manually designed embedded systems can be optimized to a level that is beyond the reach of automatic approaches, but the manual design process is often so slow that the product is already outdated at the time when its design has been finished. Thus, automatic approaches (even if sometimes producing sub-optimal results) have a clear advantage in terms of time-to-market.

Our consortium is going to use the CAL compiler ORCC developed at INSA Rennes as a one input to the design flow. At present, the CAL compiler reads applications described as CAL actor networks and outputs LLVM bytecode from the compiler backend. The consortium plans to continue from this work in two different directions: a) **developing analysis methods to produce tools to automatically interpret the behaviour of the application** and b) **use this information to streamline the application performance**. This is very relevant, as the present-day software generated from a CAL network is notably slower than a respective handwritten application. Second, our consortium plans to use the LLVM bytecode output generated by ORCC to generate code for the several platforms outlined above in order to produce concrete benchmark numbers and proof-of-concept implementations.

The CAL language appears to be an excellent match with the consortium objectives. However, it is quite certain that during the course of research, the language will show some shortcomings. Due to this reason, our consortium is also going to **continue researching a dataflow language of its own**: Canals [7]. The development of Canals started a few years ago at Åbo Akademi University. Canals aims to facilitate code generation for heterogeneous platforms, primarily in the data flow driven application domain. Canals language describes the intended behaviour of an application in a platform independent manner. All elements exist in parallel and are capable of performing computations concurrently from a resource point of view. Only true data dependencies restrict the parallelism.

Regarding the code generation for many-core systems, the consortium plans to improve on the state-of-the-art by attacking the problem of the memory bottleneck. Many-core systems are challenging to program largely due to the immense memory bandwidth requirements of the hundreds of processor cores. One way of attacking this problem is software controlled memory hierarchies. At the task level, energy-efficient use of hierarchical memories requires sophisticated **scheduling of the shared memory accesses**, which has not been studied yet to an adequate extent. On the single core level, the goal is to **study software-controlled caching** and relaxed memory consistency models to minimize the unneeded memory traffic and synchronizations depending on the constraints set by each used input language.

3.1 Benefit of having the consortium

The three parties of the consortium (ÅAU, OU, TUT) consist of researchers that complement each other's skills and knowledge. Prof. Johan Lilius' group at ÅAU has long-term experience in embedded software engineering and modeling, as well as theoretical computer science. Prof. Jarmo Takala's group at TUT is unique in the Finnish field of research as it has been working on processor architectures and compiler design, topics which have traditionally been mostly addressed overseas. The group of Olli Silvén at OU has been working on various aspects of signal processing and scheduling. Lately, the group has addressed scheduling of heterogeneous systems and analysis of CAL networks in several publications. Figure 1 outlines the sharing of responsibilities between the consortium parties.

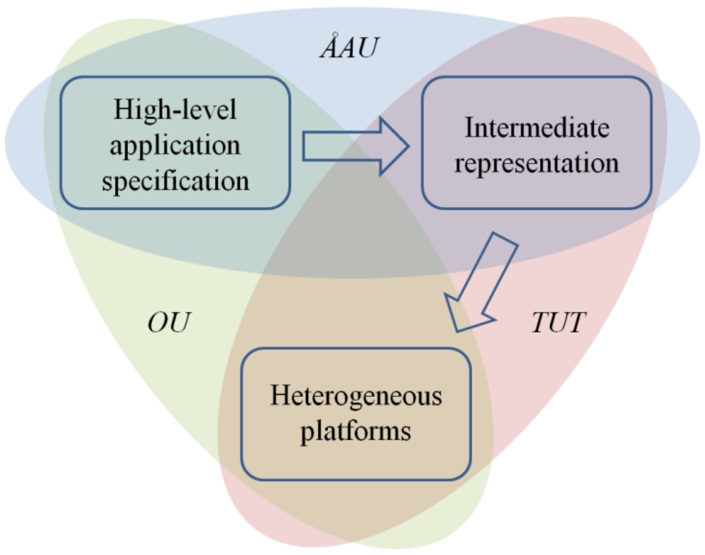
ÅAU
High-level application specification
Intermediate representation
OU
TUT
Heterogeneous platforms

