EUROPEAN COMMISSION

Community Research

SEVENTH FRAMEWORK PROGRAMME

Deliverable D10.11A

FINAL SUMMARY REPORT

Section A – "Publishable Report" including "Potential and Consolidated Impact"

EURETILE

Grant Agreement number: 247846
Project acronym: EURETILE
Project title: European Reference Tiled Architecture Experiment
Funding Scheme: Collaborative Project
Date of latest version of Annex I against which the assessment will be made: 2013 07 01
Period covered: from Jan 1st, 2014 to Sep 30th, 2014

Name, title and organisation of the scientific representative of the project's coordinator:
dott. Pier Stanislao PAOLUCCI, INFN (Istituto Nazionale di Fisica Nucleare)
Tel: +39 338 53 48 980
E-mail:pier.paolucci@roma1.infn.it
Project website address: www.euretile.eu

v20141113c

About this document

The “Final Summary Report” (deliverable D10.11) is composed of:

section A, this document, the “Final Publishable Report” (D10.11A), that includes an Executive Summary, a detailed report of R&D results, and a publishable section about the “Consolidated and Potential Impact” of the project.

Section B, document D10.11B, a “Plan for use and dissemination of foreground”.

Section C, document D10.11C, about “Societal Implications”

Section D, document D10.11D, about the “Final Distribution of Contributions”

EURETILE Final Summary Report

Pier Stanislao Paolucci[1], Iuliana Bacivarov[2], Andrea Biagioni[1], Clément Deschamps[3], Ashraf El-Antably[3], Nicolas Fournel[3], Gert Goossens[4], Rainer Leupers[5], Alessandro Lonardo[1], Luis Gabriel Murillo[5], Devendra Rai[2], Frédéric Rousseau[3], Lars Schor[2], Francesco Simula[1], Laura Tosoratto[1], Lothar Thiele[2], Piero Vicini[1]

[1]INFN Roma "Sapienza". [2]Computer Engineering and Networks Laboratory, ETH Zurich. [3]Université Joseph Fourier – TIMA Laboratory. [4]Target Compiler Technologies. [5]RWTH Aachen University – ISS & SSS

TABLE OF CONTENTS

1 Executive Summary

In the next decade, a growing number of scientific and industrial applications will require power-efficient systems providing unprecedented computation, memory and communication resources. Autonomous cars, for instance, will have to perform multi-sensorial data fusion to reach the required level of artificial intelligence, while high-impact scientific application, like brain-simulation, will require Exascale performances (10^{18} operations per second). A promising paradigm is the use of heterogeneous many-tile architectures. The resulting computing systems are complex and 1- must be protected against several sources of faults and critical events, and 2- application programmers must be provided with programming paradigms, software environments and debugging tools adequate to manage such complexity. The EURETILE (European Reference Tiled Experiment) consortium conceived, designed and implemented: 1- an innovative many-tile, many-process dynamic fault-tolerant programming paradigm and software environment (DAL), grounded over a lightweight operating system (DNA-OS) generated by an automated software synthesis mechanism (APES) that takes in account the architecture and application specificities; 2- a many-tile heterogeneous hardware system (QUonG), equipped with a high-bandwidth low-latency 3D-toroidal interconnect (APENet+). The inter-tile interconnect processor (DNP) is equipped with an experimental fault-tolerance, fault-awareness and fault-injection mechanism (LO|FA|MO); 3- a simulation environment (VEP), equipped with innovative parallelism and debugging facilities (add list). We also designed and coded a set of application benchmarks representative of requirements of future HPC and Embedded Systems, including: 4- a set of dynamic multi-media applications and 5- a large scale simulator of neural activity and synaptic plasticity (DPSNN-STDP) that will be used 1- for biological simulations; 2- to generate complex traffic patterns driving the design of future interconnect systems and computing nodes dedicated to brain simulation. One of the partner improved and experimented its: 6- tools for the cogeneration of ASIPs (TARGET). The application benchmarks, compiled through the EURETILE software tool-chain, have been efficiently executed on both the many-tile hardware platform and on the software simulator, up to a complexity of a few hundreds of software processes and hardware cores.

The consortium: The APE Parallel Computing Lab of INFN Roma is in charge of the EURETILE HW Design (QUonG system/APENet+ board/DNP (Distributed Network Processor) and Scientific Application Benchmarks. The Computer Engineering and Networks Laboratory (TIK) of ETH Zurich (Swiss Federal Institute of Technology) designs the high-level explicit parallel programming and automatic mapping tool (DAL) and a set of "Embedded Systems" benchmarks. The Software for Systems on Silicon (SSS) of the ISS institute of RWTH Aachen, investigates and provides the parallel simulation technology and scalable simulation-based profiling/debugging support. The TIMA Laboratory of the University Joseph Fourier in Grenoble explores and deploys the HdS (Hardware dependent Software) including the distributed OS architecture. TARGET Compiler Technologies, the Belgian leading provider of retargetable software tools and compilers for the design, programming, and verification of application-specific processors (ASIPs), is in charge of the HW/SW Co-design tools for custom components of the EURETILE architecture.

Project Foreground (exploitable Intellectual Properties) in alphabetic order: ASIC Design Tools, APENet+ (3D toroidal fault-aware interconnect board), APES (OS generator), DAL (Distributed Dynamic Application Layer), DALipse (graphical software development environment for DAL), DNA-OS (lightweight operating system), HySim (Hybrid Simulation), LO|FA|MO (local hardware Fault Monitor), DNP (Distributed Network Processor), DPSNN-STDP (Distributed Simulator of Polychronous Spiking Neural Net with Spiking Time Dependent Plasticity), Peer-to-peer GPU Direct RDMA IP, QUonG (many-tile fault-tolerant heterogeneous experimental platform), SWAT (System Wide Assertion), Thermal-RTC (toolbox for thermal / performance evaluation), VEP (Virtual Execution Platform), WSDB (Whole system debugger).

Grant Agreement no. 247846 Call: FP7-ICT-2009-4 Obj. FET-ICT-2009.8.1

Scientific Coordinator: Pier Stanislao Paolucci, INFN, Roma, Italy.

Administrative Coordinator: Michela Giovagnoli, INFN, Roma, Italy

2 Project context and objectives - summary

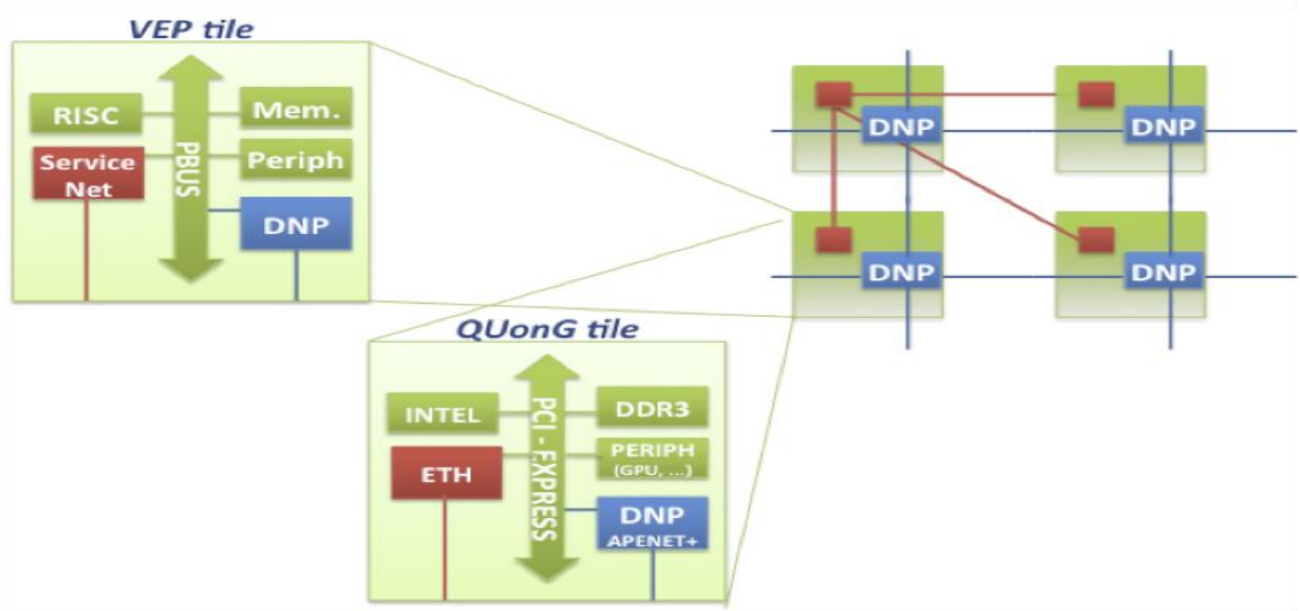
The original objective of the EURETILE project (**http://www.euretile.eu,** FET FP7 grant no. 247846) was the investigations of innovations to the many-process software and to the many-tile hardware architecture of future fault-tolerant Embedded Systems and High Performance Computers (HPC), for dynamic applications requiring extreme numerical and DSP capabilities. The project planned to deliver:

- Experimental Many-tile Hardware and Simulation platforms, for the scenarios of dynamic many-process workloads to be run on future fault-tolerant Embedded Systems and HPC;
- A many-tile programming/optimization environment, exhibiting several foundational innovations, to be applied to such dynamic many-process workload;
- A set of application benchmarks, for both HPC and Embedded System domains, coded using the new programming environment, including 1- a benchmark representative of distributed simulation of neural activity and synaptic plasticity (the DPSNN-STDP application see [Paolucci, 2013b]) that generates complex inter-process traffic patterns and 2- a set of dynamic multi-media embedded applications including a picture-in-picture software for embedded video processing systems as well as distributed implementations of a ray-tracing algorithm and an H.264 codec pair (see [Paolucci, 2014b]).

See [Schor, 2014a] for a description of the proposed software-flow and [Paolucci, 2013a] for a summary of the project motivations and a report about the first three years of the project (2010-2012).

