HiPEC: High Performance Embedded Computing

Project leader: Krzysztof Kuchcinski, LTH
Sub-project leaders: Dake Liu (LiU), Verónica Gaspes (HH), and Viktor Öwall, (LTH)

Abstract

Parallelism is the main way to provide significant performance improvement of embedded systems while keeping energy consumption low. Streaming applications are good candidates for parallelization since they are regular and exhibit data parallelism. Traditionally, ASICs have been designed to implement specific functionality with high performance and low power constraints. Recently, coarse-grained reconfigurable array architectures have been proposed as flexible but still high performance alternatives. It is therefore expected that the DSP computing system, increasingly parallel and reconfigurable, will be one of the dominating parts in OEM equipments in 2020 because it maximally exposes opportunities of parallelization. In this project, we address reconfigurable array processor architectures as well as software tools for their programming. A massively parallel execution platform with powerful computing nodes and hierarchical interconnection structure suitable for streaming applications will be developed and studied. The distinct features of our software development approach are the use of the CAL language for programming of these architectures as well as the development and use of tools for timing and energy analysis at early design stages. Combining both hardware and software experts in the same project provides a strong basis for covering the whole spectrum of this new technology.

1 Beyond the von Neumann model

Up until recently, the evolution of computing machines was characterized by an unabated increase in performance. This progress was in large part due to steady advances in silicon manufacturing technology, which provided cheaper, smaller, and faster circuits, and, to some degree due to improvements in processor architecture that exploited those advances to create ever faster processors. This development, however, has considerably slowed down in the last few years. As a result of these developments, the computational power of individual processors is no longer increasing, and consequently the only way to significantly improve the performance of a computing machine is to use more processors operating at the same time: the age of parallelism has finally arrived.

Along with von Neumann processor development, application specific integrated circuits (ASICs) have been adopted for telecom and multimedia (streaming) applications in order to provide higher performance and much lower power consumption. ASIC designs are, by nature, highly parallel but also fixed, lacking the necessary flexibility offered by processors through programming. The increasing demands for performance and complex functionality gave rise to architectures combining the flexibility and ease of use of general purpose processors with the high degree of parallelism of ASICs. These developments opened a new area of course-grain reconfigurable computational platforms, sometime also referred to as computational fabrics. Such platforms are massively parallel architectures, usually containing a regular array of processing elements, working mostly independently on local memories, but communicating with each other through special interconnect structures.

The model of the sequential instruction set computer has been a very powerful abstraction that brought enormous benefit to the computing community. This is in stark contrast with the situation in the world of parallel machines, where no such nearly-universal machine model exists. There is no common machine model tying these platforms together, and consequently there are no common tools, no common languages, and no common code in the form of applications or libraries. In this project, we propose to narrow this gap and work on both massively parallel hardware platform architecture and tools for software development.

2 Purpose and aims

Finding a universal parallel platform and programming model for massively parallel computing machines is an attractive long term goal that is too ambitious for the scope of just this one project. Nevertheless we intend to bring this closer to reality by taking several steps toward this goal. We limit our focus on stream/dataflow computing and related parallel computing platforms and programming models. In our view, only a synergy between the hardware platform and the software paradigm can truly bring forth the benefits of parallelism that computing strives after today.

The stream-based (or dataflow) programming model for parallel computing, which is a natural abstraction for many of the application domains that drive the need for higher computing performance, such as signal and media processing, networking and packet processing, coding and cryptography. In this model, individual actors interact with each other exclusively through order-preserving, lossless, directed FIFO connections, over which they exchange packets of data (tokens). Conceptualizing a program as a network of actors provides a natural degree of explicit parallelism in the form of concurrently operating and asynchronously communicating actors. Such programs can be mapped onto computing machines with different number of execution units. It is possible to use different trade-offs between parallelism, performance and cost. The stream-based programming model will *decouple the algorithm implementation from the executing hardware*, enabling a level of portability currently enjoyed only by software targeting traditional processors.