Figure 1. Responsibilities of the partners

The cooperation has already led to industrial research and development: the results of the NECST, ECUUS and PARSEC projects of the consortium are employed in the industry driven EFFIMA (Energy and Life Cycle Cost Efficient Machines) program in implementing ultra-low-power wireless sensor nodes for industrial monitoring. The work is funded by FIMECC that is a *strategic centre of science, technology, and innovation (SHOK)*. EFFIMA program aims at technologies and solutions that dramatically reduce the life cycle costs of new machines, devices and systems compared to the current ones. Also the fundamental problems in the design tool chain, such as the need to write hardware dependent software code that is difficult to port to new platforms, when the available implementation technology changes, are vital issues from the FIMECC point of view.

Furthermore, the earlier cooperative projects have enabled a substantial number of industrial development-oriented master´s thesis works ranging from video codec to digital receiver designs. These have been excellent proving grounds for the tool chains under development.

4. Research methods and material

The proposed project is a joint undertaking of three university partners that possess complementary expertise in the field. The work is divided into work packages according to the phases of the automated design flow.

WP 1 High-level application specification

The potential of a design tool chain is determined by the application specification approach it uses. The programming language, and thus the model of computation, affects the tool chain beginning from the user interface down to the final code generation. Desired properties of modern-day application specification languages are *portability*, *analyzability*, *verifiability* and *explicit concurrency,* which are partially contradictory qualities. Based on the latest knowledge, dynamic dataflow languages such as CAL or Canals, offer the best match for these requirements.

In this project, our intention is to use CAL and Canals as starting points for our proposed automated design flow. CAL, and the present set of development tools related to it, offer a portable, concurrent way of expressing applications and provide a basic design flow down to hardware and software implementations. However, the language and its present-day toolset offer practically no way for verification or systematic analysis, and its support for data-structures and scheduling is weak. Canals on the other hand, provides strong support for data-structures and scheduling specifications, while having a more restricted language for describing network structures, and weak tool support.

Our goals are thus twofold:

1. Develop static analysis methods for data-flow languages that allow us to verify the correctness of the program, and to find maximally concurrent schedules that can be then used as input to an optimization phase that takes other constraints into account
2. Explore new language structures that can be used to make the static analysis methods more accurate and thus provide more information that can be used in the later stages of the design flow.

Task 1: Calculation of schedules using network unfolding
As there are already a plethora of existing languages that offer the possibility for verification and analysis, our first aim is to translate CAL specifications into another language that already offers these properties. Preliminary investigations have revealed that translation of CAL into *Promela* used by the state-of-the-art model-checker *SPIN [13]* tool is possible with little effort.

Besides verifying the correctness of high-level specifications, the WP1 of our design flow must also provide a possibility to optimize the application model as far as possible, before compiling the model to an intermediate representation. This top-level optimization requires a consistent computational view of the whole system, which might be considerably different from the model designed and organized by the human designer. As both CAL and Canals specifications can be seen as communicating Finite State Machines (FSM), the method of *unfolding FSMs* is seen as an approach to reveal the complete state space of the specification for analysis and optimization.

At the moment, the most advanced approaches of analyzing and optimizing CAL specifications are based either on *classification* of individual CAL actors into more restricted models of computation, or dynamic (run-time) analysis of the specification. Both approaches have their shortcomings and lack in generality. *Unfolding* of the FSM network avoids these pitfalls and offers a theoretically sound basis for the specification analysis. The potential problem of the Unfolding approach is its large state-complexity. We plan to alleviate this by developing static-analysis schemas, and using abstract interpretation.

Task 2: Analysis of memory transfers
Work Package 1 is also required to produce information about the internal communication of the different parts of the application specification. Later phases of the design flow need this information for mapping the specification to different processing elements and for optimizing the memory traffic. CAL does not at the moment provide ways to annotate the consumption and production of an actor, instead this information has to be obtained by profiling. Canals on the other hand currently provides both annotations for consumption and production, as well as well defined data-structure specifications that allow the calculation of the memory requirements of a token of data. Static analysis is by nature always conservative. Thus we plan to develop an approach that uses both static analysis, as well as run-time profiling techniques to obtain as accurate estimates as possible. We will develop heuristics that hopefully can be automated that use the run-time profiling information to add accuracy to the static analysis. Finally we will explore if these heuristics can be encoded into Canals, so as to allow the programmer to annotate to program with this information, and thus increasing the accuracy of the analysis.