2.1 Experimental Hardware and Simulation Platforms for HPC and Embedded Systems

A simplified view of the two many-tile execution platform conceived by the EURETILE project plan as representative of the Embedded and High Performance Computing domain is the following:



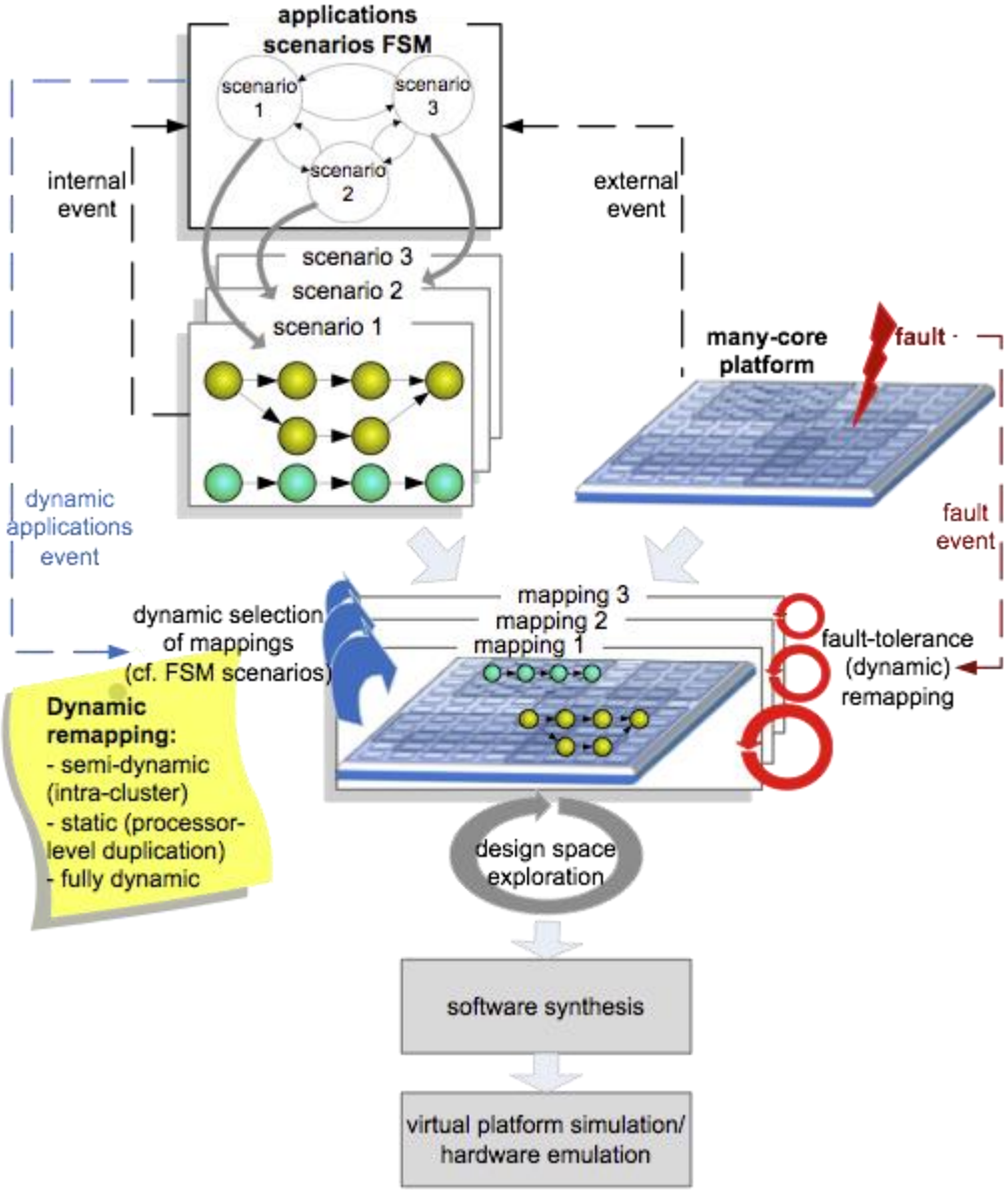
Simplified view of the EURETILE platforms (VEP and QUonG) used for experiments on innovations to many-tile hardware and many-process software of future fault-tolerant Embedded Systems and HPC platforms. The inter-tile communication is supported by a Distributed Network Processor (DNP)

1- The QUonG Hardware Platform for scientific High Performance Computing. We developed a custom interconnection board (APEnet+) that provides a 3D toroidal interconnection for many-tile systems and provides fault monitoring and fault tolerance features implemented by its DNP (Distributed Network Processors). Using the APEnet+ interconnect, the QUonG platform integrates off-the-shelf boards mounting multi-core CPUs and GPGPUs in a powerful many-tile hardware platform.

2- The VEP Embedded Systems many-tile simulation platform. VEP simulates the behaviour under-faults of many-tile systems and includes innovations to many-tile debugging and parallel simulation. The simulation framework includes many RISC based tiles, networked through a custom interconnect mesh composed of DNPs (Distributed Network Processors).

2.2 Definition of Software Tool-Chain for many-process dynamic workloads

From a software perspective, EURETILE aimed at providing a scalable and efficient extensive programming framework for many-tile platforms, by exploiting the underlying parallelism and investigating some key brain-inspired architectural enhancements specific to EURETILE.



Schematic View of the EURETILE Software Tool-Chain.

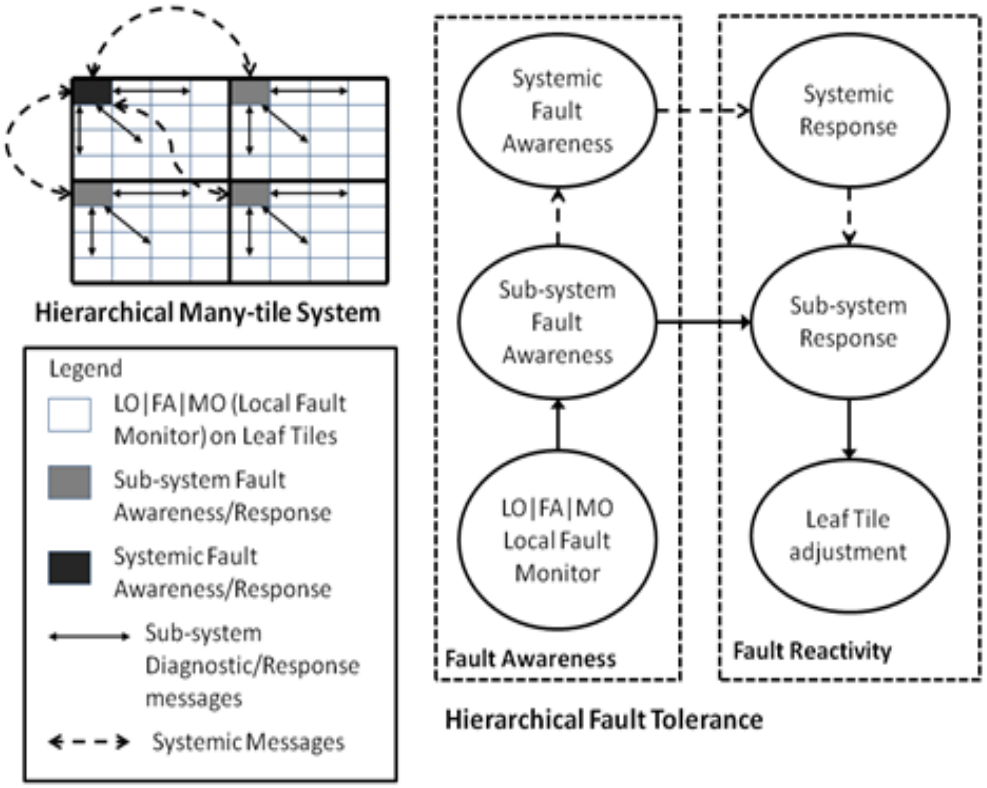
The proposed programming environment is based on a new model of computation that explicitly exposes the coarse-grain and fine-grain parallelism present in applications. Then, automatic tools are aiming at obtaining predictable and efficient system implementations by optimally matching the concurrency and parallelism present in applications, with the underlying many-tile hardware. The figure sketches the software EURETILE tool-chain.

The EURETILE software tool-chain follows a well-known Y-chart (application, architecture, executable) approach and in EURETILE each design phases have been enriched with novel concepts for programming

efficiently the layers of hierarchy present in the EURETILE platform. The proposed software tool-chain aims at optimizing issues related to system throughput while still guaranteeing real-time constraints for applications that operate under such restrictions. Additionally, fault-tolerance aspects are considered, from system-level programming. The way of representing parallelism, concurrency, and fault tolerance at system level, the implementation of the distributed real-time operating system, and the efficient fast simulation environment have all a strong impact on the overall optimality of the system implementation. The proposed extensive software design framework, including all challenges described above, created a foreground, the reference EURETILE platform, that can be used a starting point for the next years road map in many-tile processing.

2.3 Fault-Tolerance and Scalability

The "LO|FA|MO" design paradigm proposed by the project creates a *systemic awareness of faults and critical events*, thanks to a distributed approach that uses additional hardware components on each DNP, and needs dedicated software components running on each tile to create a *local awareness of faults and critical events*. This local awareness is then propagated along the system hierarchy. Onto such local and systemic fault awareness, a software approach to fault reactivity has been grounded. From a system-level perspective, two fault reactivity strategies have been investigated and implemented in the Distributed Application Layer (DAL), namely *fault recovery* and *fault tolerance*. On the one hand, the proposed fault recovery mechanism migrates processes that are assigned to faulty tiles to alternative tiles. On the other hand, to provide fault tolerance, applications with stringent performance requirements are duplicated during design space exploration.



The Network Processor of each leaf in the many-tile HW system is equipped with its own LO|FA|MO components. The Local Awareness of faults and critical events is propagated towards the upper hierarchy levels, creating Systemic Awareness. Reactions to faults and critical events are autonomously initiated by the sub-system controllers.

2.4 HdS and DNAOS

The project planned to ground its execution on the services provided by DNA-OS: a light operating system that can be customized depending on the requirements of the application and the characteristics offered by the hardware resources. DNA-OS has been developed and ported on several architectures (ARM, MIPS, …), and in the context of EURETILE had to be ported on the INTEL and RISC platform of the QUonG and VEP platforms. The plan was to develop a tool (APES) able to generate only the required services provided by DNA-OS regarding the application requirements for the target architecture. Based on this OS tool-chain, a main objective was to provide a tool-chain able to take as input the application model, as well as characteristics of the target architecture (multi-core, multi-tile) including a specific communication device (developed by INFN and called DNP) in order to generate all binary codes for all hardware computation resources. We planned to generate the software for both the execution platforms: 1- the many-tile hardware architecture for the HPC domain, and 2- the RISC-based simulation environment representative of the embedded domain.

2.5 ASIP design tools and acceleration of computation and communication

The new technologies of co-design of hardware architecture and programming tools of Application-Specific Processors that have been developed by TARGET and are being transferred from EURETILE will be an integral part of next-generation products of the leading EDA tools vendor Synopsys.

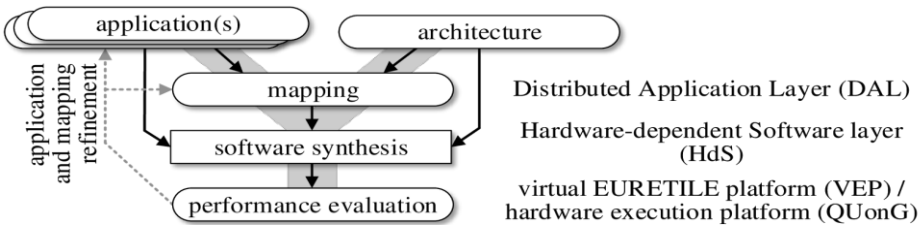
HW/SW co-design tools can operate at the level of elementary tiles. In EURETILE, Target Compiler Technologies, in cooperation with INFN investigated the application of its retargetable tools-suite for the design and programming of Application-Specific Processors (ASIPs). Building on this technology, Target contributed to the project in two major ways. First, new software-programmable accelerators can become part of the elementary tiles. These accelerators take the form of ASIPs optimised for the typical numerical kernels of the applications envisaged in the project. Architectural exploration is based on profiling of application code in the retargetable SDK. In addition, the tools generate efficient RTL hardware models of the ASIP, enabling a quick implementation on FPGAs. Secondly, Software Development Kits (SDKs) have been developed for the ASIP accelerators. Key elements of these SDKs are an efficient C compiler and an on-chip debugger. Third, TARGET investigated the application of ASIP to the acceleration of the DNP/APENet+ interconnection system.