Computational platforms for telecom and multimedia applications evolve driven by standards and technologies becoming increasingly complex. High performance data processing as well as low power consumption are the major requirements for designing systems in the future. Additionally, the future system platforms will also need to cope with various standards, and even to support multiple tasks simultaneously in areas which today are single threaded, e.g. concurrent radio baseband processing for multiple standards. Using traditional hardware accelerators dedicated to each of the desired operations is no longer a viable solution, due to their inflexibility and high non-recurring engineering (NRE) cost. Therefore, system architectures that can be *dynamically reconfigured to adapt to the current processing conditions* is a promising solution. The reconfigurable architectures enable resource sharing on a task level, which provides an efficient way to utilize the valuable hardware resources between different applications.

In the first generation of multicore processors, communication is commonly carried out through shared memory, a solution that is still heavily influenced by the von Neumann model and which does not scale well. Instead, we will develop and use architectures with distributed memory, each node having its own local memory used for processing and communication with the neighboring nodes. Global communication is achieved using a global interconnection network. One of the important topics of this project will be the investigation of different hierarchical interconnection structures for both global and local communication.

Massively parallel architectures introduce new challenges for the software development tools. Programs need to be partitioned and mapped into processing nodes, taking into account local/global data accesses. The partitioned functionality must be properly scheduled to support any necessary local/global communication. Data has to be properly partitioned and mapped to local memories of the processing nodes, relying on data lifetime analysis tools in order to fit all to live data in the available memory. Therefore, to favor a widespread use of massively parallel architectures, these must be supported by a good development environment. We believe the development environment must (a) generate efficient execution from a high level implementation language, for example a stream based language, (b) provide a short turn around time, from code writing to deployment.

Massively parallel architectures have the potential to provide tremendous amount of processing power, based on the combined performance of all the individual processing nodes. These nodes can have different architectures, from non-programmable processing units to von Neumann architecture with specific instruction sets, to even a more complex computing structure such as an array of processing elements. The hierarchical and heterogeneous organization of the platform provides the potential of reusing already developed efficient computing structures, such as systolic arrays, SIMD arrays and even complete existing SoCs as nodes. The processing nodes are connected by a network that needs to support both local and global communication needs. Since energy and power consumption will also be of major concern, each processing node must work as efficiently as possible. We will work on developing efficient processing nodes as well as interconnection network. To meet the computational energy demands and still provide the flexibility needed, we will focus on programmable and reconfigurable processing nodes. We will work on minimizing memory cost and memory accesses for instructions, configuration and data. This will provide the energy and power efficiency needed by tomorrow's battery powered devices.

To summarize, the goals of this project are as follows.

- Provide a massively parallel execution platform meeting the demands of tomorrow's applications. The key is the hierarchical organization of computing nodes and their interconnections , comprising local links and global networks, as well as memory architecture providing necessary bandwidth for streaming applications.
- Provide energy efficient and computationally powerful computing nodes for the architecture outlined above. The aim is to provide components needed by tomorrow's handheld devices. This is achieved by

utilizing low overhead reconfiguration and efficient memory accesses.

- Provide a software development environment for such systems. We are primarily focusing on efficient execution of streaming applications written in CAL language. The goal is to automatically partition, map and schedule these applications on our execution platforms.

The project covers a broad area spanning from hardware design and computer architecture to application mapping and design optimisation methods. Moreover, it is concerned with problems that go across several domains, such as energy consumption or execution performance. Because of this, we have combined four research groups into a single project. The groups have different focus but together cover the area of hardware and software design and optimization. The Circuits and Systems group from Dept. of Electrical and Information Technology, LTH (EIT/LTH), led by Doc. Viktor Öwall has competence in hardware design, array processor architectures as well as interconnection networks. The division of Computer Engineering, Dept. of Electrical Engineering, LiU (LiU), led by Prof. Dake Liu, is well know from its work on application specific processors. The Centre for Research on Embedded Systems (CERES), Högskolan i Halmstad (HH), led by Prof. Bertil Svensson has good experience in embedded high-performance parallel computing for industrial applications. For this project, the competence in Network-on-Chip and in the development of domain specific languages and language processing tools (Dr. Verónica Gaspes) is of particular importance. Finally, the Embedded System Design Lab from Dept. of Computer Science, LTH (ESD/LTH), led by Prof. Krzysztof Kuchcinski, is recognized for its work on design automation and system optimisation. These groups together provide a good match for addressing problems proposed in this application.