Task 3: Efficient schedule representations
Synchronous dataflow (SDF) is an extensively used model of computation for communications, signal processing and multimedia applications. As such, the SDF model provides valuable properties for the analysis and the verification of a system. However it can only be used to model applications having a static behavior, i.e. applications that can be modeled with a static dataflow graph topology containing dataflow graph vertices (actors) having only a static data production or consumption rate.

Several attempts have been made to provide a high-level dataflow oriented model of computation to model applications containing dynamic behaviours. For example, PSDF increases the expressive power of the base SDF model by allowing actor or edge parameters to be reconfigured at run-time. However, as such, PSDF does not provide the possibility to describe execution modes within a system and therefore will rely on the implementation of run-time scheduling decisions. An execution mode describes a explicit system state defined by specific values for the reconfigurable parameters.

The objective of this task is to explore and evaluate different trade-off between expressiveness and efficient scheduling of dataflow oriented models of computation for applications containing dynamic behaviours, and to develop a new scheduling sub-language for Canals.

WP1 is a joint effort between AAU and OU. AAU has wide-ranging experience in formal methods, programming languages and theoretical computer science in general. OU, on the other hand, has been working on the analysis of CAL specifications for several years, and has produced concrete tools for generating optimized implementations from CAL specifications.

The Work-packages will provide 3 main results:

1. Algorithms for efficient analysis of dataflow networks
2. New language constructs for more efficiently describing applications such that the analysis algorithms can act more efficiently
3. A formal representation of the obtained schedule, that is easy to use as input for WP2

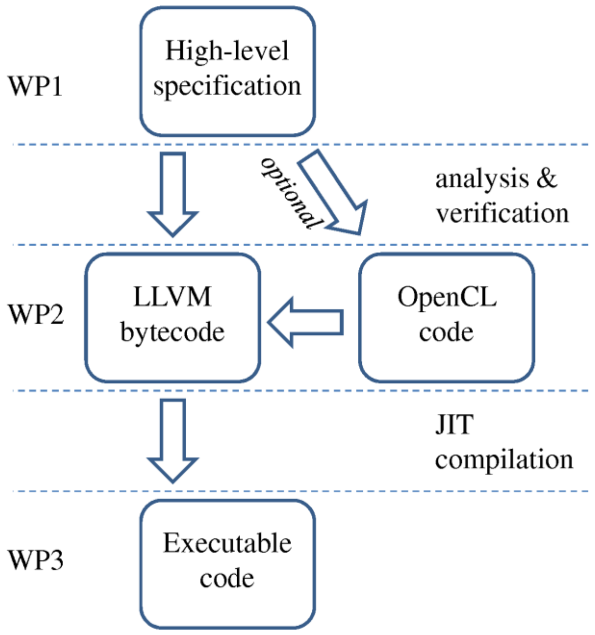

Figure 2. Compilation flow in DORADO

WP 2 Intermediate representations

An intermediate representation (IR) is produced by the compiler from the human-readable high-level specification. The compilers, such as the one developed for this toolchain, use at least one IR on which they perform optimizations and from which the executable code, and in the DORADO case also the hardware implementation, is generated from.

Regarding the DORADO project, there are two relevant IRs. The first one is LLVM bytecode that is produced by the ORCC CAL compiler and used as input by the TCE development environment of TUT. The second IR is OpenCL, which is traditionally not understood as an IR, but as an input language. This being said, OpenCL will be used in DORADO by generating OpenCL code from the high-level specifications (CAL and Canals within this project) and by compiling it for many-core platforms with vendor-specific OpenCL compilers.