3 Main scientific and technologic results and foregrounds

3.1 Many-Tile Platforms and Many-Process (Dynamic) Applications

Each hardware tile includes (a set of) general-purpose multi-core processors. The tile is equipped with a fault-tolerant distributed network processor for inter-tile communications. The tile can also include specialized accelerators. The programming environment maps multiple dynamic applications onto the many-tile hardware architecture. This elaborated high-level programming model specifies each application as a network of autonomous processes, enabling the automatic generation and optimization of the architecture-specific implementation. Behavioral and architectural dynamism is handled by a hierarchically organized runtime-manager running on top of a lightweight operating system. To evaluate, debug, and profile the generated binaries, a scalable many-tile simulator has been developed. High system dependability is achieved by combining hardware-based fault awareness strategies with software-based fault reactivity strategies. We demonstrated the capability of the software design flow to exploit the parallelism of many-tile architectures, using a set of embedded and high performance computing benchmarks targeting both the simulated and the hardware platforms, for applications described by a few hundreds of software processes, mapped on a few hundreds of hardware and/or simulated processors. Our approach reduces the "accidental complexity" when programming many-process applications and facilitates the hardware/software co-optimization of many-tile, many process systems. It automatically synthesizes architecture specific implementations and assists in the validation of many-process application and dynamic scenarios on many-tile systems. The project delivered a proprietary hardware for 3D-toroidal interconnect, equipped with fault-awareness hardware mechanisms, that can be used in conjunction with off-the-shelf interconnects. We also delivered a simulator of neural activity and synaptic plasticity that will be used 1- for biological simulations; 2- to generate complex traffic patterns that can be used to drive the design of future interconnect systems and computing nodes dedicated to brain simulation.

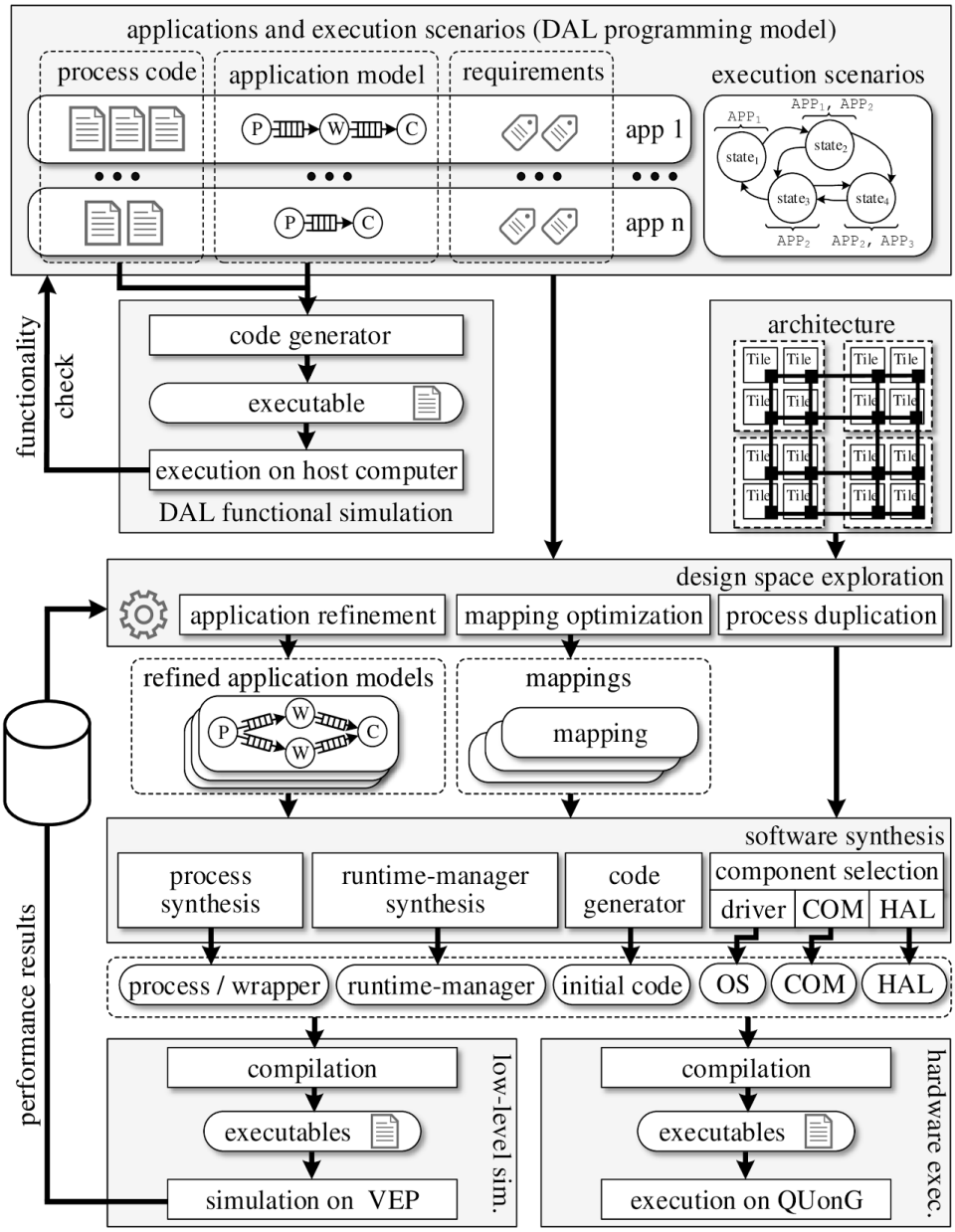
At the highest level of abstraction, the proposed design flow is represented by the following Y-chart:



The three layers of the EURETILE software design flow

3.2 EURETILE Design Flow: Many-process, dynamic applications on Many-tile platforms

The EURETILE design flow maps a set of dynamic applications that are specified as a network of processes onto a many-tile platform in a number of steps, as illustrated in the following figure.



EURETILE design flow to map a set of dynamic applications onto a many-tile platform.

3.2.1 Description of applications and scenarios: process networks and finite state machines

The input to the design flow is an abstract specification of the target architecture and a set of applications. Each application is specified according to the DAL programming model that specifies each application as an Expandable Process Network (EPN) [Schor, 2013a]. An application that is specified as an EPN consists of a set of autonomous processes, which communicate through point-to-point FIFO channels. The functionality of each process is specified in C/C++. In addition, some processes might also have a structural specification that specifies the functionality of the process as another process network. By specifying each application as an EPN, the application's best degree of parallelism can be selected automatically by the design flow so that scheduling

and inter-process communication overheads are minimized. In addition, as EPNs can be considered as an extension of Kahn process networks (KPNs) [Kahn, 1974], data races, non-determinism, or the need for strict synchronization are avoided. Each application may have its own performance requirements that must be met independent of the other applications.

Interactions between applications are represented as a finite state machine (FSM) with each state representing an execution scenario, i.e., a certain set of applications running in parallel. Transitions between scenarios are triggered by events generated either by running applications or by the operating system (OS).

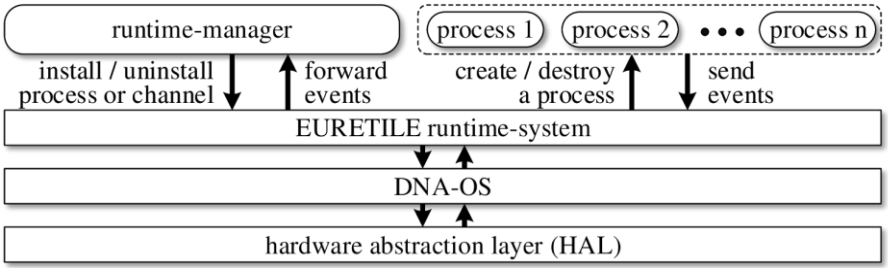
A first functionality check of the individual applications is realized using the DAL functional simulator. The DAL functional simulator executes each process as a POSIX thread on a host machine, thereby providing application-profiling data at a functional level. For more accurate performance results as, for instance, the run-times of processes, the system must be simulated on the VEP.

3.2.2 Optimization of the mapping using the VEP

During design space exploration, parallel processes are assigned to tiles and the applications are refined. Using the performance results derived from the VEP, the assignment of the processes (the so-called mapping) and the structure of the applications are iteratively improved until the performance requirements are fulfilled.

3.2.3 The software stack generated by the EURETILE design flow

During software synthesis, a four-layer software stack is generated that consists of application layer, runtime-system, OS, and hardware abstraction layer (HAL), see the subsequent figure for illustration. The employed tool chain first transforms the DAL processes to low-level threads that can be executed on top of DNA-OS [Guerin, 2009]. DNA-OS is a lightweight embedded OS that is used as OS in the proposed software stack. Afterwards, the tool chain generates one instantiation of the software stack for each tile.



Software stack generated by the EURETILE design flow.

3.3 Fault Management on EURETILE platforms

Integrated circuit technology scaling is increasingly making processors more vulnerable to faults. For instance, modern processors are more likely to experience single-event upsets, which were uncommon only a few years ago, and smaller transistors are responsible for higher power densities, which in turn cause temperature hotspots. In EURETILE, a high system dependability is achieved by combining fault avoidance, fault tolerance, and fault reactivity strategies. While the former two strategies are implemented at system-level by elaborating properties of the programming model, fault reactivity involves programming model, OS, and hardware. In the following, we will describe the key concepts of these fault management strategies.

3.3.1 Fault avoidance

Temperature related reliability issues are avoided by applying thermal-aware optimization strategies [Schor, 2013b], [Thiele, 2013]. During design space exploration, system designs that do not conform to peak temperature requirements are ruled out using formal thermal analysis methods. The methods are based on real-time calculus that is widely used to analyze and optimize real-time systems.

3.3.2 Fault tolerance

Applications with stringent performance requirements are duplicated during design space exploration [Rai, 2014]. In particular, the considered fault management strategy replicates critical sub-networks and inserts a special replicator channel that duplicates the stream to the corresponding input ports of the replicas. Similarly, a specially introduced selector channel arbitrates (merges) the data streams from the output ports of the replicas.

3.3.3 Fault reactivity

While the first two fault management strategies are applied at design time, fault reactivity is employed at runtime as a mechanism to react to faults. The proposed fault reactivity mechanism does this in a number of steps. First, the occurrence of a fault is detected by a novel hardware design paradigm named LO|FA|MO [Ammendola, 2014], which is based on fault monitors that are added to each tile. Once they detect a fault, LO|FA|MO propagates the information along the system hierarchy to the runtime-manager, which reacts to the fault by migrating processes that are assigned to a faulty processor to an alternative processor. To include the evaluation of all possible failure scenarios in the design time analysis, spare processors and tiles are allocated during design space exploration and used by the runtime-manager as target for process migration.

3.3.4 Task migration between tiles

Task migration has been known for a while in symmetrical multi-processor (SMP) architecture, as a solution to balance the workload between the different computation resources. This may be as well an answer to temperature hotspots, by shifting the execution of a task to another core or tile. In case of multi-tile, there is no shared memory that may ease communications, and the task migration becomes a real challenge. The chosen solution is a light migration method, which does not require any modification of the OS, keeping the OS as simple as possible (no virtual memory support, no dynamic loading).
The specification in DAL integrates the tile where a given task is supposed to migrate. A copy of the task is generated, but the replica is not started anymore. In case of migration request, the original task is stopped, and the duplicate is launched. We garantee that no message or data are lost during the migration. This migration method is built with additional tasks running on the different tiles to manage all the changes when a migration request is called. The overhead in terms of code size or performance is only of few %, and could be considered as negligible for complex applications.

3.4 Specification of the architecture of the target execution platform

In EURETILE, an abstract description of the architecture in terms of a hierarchically organized many-tile platform is elaborated. This representation is a generalization of the well-known tile-based multiprocessor model [Culler, 1999], which has been successfully applied in academia and industry.

The basic entities of the architecture model considered in the EURETILE project are the tiles. Each of them contains a distributed network processor (DNP), multiple general-purpose processors, and local memories. All processors in a tile might have access to commonly shared memory. Depending on the application usage, the tiles might also contain specialized hardware accelerators, like DSPs, ASIPs, FPGAs, or, in the latest hardware evolution, (GP)GPUs. The DNP is responsible for inter-tile communication and provides the interface to the network. The upper levels of the hierarchically organized architecture are constructed according to the distributed memory paradigm whereby multiple tiles together form an additional entity called cluster, which can be controlled autonomously.