3 Survey of the field

In order to meet the increased computational demands of, e.g., multimedia applications, such as video processing in HDTV, and communication applications, such as baseband processing in telecommunication systems, the architectures of reconfigurable devices have evolved to coarse-grained compositions of functional units or program controlled processors operated in coordination to improve performance and energy efficiency [26]. In this project, we will mainly consider array architectures comprising either programmable processors or functional units.

Various computational models can be used to express computations carried out in reconfigurable processor arrays. Khan process networks are historically the first dataflow-based model, where processes communicate over channels to form a network. In this project, we will use the CAL dataflow language, which further extends the dataflow computational model. CAL has a compiler to C language and a compilation and synthesis tools for FPGA. A big challenge remains to effectively compile and map CAL to reconfigurable array of processors architectures. Streaming languages, such as StreamC/KernelC, are another example of computational models applicable for streaming applications.

Many of today's highly parallel computing platforms come with their own programming environment (frequently provided by the manufacturer of the platform), and they introduce their own specialized machine model: for instance, Tilera's TILE64 requires C with a specialized library ("iLib"), Nethra's Ambric architecture uses two specialized languages, "astruct" and "ajava", PicoChip's machines are being programmed in a mixture of C and VHDL, Intellasys' SEAForth multicore machines need "VentureForth". There is no common machine model tying these platforms together and, consequently, there are no common tools, no common languages, and no common code in the form of applications or libraries. Our effort tries to close the gap between application hardware and programming by considering, in one project, both reconfigurable flexible computational platform and its programming.

Mapping and scheduling to array of processing elements has its roots in compilation to systolic arrays. The majority of research considers usually mapping of imperative languages, such as C-like languages. Such work concentrates on loops and tries to analyze and map them into regular architectures. Various kinds of dataflow graphs are used as intermediate representations (IRs) for resource assignment and scheduling but many usually do not explicitly consider the regularity of an architecture. CAL has its own internal representation (XLIM). Compiler based representation can also be used. Low Level Virtual Machine (LLVM) compiler infrastructure [2], for example, provides a state of the art infrastructure for mid-level language-independent analyses and optimizations. It is currently used as intermediate representation for most programming languages [17] and it is being studied as infrastructure for parallel platforms [18, 24].

Energy efficiency of new architectures and applications is an important characteristic that needs to be considered. For many situations it is no longer enough to be energy efficient during worst case loads, but also during periods of light or no load. That is, there is a strong need to address energy scalability, where the energy consumption follows the workload of the system. In today's processors the most pervasive method to achieve energy scalability is through the usage of dynamic frequency and voltage scaling (DVS) [7, 8].

Another approach is to use of multiple clocks for multiple cores [13]. One ambitious project addresses the energy-scalable and power-aware embedded system design is the PARTS project [23]. They however, do not consider the additional problems and possibilities with many-core systems, similar to our architectures.

The power consumption in reconfigurable processor array platforms is application dependent, but the contribution of interconnection network is significant in most cases. A Network-on-Chip (NoC) is an approach to efficiently interconnect the components of System-on-a-Chip (SoC) [9, 15, 16, 22]. Many factors determine the dynamic energy consumption in NoC for an application, including the total number of bits communicated, distance (or number of hops) for packets in NoC and the waiting times for packets in router buffers [14]. Therefore, efficient, adaptive and application specific routing algorithms are generally useful for low power NoC design.

4 Project description

This project focuses on hardware and software development of streaming applications. We address both the design of a massively parallel reconfigurable computational platform and a software development environment comprising mapping and scheduling tools.