LLVM bytecode and OpenCL have somewhat different properties. In the DORADO project, our purpose is to investigate the suitability of these two representations for suitability for the toolchain. It is possible that the research outcome may show that neither of the representations is directly applicable for our purposes, in which case there is a strong motivation to design, or augment, an IR that combines the benefits of both OpenCL and LLVM for this toolchain.

The IR used by the DORADO project must be able to accommodate all the analysis information that has been generated in WP1, and this data has to be stored in a compact and rapidly accessible format.

Task 1: LLVM intermediate representation

The initial task in WP2 is to establish a link between the ORCC compiler that produces LLVM bytecode, and TCE that is capable of taking LLVM bytecode as input. The expected amount of work for basic operation is modest, but transferring performance-enhancing parallelism information between the two tools requires writing extensions to both TCE and ORCC.

Task 2: OpenCL as an intermediate representation

OpenCL is a recent widespread standard for describing highly parallel applications for heterogeneous platforms. In DORADO, OpenCL is used in the toolchain as a portable parallel intermediate representation and also as a target. Ability to generate efficient OpenCL host and kernel descriptions from the higher-level data flow programs enables compiling code to wide range of existing manycore platforms with efficient OpenCL compilers, and also acts as an input to the application-specific manycore design flow of the TCE.

Task 3: DORADO intermediate representation (optional)
Should the intermediate representations provided by LLVM and OpenCL prove insufficient for the DORADO toolchain, a new IR must be derived based on the experiences on the well-known IRs. However, this task will only be carried out if necessary.

WP2 is a joint effort between ÅAU and TUT.

WP 3 Heterogeneous platforms

Heterogeneous platforms have been identified as the best alternative in providing energy efficiency for computing in the future [14]. Multiprocessor System-on-Chip (MPSoC) solutions contain a varying mixture of programmable processors, hardwired accelerators and even FPGA fabric. The latest addition to the collection of MPSoC building blocks are the many-core processing units that can be used for highly parallel applications.

Despite existing for a decade already, the problem of programming heterogeneous MPSoC platforms has been solved only partially. The DORADO project aims to address the challenge of MPSoC programming through the compilation toolchain that has been described in WP1 and WP2. The compiler and the high-level specifications will be used to generate fixed function accelerators and software for single-core to many-core processing units. In addition to only able to generate software, the ability to also create the processing platform itself from the application specification ensures that the interconnect between the processing units, the memory hierarchy and the selection of processing units matches the requirements of the application that is described in the high-level specification.

The heterogeneous platform generation described above contains both issues that have already been solved, and new research challenges that are to be addressed in the DORADO project. JIT compilation of intermediate representations has already been implemented for the LLVM bytecode. Likewise, the already existing ORCC CAL compiler is already capable of producing VHDL (hardwired accelerators) and software programs in the IR format. The TCE toolset is already able to generate VHDL-implementations of single core application-specific processors accompanied with parallelized code.

What mainly remains to be done in the DORADO project, is the automated generation of many-core processing units, the orchestrating of the data transfers in the memory hierarchy and efficient task-level scheduling for the heterogeneous platform.

Task 1: Automatic generation of many-core processing units

The problem of automatically generating many-core processing units is essentially a design-space exploration problem that includes determining the sizes and the banking of the local memory hierarchy, sizes and number of the individual cores, as well as the type and number of possible custom instructions. However, in the context of the DORADO project, with its availability of a parallel high-level application specification, the main challenge is not the design space exploration algorithm itself, but the static and dynamic analysis of the high-level application specification. Our goal is to develop methodologies to discover the optimal point in the design space solely based on the analysis of the high-level application-specification without *compile-simulate-estimate-modify* cycles that can take weeks to complete.

Task 2: Avoiding instruction memory overheads in many-core systems

A special focus of interest in the automatic generation of many-core processing units is the vector style encoding of TTA instructions to decrease the instruction stream bandwidth requirements. Research into this direction has already taken place at TUT and received the Stamatis Vassiliadis Best Paper Award in 2009, encouraging further work.

The practical goal is to reduce the instruction stream size of the generated manycores to such level that the instructions can reside in small local read-only memories on the chip with private local caches, or instruction buffers, enhancing the instruction memory throughput to the adequate level.