As discussed in the previous section, spare processors and tiles are allocated at design time so that the runtime-manager can efficiently react to faults. The number of spare hardware elements depends on the number of faults that must be able to be tolerated. We call the abstract representation of the architecture without spare processors, tiles, and clusters the virtual representation of the architecture.

3.5 Programming model: applications plus scenarios

Next, we describe the considered high-level specification of the system that we propose to specify applications and execution scenarios. The basic idea of the programming model is to specify each application individually as an expandable process network (EPN) [Schor, 2013a] and to specify the interactions between the applications

as a finite state machine [Schor, 2012]. With this programming model, complex and dynamic interactions between applications can be specified.

3.5.1 Application Specification

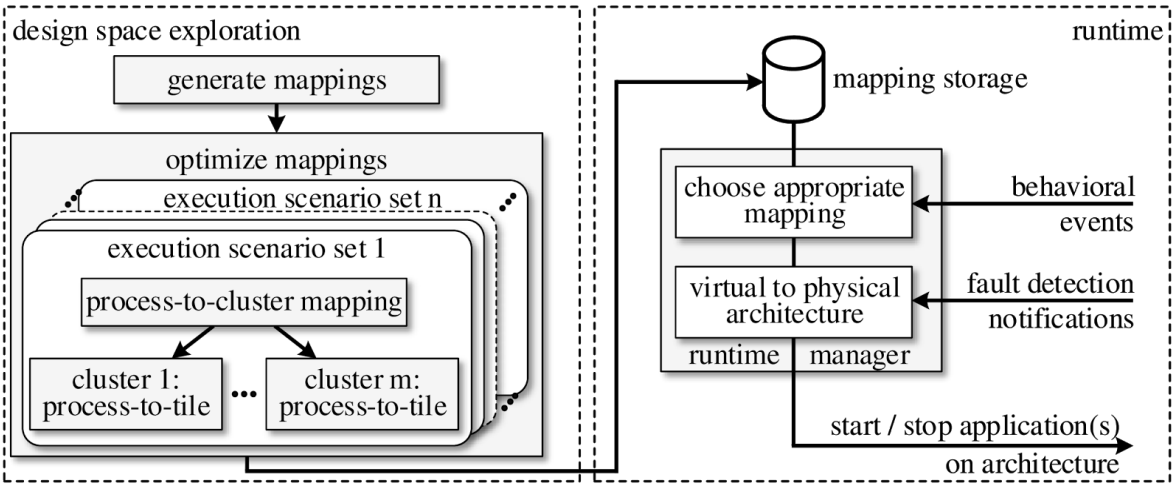
Each application is specified as an Expandable Process Network (EPN). The EPN semantics extends the KPN semantics [Kahn, 1974] by abstracting several possible granularities in a single specification. More detailed, an application specified as an EPN has a top-level process network that can be refined by hierarchically replacing individual processes by their structural specification. The structural specification describes the functionality of an application as another process network. This enables the automatic exploration of task, data, and pipeline parallelism by two refinement strategies, namely replication and unfolding. Replicating processes increases data parallelism and structural unfolding of a process increases the task and pipeline parallelism by hierarchically instantiating more processes in the process network. In addition, the functionality of each process is also specified in C/C++. The proposed application-programming interface (API) is composed of three procedures. The init procedure is executed once when the application is started. Afterwards, the execution of a process is split into individual executions of the fire procedure, which is repeatedly invoked. Finally, the finish procedure is called before the application is stopped. Each process can read from its input channels and write to its output channels by calling the high-level read and write procedures. Moreover, each process has the ability to request a scenario change by calling the send_event procedure. In addition, the topology of the application, i.e., the connections between processes by FIFOs, is specified in an XML format.

3.5.2 Execution scenarios

The dynamic behavior of the workload is captured by a set of execution scenarios forming a finite state machine (FSM). Each state represents a set of concurrently running or paused applications and each state transition corresponds to an application start, stop, pause, or resume request. Starting an application involves the installation of all processes and all FIFO channels of the application. After executing the init procedure of all processes of the application once, the fire procedures are iteratively executed by the scheduler. On the other hand, when an application is stopped, the fire procedure of all processes is aborted and the finish procedure is executed once. Finally, all processes and all FIFO channels are removed.

3.6 Mapping strategy

The mapping decides on the distribution of the processes on the architecture. As typical mapping strategies that calculate a single mapping are no longer capable of efficiently utilizing the hardware if the functionality of the system can change at runtime, a hybrid design time / runtime mapping strategy is elaborated in EURETILE [Schor, 2012]. At design time, an optimal mapping is calculated for each application and scenario where the application is running, whereby the virtual representation of the architecture is used as target architecture. Then, a runtime-manager controls the dynamic behavior of the system. Whenever an application is started or stopped, it first selects the appropriate mapping (onto the virtual representation) and then maps the virtual representation onto the physical architecture. This enables the runtime-manager to react to faults without recalculating the mapping by just adjusting the binding of the virtual representation onto the physical architecture. In the following, we will detail the design time analysis and the runtime management. The overall hybrid mapping strategy is illustrated below.



Hybrid mapping environment.

3.6.1 Design time: analysis and optimization

At design time, an optimal mapping for each pair of application and scenario, where the application is running, is calculated [Kang, 2012]. Therefore, the output of the design space exploration is a collection of optimal mappings and exactly one mapping is valid for a pair of application and scenario. To minimize the reconfiguration overhead, an application has the same mapping in all connected execution scenarios so that a running application is not affected by the start or stop of another application.

More detailed, a two-step procedure is applied during design space exploration in order to efficiently calculate the mappings. First, it is calculated, which pairs of application and scenario must use the same mapping so that no process migration is required. We do that by calculating for each application separately the maximally connected components of a sub graph, which only contains the scenarios where the application is running. At the end of this step, one mapping is allocated for each component of the sub graph. Second, the previously allocated mappings are optimized so that the objective function is minimized and additional architectural constraints (e.g., processor utilization, link bandwidth, and chip temperature) are fulfilled. The objective function depends on the intended use and may include more than one objective.

However, multi-objective meta-heuristics, that have been successfully applied to solve the mapping problems of multi-tile systems, are no longer effective for many-tile systems due to the scale of the investigated problem. Therefore, a multi-objective mapping optimization technique is used that overcomes this shortcoming by decomposing the mapping problem into independent sub-problems. As illustrated in the figure above, two different problem decompositions are considered: the execution scenarios are decoupled into independent sets and an architecture-based decomposition is applied that first assigns processes to clusters. Afterwards, and for each cluster separately, the processes are assigned to the tiles.

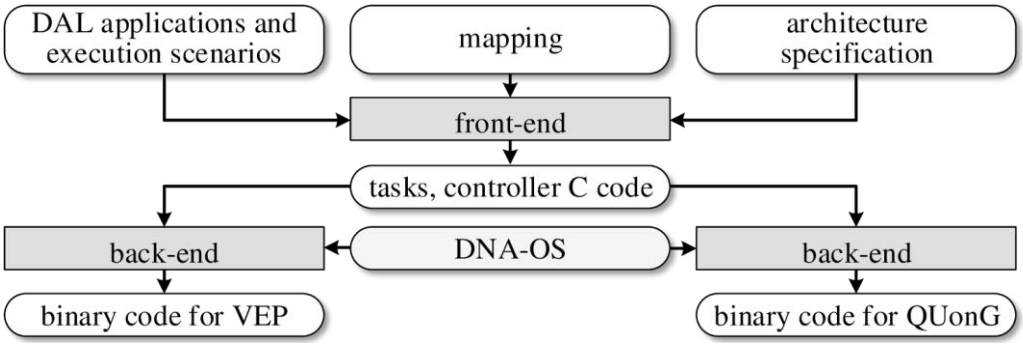
3.6.2 Runtime management

The task of the runtime-manager is to send commands to the runtime-system so that the execution semantics of the system is ensured. To this end, the runtime-manager receives and processes behavioral events and fault detection notifications. A behavioral event is sent by an application and triggers a scenario change. A fault detection notification is sent by LO|FA|MO if a hardware fault has occurred. The latter only changes the binding of the virtual representation onto the physical architecture, but not the mapping of the applications onto the virtual representation.

Consequently, the runtime-manager consists of two components. The first component is responsible to handle behavioral events and ensures the execution semantics. It is just aware of the virtual representation of the architecture. The second component processes the fault events and redirects the commands to the corresponding physical network. If the runtime-manager receives a fault event, it migrates the processes that are assigned to the faulty processor (tile) to a spare processor (tile) and changes the binding of the virtual representation onto the physical architecture by reconfiguring the second component to redirect the commands to the new target.

3.7 Generation of the executable

We have defined a SW tool-chain that is able to accept as input a DAL model and generates a binary code that runs on different architectures: x86 simulator and QUonG, and iRiSC based VEP simulator. We can split this tool-chain in two parts, the front-end part transforms DAL model to a multi-threaded C code application, and then the back-end part compiles and links the code obtained in the front-end part on top of DNA-OS and generates the binary code for the targeted processors and architecture.



Overview of the software synthesis tool-chain as elaborated in the EURETILE design flow.

3.7.1 DNA-OS

DNA-OS operating system implements exokernel architecture. The MIT defines exokernels as follows: "An exokernel eliminates the notion that an operating system should provide abstractions on which applications are built. Instead, it concentrates solely on securely multiplexing the raw hardware: from basic hardware primitives, application-level libraries and servers can directly implement traditional operating system abstractions, specialized for appropriateness and speed". DNA-OS supports or is based on traditional features of OS: Hardware abstraction layer (processor specific routines or low level boot code, …), cores services (scheduling, synchronization…), device drivers (based on the POSIX.1-2001 standard concerning the files and devices access), memory management (system and user map)

3.7.2 Software Synthesis Front End

The tool-chain starts with the following DAL input files:

- The platform description (in xml format)
- The mapping file (in xml format)
- All application network descriptions for each application (in xml format)
- C code of the application tasks (including the 3 required functions of the DAL model for each application: app_init, app_fire and app_finish)

Moreover, we take as input the C code of the controller (master and slave tasks) which manages the overall behaviour of applications. The parsing of all these files determines the need in terms of communication channels as well as all tasks which are executed on each computing unit. By this way, we can produce a main C code for each processing unit with all the possible running tasks and all possible communication channels. The controller explicates for each state tasks that are running on each processing unit. When all tasks are created, all of them that are not supposed to be executed in the initial state are paused. When entering in a new state, some of them are resumed, and some of them are paused to fit with the state task execution specification.

3.7.3 Software Synthesis Back end

The back-end part of the tool-chain consists in taking the C code (tasks, controller), the generated DNA-OS, and then to compile and link to get binaries. The two different target architectures (iRiSC-based for embedded domain and x86 QUONG machine for HPC) leads to two different tool-chain for what concerns the back-end.

3.7.4 Hierarchical Run-time manager

The dynamic behavior of the system is captured by a set of scenarios, formally described as a finite state machine (FSM). A hierarchically control mechanism is proposed. This hierarchy follows the architecture structure and is represented in DAL as a process network including three different types of controllers:

- slave controllers responsible just for the activity inside a cluster,
- interlayer controllers responsible for a network,
- master controller responsible for the upper level network that processes all events that cannot be handled by any other controllers.

The controller explicates for each state tasks that are running on each processing unit. When all tasks are created, all of them that are not supposed to be executed in the initial state are paused. When entering in a new state, some of them are resumed, and some of them are paused to fit with the state task execution specification.