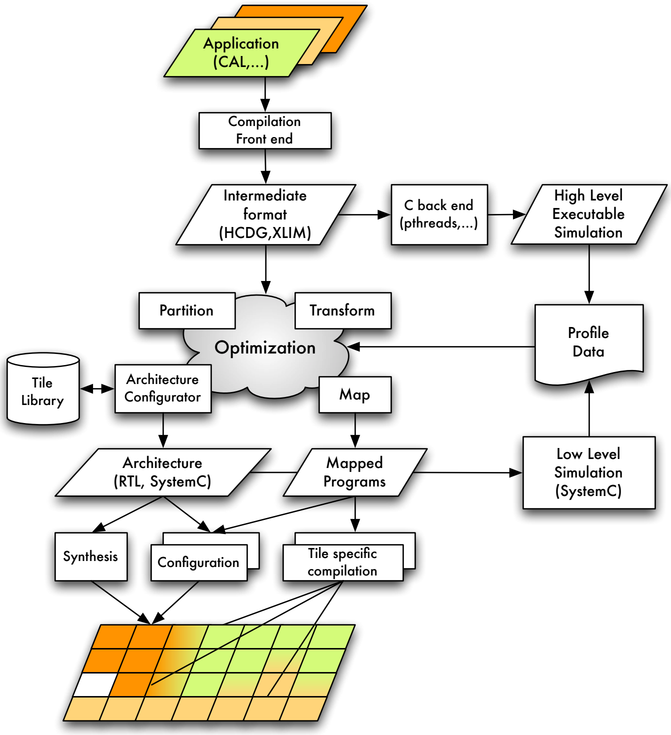
Application (CAL,...)
Compilation Front end
Intermediate format (HCDG,XLIM)
C back end (pthreads,...)
High Level Executable Simulation
Partition
Transform
Optimization
Profile Data
Tile Library
Architecture Configurator
Map
Architecture (RTL, SystemC)
Mapped Programs
Low Level Simulation (SystemC)
Synthesis
Configuration
Tile specific compilation

Figure 1: Proposed design flow for mapping of CAL applications into reconfigurable platforms.

An overview of the design flow adopted in the project is depicted in Figure 1. The process starts with a set of application specifications and produces both a specific parallel architecture, reconfigurable for optimal performance, and specific software for each of the architecture components, all finely tuned for the target applications. We selected to use CAL dataflow programming language as an input language since it is a highly parallel language that does not assume specific implementation choices. The steps required by this design flow include transformation of the initial specification to an intermediate format (for interchange and optimization purposes), architecture configuration (selecting the tiles – appropriate processing elements and interconnect of the regular structure), partitioning and mapping (assigning the right functionality to the available tiles), and finally synthesizing the architecture, along with efficient configurations and software compiled for each processing element. High level and low level simulations are employed to obtain profiling information and evaluate and guide the architectural and algorithmic choices.

The project is divided into two main sub-projects with strong coupling between them. The first sub-project addresses architecture of our reconfigurable platform that includes network and processing elements. The work on interconnection networks will be carried out, in first place by the EIT/LTH group with additional studies regarding energy consumption minimization carried out by the HH group. Processing elements, possibly reconfigurable, will be studied and designed, in first place, by the LiU group with additional work carried out by the EIT/LTH group. In the second sub-project, the work on partitioning and mapping will be carried out by the ESD/LTH group. They will do analysis of CAL programs and their partitioning, mapping and scheduling with focus on addressing memory issues and optimization methods based on constraint programming. Again, the HH group will work on energy efficiency and energy scalability, but now from the software point of view.

4.1 Massively Parallel Computational Platform

Our massively parallel reconfigurable computational platform is an array of possibly hierarchical and heterogeneous processing nodes and memories connected through an hierarchical interconnection network [12].

The project will concentrate on two main components of this architecture, *processing elements* and *hierarchical interconnection network*, both striving for flexibility through reconfigurability and/or programmability.

The work on interconnection structure will focus on the development of hierarchical network interconnections and dynamical configurations for the heterogeneous reconfigurable cell array. This part of the project will be carried out through the following phases. First, we will investigate different hierarchical network structures and network interconnections in a heterogeneous cell array. We will then develop and benchmark network interconnections for a heterogeneous cell array in respect to different design aspects, e.g. communication latency, transmission throughput, hierarchical network topology, communication protocol in a globally asynchronous locally synchronous (GALS) network, network routing overhead, area cost of routers, and power consumption. Finally, we will investigate and elaborate on dynamical configurations for a heterogeneous cell array with regard to, for example, reconfiguration time, context switching between multi-standard applications, run-time resource allocations, and encoding for dynamic configurations.

One of the major tasks of this project is to select a set of suitable base architectures and instruction set for the processing elements of our reconfigurable platform. Because the memory bandwidth bottleneck is one of the essential problems of parallel computing, we see the need for special memory architectures, such as conflict free multibank memory with low access overhead.