Task 3: Avoiding data memory overheads in many-core systems

Orchestrating the data transfers in the many-core memory hierarchy is a central problem, as the memory bandwidth is generally the limiting factor in exploiting the potential of many-cores. Following the successful experiences from the TTA paradigm, the many-core memory traffic is planned to be resolved as far as possible in the off-line compiler (WP1 and WP2), which improves the efficiency of the actual deployed system.

Two key approaches are studied to ensure sufficient memory bandwidth:

1. Compiler-assisted multithreading. When one software thread needs to wait for a slow global memory access, one of the other threads is scheduled to execute, in order to maintain high core utilization. The same idea was studied in a publication preceding PARSEC ([15]) to reduce context switch overheads. As in the preceding work, also in DORADO, the thread scheduler calls and context switch code will be added completely at compile time. Generating bundled global memory accesses accompanied with low-overhead coarse grain thread switches is one specific technique to be developed further.
2. Local, private memories. The goal in DORADO is to automate the use of local core-specific private memories. Contents of local memories are determined by the compiler, which improves energy-efficiency compared to traditional caches [16]. In addition, compiler directed caches are also easier to adapt to the input language's requirements than traditional caches. This is a clear benefit when multiple input languages are allowed.

Task 4: System-level scheduling of the heterogeneous platform

Finally, the top-level operation of the heterogeneous platform must be managed by a suitable system-level scheduling algorithm. It must be assumed that the heterogeneous platform does generally execute applications that have varying workloads, which makes the use of run-time scheduling mandatory. Balancing between the quality of the scheduling and the scheduling overhead is a very challenging topic as the traffic in the memory hierarchy must be considered as a first class citizen in the total schedule. Fortunately, the consortium has tackled such problems before by the use of highly efficient flow-shop scheduling and the use of a scheduling co-processor which provides as a basis for this further research.

Work Package 3 is a joint effort between TUT and OU.

5. Implementation

5.1 Funding

	2011	2012	2013	2014	2015	TOTAL
Salaries (effective working hours only)						
TUT						
Pekka Jääskeläinen (PostDoc)	9 039	37 238	38 353	39 512	27 129	151 271
Carlos Sanchez de la Lama (PostDo	6 026	6 206	-	-	-	12 232
N.N (PhD Student)	7 960	25 348	28 726	34 047	23 375	119 456
AAU						
Andreas Dahlin (PostDoc)	11 040	34 224	35 328	36 432	25 024	142 048
N. N. (AAU) (PhD Student)	7 360	23 184	24 288	25 392	17 664	97 888
Own personal	5 888	17 664	17 664	17 664	11 776	70 656
OU						
Jani Boutellier (PostDoc)	13 246	40 793	41 849	42 905	29 308	168 101
N. N. (PhD student)	8 635	26 960	28 016	29 072	20 085	112 768
Indirect employee costs						
TUT	11 282	33 708	32 869	36 044	24 747	138 650
AAU	12 630	39 037	40 186	41 334	26 790	159 977
OU	10 941	33 877	34 933	35 989	24 697	140 437
Overheads						
TUT	28 818	86 100	83 956	92 067	63 211	354 152
AAU	29 904	92 428	95 147	97 866	65 816	381 161
OU	29 212	90 451	93 270	96 090	65 940	374 963
Travel and visits						
TUT	6 000	30 000	30 000	20 000	10 000	96 000
AAU	5 000	30 000	30 000	30 000	20 000	115 000
OU	4 000	25 000	20 000	20 000	10 000	79 000
Equipment						
TUT	1 000	5 000	5 000	5 000	2 000	18 000
AAU	2 000	5 000	5 000	5 000	3 000	20 000
OU	1 000	1 000	1 000	1 000	1 000	5 000
Funding from the Academy (80%)						
TUT	56 100	178 880	175 123	181 336	120 370	711 809
AAU	59 058	193 230	198 090	202 950	136 056	789 384
OU	53 627	174 465	175 254	180 045	120 824	704 215
Funding from own organization (20%)						
TUT	14 025	44 720	43 781	45 334	30 092	177 952
AAU	14 764	48 307	49 523	50 738	34 014	197 346
OU	13 407	43 616	43 814	45 011	30 206	176 054
Consortium total						
	210 981	683 218	685 585	705 414	471 562	2 756 760

Table 1. The consortium budget in Euro

5.2 Timetable

Figure 2 depicts the planned schedule of DORADO. All work packages will run in parallel, as the intention is to have a working toolchain as early as possible on which further improvements are applied. Based on the experiences of the initial toolchain, more advanced features can be added to the flow as the work progresses.