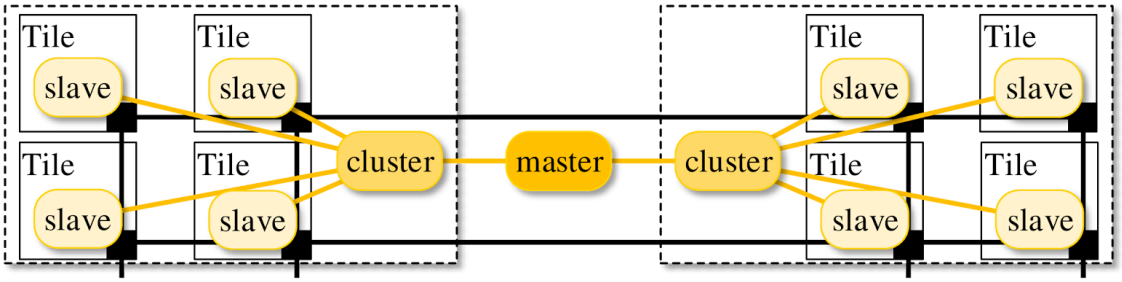


Illustration of a hierarchically organized runtime manager.

3.7.5 Specificities of generation for VEP Simulated Platform

The IRISC processor is the main programmable processing unit used in the embedded flavour of the Virtual EURETILE Platform (VEP). This core was developed by RWTH Aachen following an ADL-based (Architecture Design Language) methodology, which is supported by a centralized processor model. From this centralized model, it is possible to generate a set of software development tools in a (semi-)automatic way, which is guaranteed to comply with the processor specification. The most notable tools among the IRISC tool-chain are: lasm (LISA assembler), llnk (LISA linker), lcc (LISA C compiler) and ldb (LISA GDB). lasm, llnk and lcc enable creating binary executable objects for the IRISC processor. All tools provide a command line interface, which allows their integration into the DNA-OS (ported on the iRiSC) code generation process. These tools are already integrated into the EURETILE tool-chain, thus allowing the compilation and linking of the hardware-dependent software and the DAL applications for IRISC.

Additionally, there are IRISC tools, which support programmers during development and debugging. An application can be debugged either by using the command line GDB tool for the IRISC (ldb) or the Synopsys Processor Debugger GUI (pdbg). The former provides a traditional source-level debugger whereas the latter provides advanced features for low-level debugging of the processor (e.g. pipeline stages and registers). Both debuggers have been used for porting the DNA/OS version to IRISC and for verifying the tool-chain.

Finally, the IRISC tool-chain also provides a set of basic C libraries for software programming, namely the ISO C library (LIBC), a C runtime support library (LIBRT), a floating point emulation library (LIBFPE) and an abstraction layer library for retargeting to different OSs (LIBOS). It is worth to mention that LIBOS was ported to support DNA/OS as host operating system.

3.7.6 Specificities of generation for QUonG Hardware Platform

DNA-OS has first been ported on x86 and QUonG hardware platform. The PCI is a local bus, which provides a simple way to connect boards, and it was introduced by Intel in 2004. PCIe (PCI express) is derived from the PCI (Peripheral Component Interconnect) and offers more powerful features. A PCIe driver has been developed in 2012. This feature is now used to access real hardware components on the QUonG board, as the QUonG board is connected to the motherboard using a PCI express bus. For two years, DNA-OS has supported the PCIe.

The driver represents about 10 000 lines of C code specifically written for DNA-OS (based on Linux PCIe driver).

3.8 The Virtual EURETILE Platform and Supporting Simulation and Debugging Technologies

Full-system simulation plays an essential role to enable early design exploration, bring up the system software and assess behavioural characteristics. that would be difficult to examine with hardware prototypes. Due to the massively parallel nature of the EURETILE architecture, the requirements for a full-system simulator environment exceed those of state-of-the-art tools, as it should:

- Provide fast and scalable simulation to experiment with a possibly large number of tiles/cores.
- Adapt to different use cases like software development and network scalability tests.
- Provide means to facilitate debugging and profiling of concurrent software.
- Allow fault injection to experiment with fault-awareness mechanisms at other system levels.

In EURETILE, research has been carried out on abstract and parallel simulation as well as on multicore debugging/profiling technologies. The outcome of this research was materialized in the Virtual EURETILE Platform (VEP) and a set of technologies which are described in the following.

3.8.1 VEP characteristics and architecture

The VEP is a SystemC based simulator that models an embedded version of the EURETILE architecture. It allows simulating system configurations composed of several tiles arranged in a complete 3D grid and connected in a 3D toroidal topology. The simulator was created in a flexible way so as to allow runtime user specification of the number of tiles, memory address maps, processor frequencies, among others. VEP tiles contain a small form factor RISC processor tailored for embedded applications, namely the IRISC. The IRISC is created with Synopsys Processor Designer in the LISA language [Synopsys, 2014] and thus allows easy customizations for deriving new specialized ASIPs. A VEP tile also consists of TLM2 (transaction level) models of a memory management unit, memory blocks, a bus, a serial I/O interface, timers, a real-time clock, an interrupt controller and the DNP. A service network interface used for fault monitoring completes a tile. The VEP simulation models are augmented with debugging, tracing, and fault injection capabilities. For the latter, users can specify faults in different components and at precise points in time. Supported faults include processor and DNP breakdowns, severed DNP connections, isolation of tile groups, random bit-flips, among others.

3.8.1.1 Processor model abstractions and optimized variants

The processor model that is in every tile incurs the major cost in terms of simulation speed and memory-footprint. The IRISC instruction-set simulator (ISS) was created at different levels of abstraction to allow trading off speed and accuracy. A cycle-accurate (CA) and an instruction-accurate (IA) IRISC model based on Just-in-time Cache-compiled (JIT-CC) technology [Nohl, 2004] were developed to facilitate two principal use cases: accurate performance analysis and software development. With these models, it is possible to simulate up to 200 tiles. Furthermore, two different optimized variants of the IA ISS were created to satisfy other use cases. The first variant, namely the IRISC IAP, is a simplified model without debug features but compatible with 64-bit compilation, which allows increasing the maximum number of simulated tiles. In a stand-alone simulation, the IRISC IAP executes approx. 24x and 4x faster than the CA and IA models, respectively. Scenarios running applications on 512 tiles have been successfully tested on the IAP but up to a few thousand tiles can be fit in memory. The second variant, namely the IRISC DBT-IA, is based on a dynamic binary translation (DBT) engine [Jones, 2009] that achieves higher speed than the JIT-CC engine. It executes approx. 170x, 30x and 7x faster than the CA, IA and IAP models, respectively, but only 32 tiles can be simulated in a single host due to its stringent memory consumption.

3.8.1.2 Abstract simulation

The VEP can also be executed as an abstract platform (AS) where the ISS is replaced by a host-compiled simulator. In this form, every tile is provided by the Abstract Execution Device (AED), which is a SystemC module that can execute target software compiled for the host machine. The AED communicates with the DNP and other on-tile devices through an interface that is identical to that of the IRISC ISS. It also provides features like software timing annotation and limited visibility of tile-scope global variables. This mechanism was used

to ramp up DNP drivers, test software before the tool-chain for the IRISC was available and evaluate the network scalability. Due to its host-compiled nature, the AED is limited to software without target-specific constructs (e.g., inline assembly). To circumvent this issue, the hybrid simulation technology (HySim) [Kraemer, 2007] that was previously developed by RWTH for the FP6 SHAPES project was revamped. HySim has a virtualization tool-chain that automatically selects and instruments code that can be executed in AS mode, and switches dynamically between AS and ISS when needed. The new HySim can now be used with (i) customizable architectures [Jovic, 2012] and (ii) multi-core systems [Murillo, 2012a].

3.8.1.3 VEP use cases

The VEP allows targeting different use cases, as shown in the Table. While the CA version is good for accurately assessing the performance of applications, its speed is not enough for regular programming tasks. The IA models are better suited for software development. The IRISC IA is commonly used for software programming and debugging of medium size systems (32 to 200 tiles). The IRISC IAP is used for software testing in large systems and DNP network stress tests (from 200 tiles to approx. 1000 tiles). The IRISC DBT-IA is used for development and debugging of small systems running computational expensive applications. In AS mode, it is possible to simulate very large systems. Thus, the AS is better suited as a DNP-centric simulator to test network scalability and fault awareness features.

Summary of VEP levels of abstraction and use cases

	IRISC CA	IRISC IA	IRISC IAP	IRISC DBT-IA	AED
Stand-alone MIPS	~1	~6	~25	~160	~3000
Number of Tiles	~200	~200	~1000	~32	~20000
Debug Support	Yes	Yes	No	Yes	No
Use Case	Performance analysis	SW development, debugging, small to medium systems	DNP stress tests, SW testing in large systems	SW development debugging, complex. apps, small systems	DNP network scalability testing

3.8.1.4 Other speed optimizations

The VEP has various optimizations which are commonplace in full-system simulators. Temporal decoupling (see e.g. [Murillo, 2012a]) and Direct Memory Interface (DMI) access were introduced. Moreover, unnecessary overhead caused by SystemC clock objects was mitigated by, first, adding a custom fast clock implementation and, second, combining multiple clocks with identical timing into single objects. Finally, a processor idle state that is made visible to the simulation kernel was introduced to avoid processing unnecessary core-related SystemC events.

3.8.2 Parallel Simulation Technologies

Although the VEP is compatible with the regular OSCI SystemC kernel, the performance degradation of large system topologies called for the adoption of parallel simulation technologies. This resulted into the creation of two parallel SystemC kernels and a library for distributed SystemC.

3.8.2.1 parSC, SCandal and legaSCi

The parSC simulation kernel [Schumacher, 2010] enables parallel execution by distributing simulation events (such as CPU clock ticks or interrupts) that happen at the same point in time (i.e., the same SystemC delta cycle) among different threads. This strategy is convenient for detailed simulation models (e.g., CA) since they usually have a high number of events occurring concurrently and can as such benefit most from fine-grained parallelism. The parSC approach is, however, prone to concurrency bugs (e.g., data races) because state is usually shared between different SystemC processes in a module. To mitigate this issue, SCandal [Schumacher, 2012a] is a tool designed to analyse non-deterministic traits in SystemC code that might lead to functional anomalies. It was specifically designed to target industry level simulators that use third party binaries where no source code

is available. Additionally, legaSCi [Schumacher, 2013] is an extension to parSC that enables race-free operation by restoring and enforcing sequential process execution, inside the simulation kernel. legaSCi enhances the parSC process scheduler such that all processes marked as part of the same legacy component are scheduled on the same OS thread. parSC with the legaSCi extensions achieves a performance of more than 2x when used with the VEP on a quad-core simulation host. As complementary work in the area of deterministic simulation, a deep analysis of the effects and causes of nondeterministic behaviour in sequential and parallel SystemC simulations was performed [Schumacher, 2012b].

3.8.2.2 SCope

SCope [Weinstock, 2014] is a SystemC kernel that exploits coarse-grained parallelism in simulations. It achieves high speedups when simulating less detailed models, such as those compliant with TLM2. SCope groups several simulation models (e.g., the components of a tile) and then assigns the groups to different host threads. The groups then operate temporally decoupled with regards to the system, i.e., individual groups are allowed to advance in time without frequent synchronization with the other threads. By decreasing the amount of synchronizations, the overall performance is increased. On a quad-core host, SCope gives an speed-up of around 4x compared to OSCI kernel. Dynamic load balancing has also been implemented for SCope that enables moving around components from thread to thread at the SystemC module level. Load balancing has been proven on the VEP with application scenarios that examine different tile workload profiles. For fully unbalanced simulation scenarios, the simulator is 1.35x faster when the load balancing feature is enabled.

3.8.3 Multicore Debugging Technologies

Software debugging for parallel systems is one of the most expensive tasks during the design cycle. The following presents the EURETILE multicore debugging technologies used with the VEP.