The architecture development will be carried out based on selected applications and inputs from all partners, in order to ensure that the reconfigurable platform can be reasonably targeted by the toolchain developed in the project. Another important task is to propose domain specific extensions to the architecture platform targeting areas such as baseband processing and multi-media processing blocks. The first specification of the architecture will be ready within 12 months (see project plan). However, we will continuously be working on the architecture by cooperating between the hardware development teams and software teams.

Another major task is to create an extendable simulation model of the reconfigurable platform to be used either for architectural experiments or as a development environment for *configware*. The development of *configware* for the platform is also a major task. The most important application domain is baseband processing for multi-standard multi-stream applications including, e.g., forward error correction (FEC), entropy coding, MIMO processing and channel estimation. As an example, a FEC module could include combined Viterbi, Turbo, Reed-Solomon and LDPC, MIMO. The groups will work together for function mapping and scheduling for selected algorithms. A manually mapped application will also serve as a reference point when evaluating the efficiency of our tools. In the second phase of the project the target algorithms will be mapped onto the reconfigurable platform.

In addition we will investigate the energy scalability concept in relation to network-on-chip (NoC) communication within multi-core systems. It will be useful to include the property of scalability in the platform architecture as well as software tools. At low computational load it should be possible to use only a small part of the platform to run applications. We will investigate how well established scaling concepts like dynamic voltage scaling (DVS) can be applied to energy-scalable NoC and furthermore develop NoC concepts that support energy aware mapping of dataflow intermediate representation onto multi-core architectures.

4.2 Software Development Tools

In this project, we will study methods for mapping dataflow descriptions into reconfigurable architectures consisting of interconnected programmable processors and local memories. We will concentrate on CAL dataflow language for specifying applications, but if this proves to be a limitation, we will consider using other similar languages. Furthermore, we will develop tools that optimize different objectives, such as performance, area and power/energy consumption. Our approach will localize communication between different CAL actors by using various partitioning strategies. Partitioning combined with mapping and scheduling will provide different application implementation strategies. Our approach will be evaluated on both simulators of various architectures as well as prototype architectures developed in the project.

Massively parallel architectures have the potential to deliver the performance required by tomorrow's applications. One problem that must be solved to achieve this is the memory size and bandwidth limitation since the local memories embedded in the architecture are of limited size. We will address this problem by aligning data producers and consumers. The goal is to generate efficient access patterns and minimize the amount of live data by employing data dependence analysis. We will look at multiple iteration spaces at the same time, in contrast to earlier work which has been focusing on one loop at a time.

Another important research topic is how to improve the energy efficiency of a design. Designers of embedded systems today need to be able to assess and optimize the energy consumption of the code and algorithms they design. Important is also that the system achieves high efficiency not only for high throughput situations, but also in lightly loaded ones. To achieve this, both the hardware and the software need to be energy scalable, adapting their energy consumption to the variations in performance demands.

We will focus our effort on the back-end environment and intend to address energy efficiency in two ways. First, at design time, our tools performing partitioning, mapping and scheduling will have energy consumption as part of the optimization goal. We will devise analysis techniques especially adapted to our platform, dedicated to estimating the energy consumption used to drive our optimization. For this, we plan to use the low level virtual machine (LLVM) infrastructure. Second, at runtime, we will examine methods and techniques for managing the energy budget online, adapting to runtime performance demands.

4.3 The Project Team

The graduated researchers involved in the project are listed below. Some of them will be financed from other sources (indicated by "-" in the table), for example ELLIIT center. Dr. Jörn W. Janneck is one of the authors of CAL language and is a valuable source of knowledge regarding CAL. We also have contacts with Johan Eker at Ericsson/Lund that can provide us with CAL benchmarks relevant for industry and our project.

Name	% of full time[a]	Group	Speciality
Prof. Krzysztof Kuchcinski	20%	ESD/LTH	Design optimization
Dr. Per Andersson	20%	ESD/LTH	Mapping and scheduling
Dr. Flavius Gruian	10%	ESD/LTH	Energy scalability, Scheduling
Dr. Jörn Janneck	10%	ESD/LTH	CAL language, partitioning and mapping
Prof. Dake Liu	10%	LiU	CPU design, DSP
Dr. Andreas Ehliar	20%	LiU	Reconfigurable computing components and reconfigurable SoC
Doc. Viktor Öwall	10%	EIT/LTH	ASIC design, DSP
Dr. Verónica Gaspes	10%	HH	Models of computation, Domain specific languages
Dr. Tomas Nordström	10%	HH	Energy scalability
Dr. Jerker Bengtsson	20%	HH	Modeling and application development tools

[a] the numbers represent involvement in the project; the financing is partially provided by by HiPEC SSF project and partially by other sources.