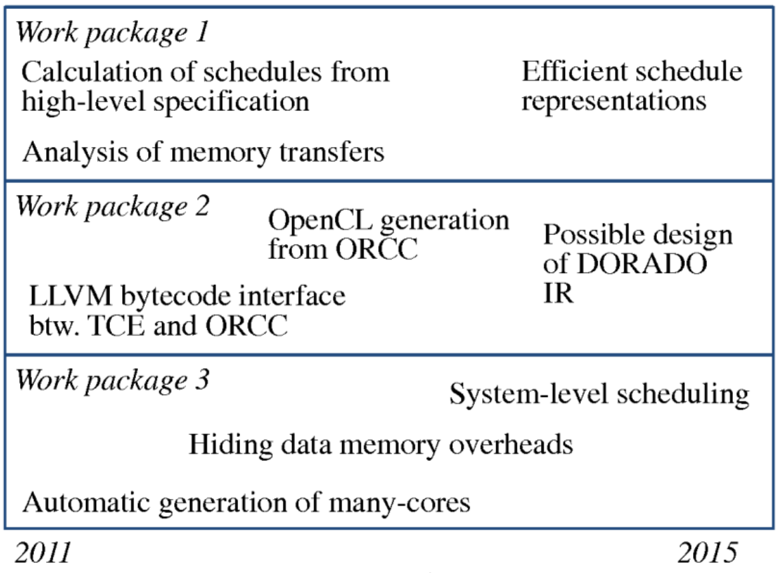
Work package 1
Calculation of schedules from high-level specification
Efficient schedule representations
Analysis of memory transfers
Work package 2
OpenCL generation from ORCC
Possible design of DORADO IR
LLVM bytecode interface btw. TCE and ORCC
Work package 3
System-level scheduling
Hiding data memory overheads
Automatic generation of many-cores
2011
2015

Figure 2. The schedule of the work packages

6. Researchers and research environment

6.1. Members of research teams

Jani Boutellier completed his MSc (2005) and PhD (2009) degrees in the Information Processing Laboratory of OU. While working for his PhD, he studied various issues related to scheduling of embedded multiprocessing systems. One of his papers concerning scheduling was awarded with the best paper award at the IEEE SIPS conference in 2007. Jani Boutellier has contributed to 2 peer-refereed journal articles and 15 peer-reviewed conference articles.

Andreas Dahlin completed his MSc degree in 2007 and he is expected to defend his PhD thesis and graduate in mid 2011 from the department of information technologies at Åbo Akademi University. Therefore he would work full time as a post-doc in this project. Andreas Dahlin is working on data-flow based description and is exploring language specific descriptions for schedulers and run-time system support. Andreas Dahlin is working from the beginning on the development of the Canals language and has contributed to the Rialto 2.0 language for heterogeneous computations.

Pekka Jääskeläinen has worked in the TCE project since its inception in 2002. He wrote his master's thesis of the instruction set simulator for the toolset in 2005 and has since worked mostly on the compiler research around the platform. He has published several conference and journal papers of parts the toolset, including the simulator and the compiler, and of TTA-processor based algorithm implementations. Currently he is doing preliminary work on extending TCE to support manycore designs. His planned Dr.Tech. graduation is in 2011.

Carlos Sánchez de La Lama graduated as MEng in Telecommunications Engineering from University of Cantabria (Spain). The subject for his thesis was a microcontroller design for FPGA implementation. After graduating he worked in industry for 4 years, getting practical experience in hardware design and low level programming. His interest in TTA led him to start his doctoral studies in 2007 with URJC and get in contact with the TCE group of TUT at the beginning of 2008, starting the URJC-TUT collaboration. The projected PhD graduation time is mid 2011, thus he would also work as a post-doc in this project.