3.8.3.1 The Whole-system Debugger – WSDB

Software debuggers for parallel systems should address key issues like abstraction, retargetability, scalability and convergence of information from different data sources. WSDB is a new generic full-system interactive source debugger based on a component-based architecture [Murillo, 2012b] that fulfills these requirements. WSDB offers APIs that facilitate interacting with systems composed of multiple cores. For instance, it allows iterating on groups of tiles, cores and threads for breakpointing, inspection and control. Furthermore, it can be easily extended to perform a multitude of debug tasks by setting user-defined callbacks on internally triggered events (e.g., OS-level events). Other features of WSDB include support for the ELF, DWARF and STABS formats, a flexible stack unwindig API, an API for defining OS-awareness layers, and a debugger plug-in for the Eclipse IDE.

3.8.3.2 SWAT: System-wide Assertions

System-wide Assertions (SWAT) [Murillo, 2015] allow correlating concurrent events from different sources for debugging purposes. SWAT allows inspecting and evaluating software variables and hardware registers/signals as well as defining behavioural patterns with linear temporal logic (LTL), which are then monitored at runtime. Event correlations defined in SWAT are useful for debugging traditional sequential issues and concurrency bugs. Issues at the hardware/software boundary can also be handled efficiently with SWAT, as it provides the capabilities of a source debugger and an assertion-based verification system. SWAT provides a language and a compilation framework. The framework's output is an automatically generated runtime assertion monitor. These monitors are executed externally on a non-intrusive debugger without altering the target system's behaviour. SWAT can also be used as a low level language for finding bugs based on concurrency bug patterns [Murillo, 2011] or other similar specifications.

3.8.3.3 Concurrency analysis and behaviour exploration framework

Dealing with concurrency bugs is a difficult task. Most bugs remain unnoticed and, when they appear, they are hard to reproduce. We developed a concurrency exploration framework for virtual platforms based on dynamic analysis of event ordering constraints [Murillo, 2014]. The approach identifies conflicting concurrent interactions among system components, which are in turn used by the framework to reproduce a previously observed behaviour or intentionally trigger a buggy states. The framework also enables precise control of the

execution of all individual software tasks and components on the target (at the level of monitored events). This is used by bug finding algorithms to manipulate the target's behaviour. An exploration process based on random constraint swapping, which helps to detect actual bugs, completes the debug approach.

3.9 The Distributed Network Processor in VEP

The Distributed Network Processor (DNP) is the INFN intellectual property that implements the 3D torus network connecting the computing nodes in the EURETILE platform. It provides deadlock-free data packets routing and guarantees zero-copy data transfers by supporting the RDMA protocol.

A SystemC TLM model of the DNP is integrated in the VEP platform, one instance per tile. The DNP channels connect the tiles each other in a 3D mesh, and the component simulate the data transfer and routing functionality, with real hardware (APEnet+) latency timings. The DNP model allowed to validate the DNP architecture and to test its scalability. Moreover it has been used to prove the validity of the LO|FA|MO approach to systemic fault awareness (see next section), before of its hardware implementation.

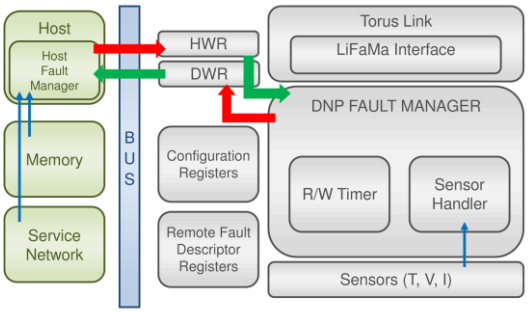
As the model is lightweight and frugal in resource consumption, simulation techniques could be applied aimed at increasing the simulation speed and the possibility to scale the size of the simulated platform. The VEP tile processor (either IRISC or AED) can program the DNP, as a memory mapped device. A higher level API, the DNP RDMA API, allows application level programming, but a more user-friendly interface is the PRESTO library that, on top of the DNP RDMA API, provides MPI-like send/receive primitives for data exchange.

3.10 LO|FA|MO: Fault Detection and systemic Awareness for QUonG and VEP

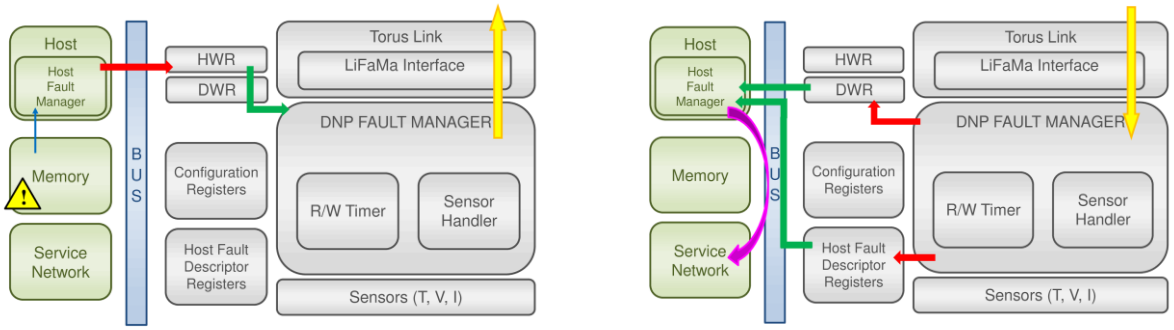
Local Fault Monitor (LO|FA|MO) is a systemic approach to fault detection and awareness for distributed systems. We wanted this approach to guarantee a *no-single-point-of-failure* fault awareness, meaning that the presence of a faulty component should not prevent the system to become aware of it. With these guidelines we designed LO|FA|MO as based on the following principles:

- A LO|FA|MO-enabled 3D toroidal network interface, i.e. the DNP, sporting a dedicated hardware component (DNP Fault Manager-DFM) , able to assess the DNP status, and registers containing (encoded) the DNP status, the host status and the status of the first neighbours hosts in the 3D network.
- A dedicated software (Host Fault Manager-HFM) running on each host able to assess the host status and the DNP status as explained in the next point.
- A Mutual Watchdog mechanism between host and DNP on each node, in which the two are peers reading each other status from the watchdog registers and updating their own; periodical read and write timing (with Tread > Twrite) ensure that the peers can determine each other liveness.
- A Service Network for diagnostic messages and accessible by each HFM instance.
- The 3D network implemented by the DNPs as a secondary path for diagnostic messages issued by each DFM.

In the mesh of nodes we also name Supervisors those nodes that monitor the system at higher level in the hierarchy, thus being final target of all diagnostic messages and center of decisions about fault reactivity.



LO|FA|MO mutual watchdog mechanism in which host and 3D Network interface are peers monitoring each other by periodically reading and writing special DNP registers (DNP Watchdog Register and Host Watchdog Register).



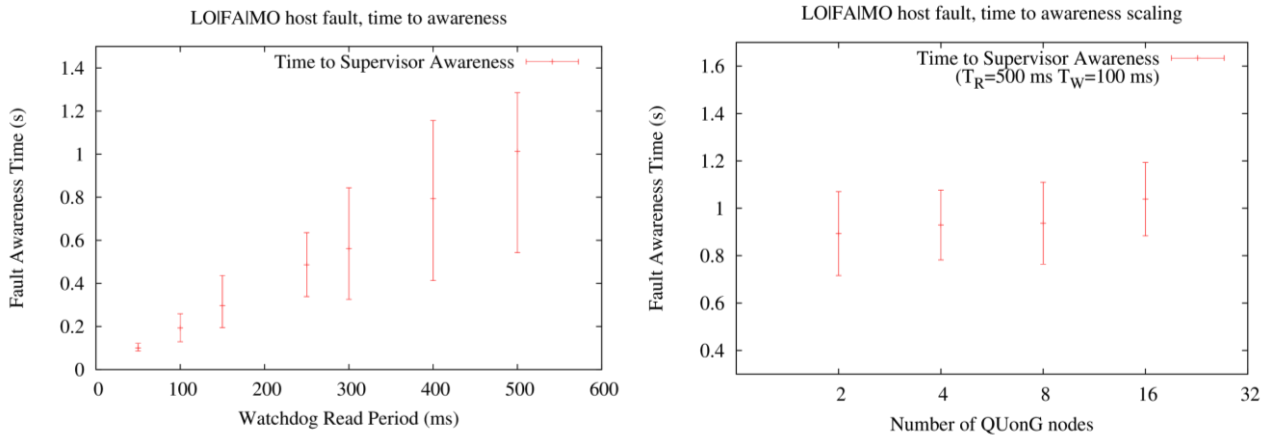
An example of a host memory fault detection and systemic awareness. (Left) The fault is detected and reported in the Host Watchdog Register, the DNP Fault Manager reads the register and sends a diagnostic message via 3D network to the first neighbour nodes. (Right) Information about the fault carried by the diagnostic message coming from the 3D network is reported in the other node's DNP Watchdog Register where the Host Fault Manager can read it. Then Host Faul Manager can inform the Supervisor node through the Service Network.

3.10.1 LO|FA|MO implementation on VEP

The DNP model in the VEP contains the DNP Fault Manager block as already mentioned in section 3.9. A Host Faut Manager software implementation runs in the AED processor to complete the mutual watchdog functionality. The entire LO|FA|MO approach has been validated on the VEP by using its fault injection facilities. The validation covered the DNP diagnostic logic in the model, the VEP service network and the AED Fault Manager. VEP configurations with 64 (4x4x4) and 512 (8x8x8) tiles were used for the validation.

3.10.2 LO|FA|MO implementation on QUonG

LO|FA|MO has been implemented and used efficiently on the QUonG cluster [Ammendola, 2014]. In this implementation the APEnet+ core contains a DFM component, whose resource occupancy (FPGA gates) is very low compared with those of the whole DNP core. The Watchdog Registers are on the FPGA and accessible in target mode on the PCIe bus in ~6 microseconds. Note that at the access rate corresponding to Tread and Twrite (in the range ~10-1000 milliseconds) the bus occupancy is very low.



(left) Time to obtain Supervisor awareness (T_{aw}) in case of host breakdown fault, plotted varying the watchdog period T_{read}. For each T_{read} value the mean value and the minimum and maximum values are shown for T_{aw}.

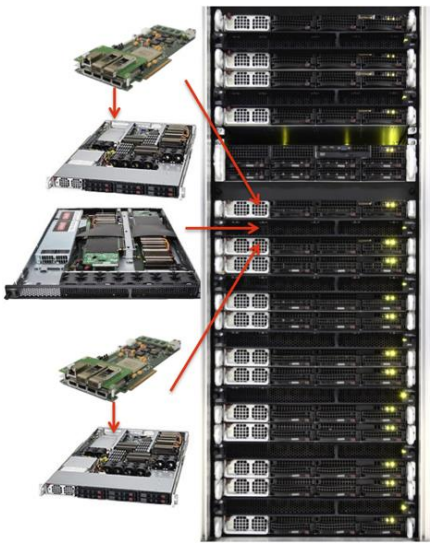
(right) Time to Supervisor awareness (T_{aw}) scaling the number of QUonG node running LOFAMO. T_{read} = 500ms and T_{write} = 100ms.

The DFM is able to embed diagnostic messages in the physical link protocol of the 3D network, leading to a zero impact of these messages on the NIC performance. Finally the HFM is a Linux multi-thread daemon running on each node CPU, but usage fraction is negligible.

3.11 Hardware Experimental Platform (QUonG)

QUonG is a comprehensive initiative aiming at providing a hybrid, GPU-accelerated x86_64 cluster with a 3D toroidal mesh topology, able to scale up to $10^4/10^5$ nodes. During 2013, a QUonG cluster was assembled and put into service, with 16 identical tiles (nodes).