5 Strategic relevance

It has been clear for some time that the development of parallel architectures will be the dominating way to increase performance in embedded systems. The rapid development of the requirements is illustrated by the step from third to fourth generation (4G) wireless technology, which represents an increase in computational requirements from tens of Gops (Giga operations per second) to thousands of Gops, with a power budget of approximately 1W for all the computations in the mobile terminals [3, 27]. This is only for the wireless protocol; other forms of embedded signal processing, such as high-definition video, show similar increases compared to current standards. Within other application areas there are plans for similar development; one example is car radar systems with phased array antennas, requiring hundreds of Gops in a safety critical setting.

Parallelism is important not only in the most performance demanding applications, but also in applications where extremely low power consumption is of prime concern. Since clever use of parallelism is a key to power-efficiency, the developed architectures and programming techniques are enabling technologies for many embedded applications related to, e.g., health care and mobile services. Our reconfigurable platform will be able to handle important parts of these application domains that cannot be handled efficiently by a general programmable processor or DSP processor.

6 Significance

The proposed project is important for the area of application specific hardware, both for processing and interconnect structures, parallel programming and compilation, and mapping of parallel programs. We propose to co-design and co-optimize a reconfigurable and flexible hierarchical computing platform and its programming environment. Moreover, we believe that some analysis and compilation methods develop in this project can also be used for other architectures.

New parallel reconfigurable architectures for embedded systems and in particular their efficient network interconnections have significant impact on the field of embedded systems. Data transmissions between network nodes improve not only network performance per se, but also the interactions between different processing elements. Furthermore, efficient on-chip networks may also help to reduce physical interconnections, to ease data routing, and to reduce hardware overhead in reconfigurable architectures.

Dynamic re-configuration is an important system feature, essential for task-level resource sharing between multi-standard applications.

7 Preliminary results

During the recent years the ESD/LTH group has been working in related areas and some of their previous results are relevant for this project. These include a finite domain solver with graph constraints used for different problems in optimization of processors and processor arrays [28–30]. The ESD/LTH group has also competence in high-level synthesis (HLS) that can be used in this project. In our previous work, we have extended HLS memory considerations [4] and presented a technique for automatic generation of a memory architecture, data paths and associated controllers from a high level language such as C. The technique is based on duplication of data in local memory. High memory efficiency is achieved by rearranging the data memory layout when data migrates to the local memory. Finally, ESD/LTH has also experience in the area of energy efficient scheduling on dynamic voltage processors, having proposed some of the first approaches in this area [10, 11].

The group at HH has proposed and developed a many-core intermediate representation (IR) suitable for implementation of compilers for domain-specific languages for the DSP domain [5], one example of which is CAL. In the IR, communication and memory resources, as well as the program mapped on those resources, are abstracted by means of a hierarchical heterogeneous dataflow model of computation. In particular, the execution of the IR provides means for dynamic analysis of non-functional properties of programs mapped onto many-cores. So far, the IR, in the form of a timed configuration graph, has been shown to be useful for providing compiler-generated feed-back of the timing properties of programs, when the represented program is abstractly executed on a specific many-core target [6]. In this project we will extend this work to include energy aspects, such as awareness, efficiency, and scalability.

In [12] it is shown that a hierarchical organization of routing in large and complex multi-core systems results in improvement in communication performance and lower power consumption. It also point out that hierarchical organization provides the possibility of raising the level of reuse from cores to subsystems in multi-core platform design.

A pre-study conducted by LiU indicates that FEC for a typical wireless communication system can be handled with a reconfigurable platform consisting of less than 100 functional units. In the SSF project “DSP Platform for emerging telecommunication and multimedia”, LiU has developed a parallel architecture with reconfigurable micro code datapath and reconfigurable data access path for conflict free memory bank access which we expect to be a key technology of the reconfigurable platform [21].