In addition to the postdoc researchers described above, each university will assign 1 PhD student to work on DORADO.

6.2 National and international cooperation

Our consortium is going to organize research visits to foreign universities and respectively invite researchers of these universities to visit their Finnish partners. The collaborating research institutes confirmed at the time of writing this application are the University of Maryland (U.S.), INSA Rennes (France) and EPFL (Switzerland).

University of Maryland has been a long-term partner of OU, AAU and TUT. In 2006, researchers Perttu Salmela (Jarmo Takala’s group) and Jani Boutellier (Olli Silvén’s group) stayed in Shuvra Bhattacharyya’s group for a while. In 2007, Vladimir Guzma (TUT) and Sébastien Lafond (AAU) conducted research in Maryland as well. The cooperation has resulted in a number of research papers that have contributed to several PhD theses. During this project, this fruitful cooperation is intended to be continued.

Research visits have also been conducted between the group of Dr. Mickaël Raulet at INSA Rennes and AAU. Andreas Dahlin (AAU) visited the INSA Rennes, and Jérôme Gorin (INSA Rennes) visited Åbo Akademi University for several weeks in 2010. As the role of INSA in the development of the CAL dataflow language compiler is very central, it is extremely important to strengthen the cooperation with INSA Rennes. A 5 months visit to INSA Rennes has been planned, and it is expected that a researcher from INSA Rennes will stay 5 months in Prof. Johan Lilius’s group during the project.

The research group of Dr. Marco Mattavelli at the EPFL has a central role in the development of the Reconfigurable Video Coding (RVC) Standard, which is at the writing of this the most important application that uses the CAL language. Jani Boutellier from OU acted as a visiting researcher at the EPFL during 2007-2008. Since then, several research papers have been written in collaboration with this group. During DORADO, Jani Boutellier has been invited for a research visit of 3 months to Dr. Mattavelli’s group. The purpose of this visit is to study the requirements of Reconfigurable Video Coding regarding the DORADO toolchain, and thus enable the toolchain to create efficient instantiations of RVC decoders. A 2 months visit is also planned from AAU to the EPFL to work on the model-checking of CAL specifications. This visit will assess how the approach fits into the overall CAL tool workflow.

Finally, the collaboration with URJC is intended to be continued. During the PARSEC project the collaboration was very active. During 2009-2010, total of two visits from TUT to URJC and three visits from URJC to TUT were made. Researcher Carlos Sánchez de La Lama will be collaborating in the continuum project either as an employee of TUT (thus relocating to Tampere) or with mutual research visits on each year of the project. This is to be decided at the beginning of the project.

The research groups have an established way of working together that has been ironed out in the NECST, ECUUS and PARSEC projects. This includes common mailing lists, repositories for information and regular informal meetings to discuss specific issues that arise in the project. We expect these working habits to be carried over to this project.

6.3 Miscellaneous issues

The consortium leader and the sub-project leaders are permanent employees of universities and thus do not require funding from the project budget.

The research environments of the three groups belonging to the consortium are normal Finnish university environments that offer the office and research equipment (*i.e.* computers, software), as well as access to scientific publication databases.

7. Researcher training and research career

DORADO has been planned to be carried out by 4 post-doctoral researchers and 3 post-graduate students. The post-graduate students are expected to work towards their PhD degrees during the progress of the project. The project publications are planned to form a basis for their theses. While selecting suitable post-graduate students, gender equality will be promoted by giving priority to female applicants who have a suitable background.

The research careers of the post-doctoral researchers will be promoted by aiming for publishing the research results in high-quality journals and conferences. The international research visits of the post-doctoral researchers promote their careers as well.

8. Expected research results

It is expected that the project develops high impact techniques for designing and programming performance portable heterogenous systems. For example, the research aiming to reduce the memory bottlenecks of manycore systems is expected to gain global interest as it is one of the hot topics world-wide.