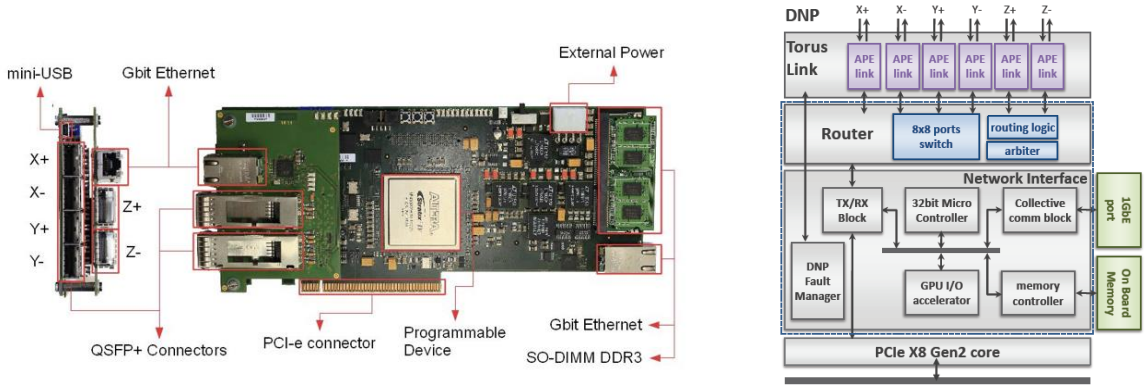
They are arranged into a 4x4 network mesh served by one 6-links APEnet+ card per tile. Each tile provides 2 Xeon Westmere-family CPUs, 2 M2075 NVIDIA GPUs and 48GiB of memory; QUonG tiles also partake in two supporting networks, InfiniBand– and GbE-based. Applications can use either the EURETILE software tool-chain (DAL+DNA-OS for QUonG), or standard high-level communication semantics like the Message Passing Interface (OpenMPI and MVAPICH2 implementations are available for InfiniBand and APEnet+ networks) or low-level, interface-specific ones like Verbs or Sockets Direct Protocol (SDP) for InfiniBand and the custom RDMA API for APEnet+. A disparate set of scientific codes has been run on QUonG, in order to stress-test the installation, among which the Distributed Polychronous Spiking Neural Network simulator. In parallel, joint work with TIMA led to a quite stable release of DNAOS on x86_64 architecture whose highlights are the support of APEnet+ and Intel GbE Ethernet cards as channels for remote communication among DAL processes.



QUonG experimental many-tile hardware platform: 2013 installation

3.12 The APEnet+ interconnection system

APEnet+ is a point-to-point, low-latency network controller developed by INFN for a 3D-torus topology integrated in a PCIe Gen2 board based on an Altera Stratix IV FPGA. It is the building block for the QUonG hybrid CPU/GPU HPC cluster inside INFN [Ammendola, 2011] and the basis for a GPU-enabling data acquisition interface in the low-level trigger of a High Energy Physics experiment [Ammendola, 2013]. The board provides 6 QSFP+ modules which are directly connected to its embedded transceivers; 4 out of 6 modules are placed onto the main board and 2 more reside on a small piggy back card, thus resulting in a 2-slot wide card. PCI Express mechanical compatibility is preserved with x16 cards. Each Altera Embedded transceiver is capable of a data rate up to 8.5 Gbps in fully bidirectional mode, and a single remote data link is built up by bonding 4 transceivers composing up a link operating at up to 34 Gbps. A complete assembly of the card is shown in figure below.

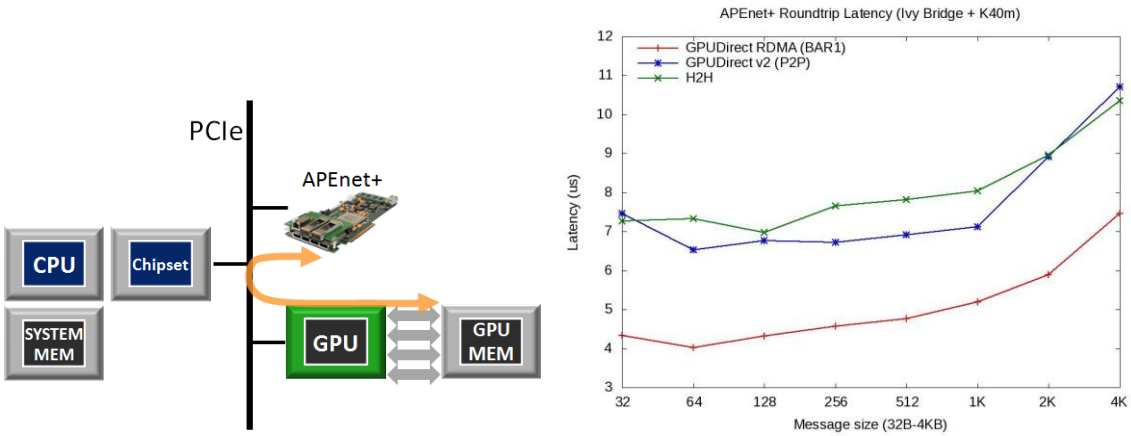


(Left) APEnet+ interconnect board: 2011 release. (Right) Outline of the main logic blocks of the FPGA architecture.

The Distributed Network Processor (DNP) is the core of the APEnet+ architecture; it acts as an offloading engine for the computing node, performing inter-node data transfers. The Torus Link block manages the data flow by encapsulating packets into a light, low-level word stuffing protocol able to detect transmission errors via CRC. The APElink [Ammendola, 2013a] data transmission system sustains a channel bandwidth up to 2.8 GB/s. APElink custom logic [Ammendola, 2013d] enables a global APElink efficiency of 93%. The Router component is responsible for data routing and dispatching, dynamically interconnecting the ports of the cross-bar switch; it is able to simultaneously handle 7 flows @3.0 GB/s. The Network Interface is the packet injection/processing logic; it manages data flow to and from either Host or GPU memory. On the receive side, it provides hardware support for the Remote Direct Memory Access (RDMA) protocol, allowing remote data transfer over the network without involvement of the CPU of the remote node. An integrated microcontroller provided by the FPGA platform allows for straightforward implementation of RDMA semantics. The Network Interface is also able to directly access the memory of Fermi- and Kepler-class NVIDIA GPUs implementing GPUDirect V2 (peer-to-peer) and GPUDirect RDMA capabilities [Ammendola, 2013b]. A dedicated component (DNP Fault Manager) provides Fault Awareness capabilities as described in section 3.10.The hardware blocks in the DNP structure are depicted in figure below.

3.12.1 GPUDirect technology on APEnet+

Fermi is the first NVIDIA GPU architecture which externally exposes a proprietary HW-based protocol to exchange data among GPUs directly across the PCI Express bus (PCIe), a technique which is generically referred to GPUDirect v2 or peer-to-peer (NVP2P). The NVIDIA peer-to-peer protocol basically allows one GPU to read and write the memory of another GPU, provided that they are on a compliant platform (suitable PCIe bus topology). This protocol can be exploited by a third-party device to gain direct access to the GPU memory. NVIDIA provides an additional access method for third-party devices, the GPUDirect RDMA (or BAR1), officially introduced a public API to support it on Kepler-based Tesla and Quadro GPUs. APEnet+ is the first third-party device to provide GPUDirect hardware support. The current implementation of the GPU data transmission module is able to generate the read requests towards the GPU with a steady rate of one every 80 ns and implements a pre-fetch logic which attempts to hide the GPU response latency.



(left) APEnet+/GPUDirect RDMA. (right): APEnet+ GPU-to-GPU and Host-to-Host latency

3.12.2 RDMA on APENet+

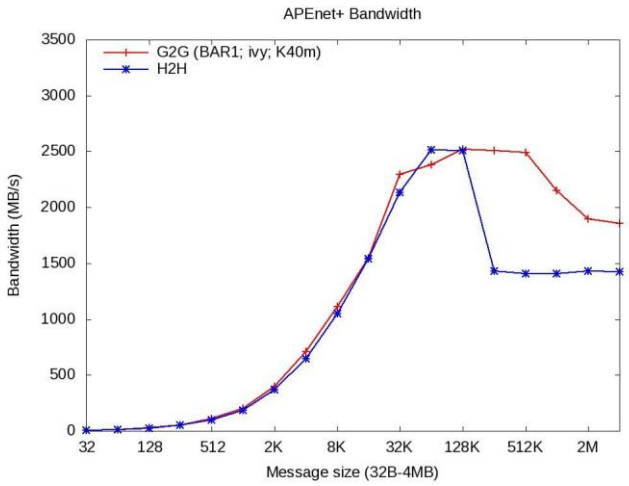
The Network Interface of APEnet+ implements RDMA, a reliable transport protocol in hardware supporting zero-copy networking with kernel bypass. The RDMA request is issued from an application running in user space to the local NIC and then carried over the network to the remote NIC without requiring any kernel involvement. For this to work, RDMA semantics imply three stages:

- a "registration" phase, when sender and receiver 'pin' the relevant areas of their (virtual) memory (making them ineligible for *swapping*) and resolve their physical addresses;
- a "rendezvous" phase, when sender and receiver pass to each other the address of the buffer to be transferred;
- the actual transfer phase, a "one-sided" operation initiated by the sender — a PUT semantics — or the receiver — a GET semantics — targeting the addresses registered in the first phase.

On the receive side the NIC needs to directly write the data to the destination buffer (DMA) requiring virtual to physical address translation. Buffer access (read/write) completion acknowledge is pushed into an *event queue* accessible to the application, that can pop it asynchronously.

3.12.2.1 RDMA task implementation and acceleration

APEnet+ implements the receiving side mechanism in two manners: a solely microcontroller based way; and a hardware assisted way using a Translation Lookaside Buffer (TLB) to accelerate the buffer search (BSRC) and address translation (V2P) tasks. Altera provides the 32bit general-purpose RISC processor core, the Nios II. The microcontroller firmware, tasked with such operations, takes to a total execution time of a search operation on Nios II is ~380 clock cycles. The hardware accelerated implementation [Ammendola: 2013c] includes in APEnet+ design two hardware blocks to accomplish the two different tasks.
The BSRC block is implemented as a small dual-port RAM memory used as a cache. The BSRC iteratively checks if the virtual address of an incoming packet is in the table. The V2P block has been designed integrating a Content Addressable Memory (CAM). The latency for an address translation has moved from 1900 ns to 124 ns with the TLB hardware. Peak bandwidth of a buffer transfer from host to host has improved from 1520 MB/s to 2500 MB/s for 128 KB message size; a transfer to GPU memory has improved from 1380 MB/s to 2500 MB/s in the range 64 KB ÷ 2 MB.



APEnet+ GPU-to-GPU and Host-to-Host bandwidth.

3.12.3 MPI on APENet+

MPI is the defacto standard communication API for parallel computing platforms. The MPI API and its semantics has been implemented in a number of library that support a variety of Network Interfaces. For APEnet+ we choose OpenMPI, as it is open source and portable. To add support for the APEnet+ interconnect we developed a component (apelink BTL, Byte Transfer Layer) that can be loaded by the low level framework of OpenMPI to interface the APEnet+ NIC for data transmission. *apelink BTL* relies on the RDMA API to implement two-sided communication and can be selected at launch time as alternative to ones usually used for inter-node communication on cluster installation, like *openib* and *tcp*.

3.13 ASIP design tools applied to computation and interconnection

The Euretile consortium's approach towards the design of ASIP is to use a retargetable tool-suite based on a processor description language. The tool-suite supports architectural exploration, the generation of a software development kit, and the generation of an RTL hardware implementation of the ASIP. TARGET's IP Designer tool-suite is used for this purpose. In 2011, TARGET developed new extensions and optimizations for its retargetable tool-suite.