System infrastructure of dynamically reconfigurable cell array has been developed at EIT/LTH [19, 20, 25]. It includes various functional elements, processing, memory, and network routing cells. The functional elements are interconnected in a hybrid on-chip network. The local mesh network enables low latency high throughput data transmission, whereas a tree-structured global interconnection provides communication flexibility and access to external devices. So far, applications have been pursued within the area of digital baseband processing. EIT/LTH architectures have been developed for reconfigurable FFT processing and for time synchronization in multi-standard OFDM systems. In the latter case, the system, consists of a 2x2 reconfigurable cell array, supports three different wireless radio standards, WLAN 802.11n, LTE, DVB-H, and is capable of processing two concurrent data streams from any three of supported radio standards.

8 Project Plan

The project is planned for four years with the following four stages.

Year 1: Specification of the execution platform and software tools. The goal of this phase is to create a specification of an execution platform and basic software tools for application partitioning, mapping and scheduling. A proposal of a preliminary architecture will be ready within 6 months to allow for feedback from the partners, and a finalized version of the architecture will be ready within 12 months, including the ESL behavior model and RTL model. Methods for software tools will be studied and simple prototype tools will be developed.

Year 2-3: Method exploration. In this phase we will implement an architectural platform with different processing elements and further explore application requirements for our platform. We will work on optimisation methods for our software tools to better explore partitioning, mapping and scheduling as well as code generation for processing units. We will also develop a simulator for the execution platform for the purpose of functional and timing verification of applications, design space exploration and for evaluating efficiency of other software tools.

Year 4: Experimentation and refinement In this phase we will carry out experiments to map different applications into our platform. The goal is to do final refinement of the architecture and tools based on the feedback from experiments. Existing high performance and low power computing structures will be evaluated for their suitability as nodes in our platform. Suitable interconnection architecture will be developed for integrating them in the platform.

Year 5: Final experiments and dissemination. We will further explore other application domains to broaden the platform. In this phase we will disseminate the results.

We apply for financing of eight PhD students and some senior researchers in each group. The cost of supervision and participation of other senior researchers will be mainly financed separately by ELLIIT [1].

References

[1] ELLIIT, Excellence Center at Linköping - Lund in Information Technology, `http://elliit.liu.se/`.

[2] The LLVM compiler infrastructure, `http://www.llvm.org/`.

[3] Recommendation ITU-R M.1645: Framework and overall objectives of the future development of IMT-2000 and systems beyond IMT-2000. `http://www.ieee802.org/secmail/pdf00204.pdf` (checked April 12, 2010).

[4] P. Andersson and K. Kuchcinski. Automatic local memory architecture generation for data reuse in custom data paths. In *International Conference on Engineering of Reconfigurable Systems and Algorithms*, June 21–24 2004.

[5] J. Bengtsson. *Models and Methods for Development of DSP applications on Manycore Processors*. PhD thesis, Chalmers University of Technology, 2009. Ph. D. Thesis, Technical Report No 62D.

[6] J. Bengtsson and B. Svensson. Manycore performance analysis using timed configuration graphs. In *Proc. of IEEE Intl. Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS IX)*, Samos, Greece, July, 20-23 2009.

[7] L. Benini, A. Bogliolo, and G. De Micheli. A survey of design techniques for system-level dynamic power management. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 8(3):299 –316, jun 2000.

[8] L. Benini and G. de Micheli. System-level power optimization: techniques and tools. *ACM Trans. Des. Autom. Electron. Syst.*, 5(2):115–192, 2000.

[9] M. F. Chang, J. Cong, A. Kaplan, M. Naik, G. Reinman, E. Socher, and S.-W. Tam. CMP network-on-chip overlaid with multi-band RF-interconnect. In *Proc. of IEEE 14th Intl. Symposium on High Performance Computer Architecture*, pages 191–202, 2008.

[10] F. Gruian. Hard real-time scheduling for low-energy using stochastic data and dvs processors. In *Proceedings of the 2001 International Symposium on Low Power Electronics and Design*, pages 46–51, August 6–7 2001.

[11] F. Gruian and K. Kuchcinski. LEneS: Task-scheduling for low-energy systems using variable voltage processors. In *Proceedings of the 2001 Asia South Pacific – Design Automation Conference*, pages 449–455, January 30 – February 2 2001.