The source code of the toolchain developed in the project will be continued to be licensed with the liberal MIT license, thus enabling free academic and industrial exploitation of the research results without copyright issues. The open source model is suitable for academic world as it allows building on the contributions of the others.

The research results of all work packages are planned to be published in international journals and conferences. Should a new breakthrough take place, it can be expected that the results would be announced in the public media.

The DORADO project will maintain a public website, where the progress of the project is visible, and publications and source code can be downloaded.

References

[1] E. A. Lee, "The problem with threads," *IEEE Computer*, vol. 29, no. 5, pp. 33-42, May 2006.
[2] H. Sutter, "The Free Lunch Is Over: A Fundamental Turn Toward Concurrency in Software", *Dr. Dobb's Journal*, 30(3), March 2005.
[3] K. Asanovic, R. Bodik, B. C. Catanzaro, J. J. Gebis, P. Husbands, K. Keutzer, D. A. Patterson, W. L. Plishker, J. Shalf, S. W. Williams and K. A. Yelick, "The Landscape of Parallel Computing Research: A View from Berkeley", EECS Department, University of California, Berkeley, Technical Report No. UCB/EECS-2006-183, December 18, 2006
[4] E. A. Lee E and D. Messerschmitt, "Synchronous data flow", Proceedings of the IEEE 75(9): 1235–1245, 1987.
[5] B. Bhattacharya and S. S. Bhattacharyya, "Parameterized dataflow modeling for DSP systems", IEEE Transactions on Signal Processing 49(10): 2408–2421, 2001.
[6] M. Mattavelli, I. Amer and M. Raulet, "The Reconfigurable Video Coding Standard", IEEE Signal Processing Magazine, vol. 27, no.3, pp.159-167, 2010.

[7] A. Dahlin, J. Ersfolk, G. Yang, H. Habli and J. Lilius, "The Canals Language and its Compiler", SCOPES '09: Proceedings of th 12th International Workshop on Software and Compilers for Embedded Systems, Apr 2009.
[8] J. E. Stone, D. Gohara and S. Guochun, "OpenCL: A Parallel Programming Standard for Heterogeneous Computing Systems", Computing in Science & Engineering, vol. 12, no. 3, pp. 66-73, 2010.
[9] F. Balarin, Y. Watanabe, H. Hsieh, L. Lavagno, C. Passerone and A. Sangiovanni-Vincentelli "Metropolis: an integrated electronic system design environment", Computer, Volume: 36, Issue: 4, Pages: 45 - 52, 2003
[10] J. Eker, J. W. Janneck, E. A. Lee, J. Liu, X. Liu, J. Ludvig, S. Neuendorffer, S. Sachs, Y. Xiong, "Taming heterogeneity - the Ptolemy approach," Proceedings of the IEEE, vol. 91, no.1, pp. 127- 144, Jan 2003
[11] S. Ha, S. Kim, C. Lee, Y. Yi, S. Kwon, Y.-P. Joo, "PeaCE: A hardware-software codesign environment for multimedia embedded systems", ACM Trans. Des. Autom. Electron. Syst., vol. 12, no. 3, pages 1-25, ACM, New York, USA, 2007
[12] J. Lilius, A. Dahlin and L. Morel, "Rialto 2.0: A Language for Heterogeneous Computations" Proceedings of Distributed, Parallel and Biologically Inspired Systems, IFIP Advances in Information and Communication Technology, Volume 329/2010, 7-18, 2010
[13] http://spinroot.com/
[14] R. Kumar, D. M. Tullsen, N. P. Jouppi, P. Ranganathan, "Heterogeneous chip multiprocessors," Computer , vol.38, no.11, pp. 32- 38, Nov. 2005
[15] P. Jääskeläinen, P. Kellomäki, J. Takala, H. Kultala, M. Lepistö, "Reducing Context Switch Overhead with Compiler-Assisted Threading", The 3rd International Workshop on Embedded Software Optimization, Shanghai, China, Dec, 2008
[16] S. Steinke, L. Wehmeyer, B.-S. Lee, P. Marwedel, "Assigning program and data objects to scratchpad for energy reduction," Proceedings of the Design, Automation and Test in Europe Conference and Exhibition, pp.409-415, 2002