[12] R. Holsmark, S. Kumar, M. Palesi, and A. Mejia. HiRA: A methodology for deadlock free routing in hierarchical networks. In *Proc. International Symposium on Networks on Chip (NOCS)*, 2009.

[13] A. Iyer and D. Marculescu. Power efficiency of voltage scaling in multiple clock, multiple voltage cores. In *ICCAD '02: Proceedings of the 2002 IEEE/ACM international conference on Computer-aided design*, pages 379–386, New York, NY, USA, 2002. ACM.

[14] J. Kim, D. Park, T. Theocharides, N. Vijaykrishnan, and C.R. Das. A low latency router supporting adaptivity for on-chip interconnects. In *Proceedings of 42nd Design Automation Conference*, pages 559 – 564, june 2005.

[15] S. Kumar, A. Jantsch, J.-P. Soininen, M. Forsell, M. Millberg, J. Oberg, K. Tiensyrja, and A. Hemani. A network on chip architecture and design methodology. In *Proceedings IEEE Computer Society Annual Symposium on VLSI*, pages 105 –112, 2002.

[16] S. Kundu and S. Chattopadhyay. Interfacing cores and routers in network-on-chip using GALS. In *Proceedings of International Symposium on Integrated Circuits*, pages 154–157, 2007.

[17] Ch. Lattner. LLVM and Clang: Next generation compiler technology. In *The BSD Conference*, Ottawa, Canada, May 2008.

[18] S. J. Lee, D. K. Raila, and V. V. Kindratenko. LLVM-CHiMPS: Compilation environment for FPGAs using LLVM compiler infrastructure and CHiMPS computational model. In *Reconfigurable Systems Summer Institute (RSSI'08)*, Champaign, USA, July 2008.

[19] T. Lenart. *Design of Reconfigurable Hardware Architectures for Real-time Applications: Modeling and Implementation*. PhD thesis, Lund University, 2008.

[20] T. Lenart, M. Gustafsson, and V. Öwall. A hardware acceleration platform for digital holographic imaging. *J. Signal Process. Syst.*, 52(3):297–311, 2008.

[21] D. Liu, J. Sohl, and J. Wang. Parallel computing and its architecture based on data access separated kernels. *ISI IJERTCS, International Journals Embedded and Real-Time Communication systems*, Jan–March 2010.

[22] M. Palesi, R. Holsmark, and S. Kumar. A methodology for design of application specific deadlock-free routing algorithms for NoC systems. In *Proceedings of the 4th International Conference Hardware/Software Codesign and System Synthesis, 2006. CODES+ISSS'06.*, pages 142 –147, oct. 2006.

[23] PARTS. Power-Aware Real-Time Systems, `http://www.cs.pitt.edu/PARTS/`, 2003.

[24] T. Riegel, C. Fetzer, and P. Felber. Automatic Data Partitioning in Software Transactional Memories. In *20th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2008.

[25] H. Svensson. *Reconfigurable Architectures for Embedded Systems*. PhD thesis, Lund University, October 2008.

[26] Z. Ul-Abdin and B. Svensson. Evolution in architectures and programming methodologies of coarse-grained reconfigurable computing. *Microprocess. Microsyst.*, 33(3):161–178, 2009.

[27] M. Woh, M. Mahlke, T. Mudge, and Ch. Chakrabarti. Mobile supercomputers for the next-generation cell phone. *Computer*, 43:81–85, 2010.

[28] Ch. Wolinski and K. Kuchcinski. Automatic selection of application-specific reconfigurable processor extensions. In *Proc. Design Automation and Test in Europe*, Munich, Germany, March 10-14, 2008.

[29] Ch. Wolinski, K. Kuchcinski, and E. Raffin. Automatic design of application-specific reconfigurable processor extensions with UPaK synthesis kernel. *ACM Trans. Des. Autom. Electron. Syst.*, 15(1):1–36, 2009.

[30] Ch. Wolinski, K. Kuchcinski, J. Teich, and F. Hannig. Area and reconfiguration time minimization of the communication network in regular 2D reconfigurable architectures. In *Proc. of the International Conference on Field Programmable Logic and Applications (FPL)*, Heidelberg, Germany, September 8-10, 2008.