These developments focussed on the following main topics:

· A new methodology has been defined to model memory, communication interfaces of ASIPs and synthesis of I/O interfaces. In particular, a new concept of "I/O modules" has been implemented, which enables the modular design of complex I/O interfaces by chaining elementary modules with orthogonal functionalities.

· New techniques have been developed and implemented to provide enhanced feedback to users of the retargetable tool-suite during the architectural optimisation process. Specifically, the tools can now analyse and report about connectivity issues within an ASIP architecture. The tools have been extended with capabilities to model multi-threaded architectures. This includes the generation of hardware support for storing the context of a thread upon a context switch, as well as C programming aspects. In particular "interleaved multi-threading" has been investigated. This allows for a zero-overhead context switch, as each pipeline stage can operate on a different register context.

A number of exploration have been jointly conducted by INFN and Target about the application of ASIP technologies to numerical kernels and APENet+ interconnect.

An efficient ASIP for LQCD has been designed using the IP Designer tool-suite. The ASIP has been named "VCFLX" (referring to Vector Complex Floating-point – Flexible"). Benchmarking shows that a single VCFLX ASIP can implement LQCD 33x faster than a 32-bit floating-point CPU and 4x faster than the mAgicV DSP. About 50 instances of the VCFLX ASIP can fit on a single Virtex-7 FPGA, thus speeding up LQCD even

further, provided that the FPGA's RapidIO fast communication channels can be used to achieve a sufficiently high communication bandwidth.

EURETILE's many-core platforms critically depend on efficient packet-based communication between tiles in the 3D network. Efficiency refers to both high bandwidth and low latency. Our observation for the HPC platform is that the current DNP implementation, which includes firmware running on a NIOS soft-core from Altera, offers high-bandwidth communication, but with a too high latency. To reduce the latency, a significant acceleration of networking functions (e.g. buffer search, virtual-to-physical address translation) is needed. Regarding the application of ASIP techniques to improvement to the APENet+ design several studies has been conducted. A preliminary evaluation of effects of introduction on RX RDMA path of a TLB (Translation Look aside Buffer) for virtual to physical address translation has been performed. A first version of a DNP ASIP has been modelled and optimised, using the IP Designer tool-suite. The basic architecture is a 32-bit microprocessor. Adding dedicated hardware and instructions to accelerate the buffer-search functionality for RDMA tasks has further optimised the architecture. As a result, the cycle count of typical buffer-search firmware programs has been reduced by more than 85% compared to the existing implementation on NIOS. Currently the clock frequency of the ASIP is still lower than that of NIOS. Nonetheless, buffer search already executes significantly faster on the ASIP compared to NIOS. Therefore, it has been further extended to efficiently implement the virtual memory management aspects of the inter-processor communication protocol. Results show a 3.6x performance improvement in the implementation of the physical address look-up function, which is the most critical function for virtual memory management.

3.14 Application benchmarks

We developed and used two principal sets of application benchmarks: 1- dynamic multi-media many-process Applications, 2- a many-process simulator of neural activity and plasticity representative of brain-simulation, with potential application to embedded robotics.

3.14.1 Dynamic Multi-Media Many-Process Applications

In order to evaluate the proposed design flow, a diverse set of applications has been developed using the previously proposed programming model. In the following, we summarize the individual applications by categorizing them into applications that are typically used in digital signal processing, multimedia applications, and other applications. Finally, describe an advanced picture-in-picture (PiP) software for embedded video processing systems. The benchmarks themselves are available online for download at http://www.dal.ethz.ch.

3.14.1.1 DSP/Data-flow Kernels

To demonstrate the capabilities of the EURETILE design flow, various applications have been developed that are typically used in digital signal processing (DSP), namely a distributed implementation of an N-point discrete Fourier transformation (DFT), an N-order IIR filter, and a distributed implementation of a matrix multiplication.

3.14.1.2 Multimedia Applications

Multi-Stage Video Processing Application: A multi-stage video-processing application has been implemented that decodes a motion-JPEG video stream and then applies a motion detection method to the decoded video stream. The MJPEG decoder can decode multiple video frames in parallel. The motion detection method is composed of a Gaussian blur, a gradient magnitude calculation using Sobel filters, and an optical flow motion analysis.

H.264 Codec: The H.264/MPEG-4 AVC (Advanced Video Codec) is one of the most widely used video coding standards in recent years. The considered implementation of the H.264 codec is based on code for HOPES [Jung, 2014], which has been provided by the Seoul National University and supports the baseline profile of the coding standard [Kwon, 2008]. The basic implementation of the encoder uses task division on the macroblock level. In particular, the set of functions is divided among five processes: Init, ME, Encode, Deblock, and VLC.

Ray Tracing: Providing a high degree of realism, ray tracing is expected to be implemented as a real-time rendering algorithm in the next generation of embedded many-tile systems. Ray tracing applications naturally consist of three logic parts. The first part of the application comprises the generation of the rays. The next part

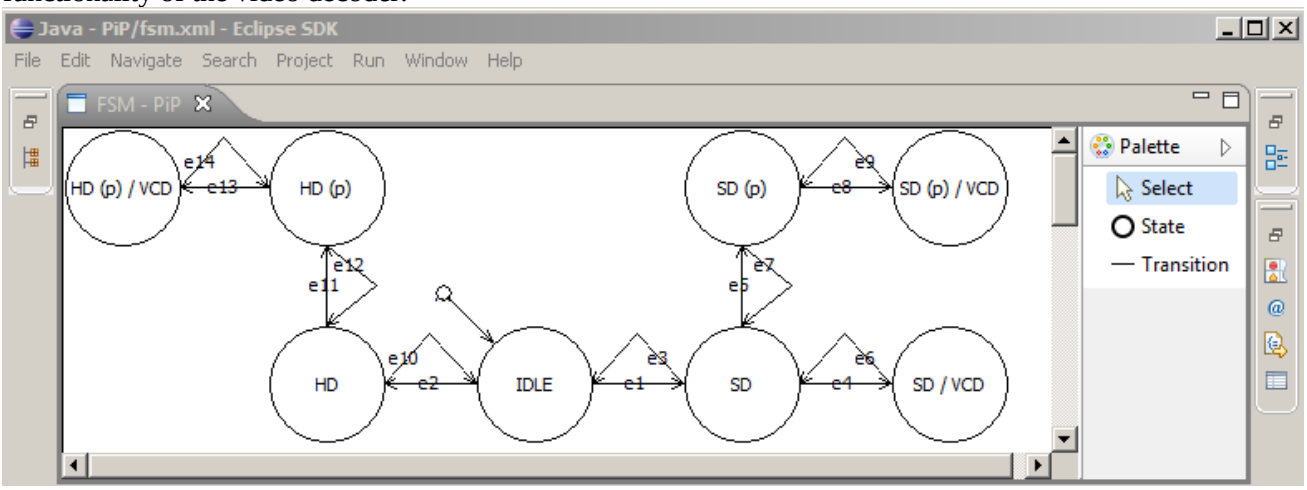
is the second and most computational intensive part of the application, the intersection of the rays with the scene, which is to be rendered. Lastly, the calculated values are aggregated and stored to an image file in the third part. In fact, this partition is used for the process network specification, as it allows us to neatly separate the intersection part, which we are interested in parallelizing as it has the biggest contribution to the application's total execution time.

3.14.1.3 Other Streaming Applications

Recursive array-sorting: The considered sorting algorithm is quicksort. In fact, as quicksort is based on recursion, it cannot be specified using conventional specification models of process networks. However, the application can be specified efficiently as an EPN. The top-level process network consists of three processes: Process "src" ("dest") generates (displays) the input (output) array, and process "sort" sorts the elements in ascending order. As the quicksort algorithm rec*u*rsively sorts the array, process "sort" can be replaced by a structural description, which divides the array into two smaller arrays that can be individually sorted.

3.14.1.4 Picture-In-Picture (PiP) Video Decoder

The proposed PiP video decoder is composed of eight scenarios and three different video decoder applications. The HD application processes high-definition, the SD application standard-definition, and the VCD application low-resolution video data. The software has two major execution modes, namely watching high-definition (scenario HD) or standard-definition videos (scenario SD). In addition, the user might want to pause the video or watch a preview of another video by activating the PiP mode (i.e., starting the VCD application). Due to resource restrictions, the user is only able to activate the PiP mode when the SD application is running or paused, or the HD application is paused. The figure shown below illustrates the finite state machine describing the functionality of the video decoder.



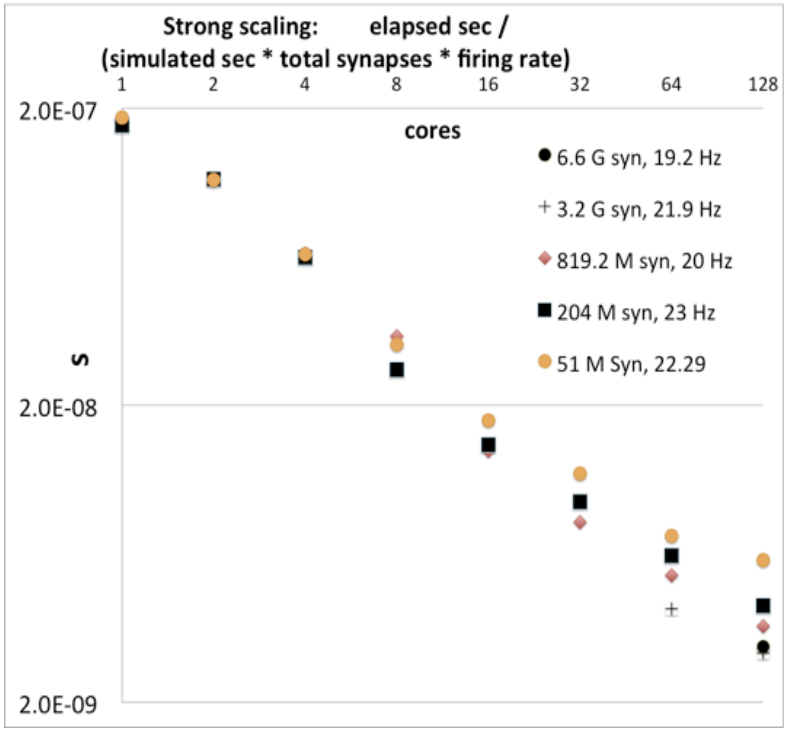
Finite state machine of the considered PiP benchmark.

3.14.2 Distributed simulation of polychronous and plastic spiking neural networks

INFN developed a natively distributed mini-application benchmark representative of plastic spiking neural network simulators (DPSNN_STDP) (see [Paolucci, 2013b]). It generates complex inter-process traffic patterns that vary with time and individual process. Processes describe synapses in input to cluster of neurons, and the resulting topology of interconnection between software processes is non embarassingly parallel. It can be used to measure performances of existing computing platforms and to drive the development of future parallel/distributed computing systems dedicated to the simulation of plastic spiking networks. The mini-application is designed to generate spiking behaviors and synaptic connectivity that do not change when the number of hardware processing nodes is varied, simplifying the quantitative study of scalability on commodity and custom architectures. The mini-application has been designed to be easily interfaced with standard and custom software and hardware communication interfaces. It has been designed from its foundation to be natively distributed and parallel, and should not pose major obstacles against distribution and parallelization on several platforms.

3.14.2.1 Validation of functionality and scaling of the many-process neuro-synaptic simulator

The many-process network of C++ processes describing the simulation of neural activity and synaptic plasticity has been coded to be compatible with both the EURETILE DAL over DNAOS environment and with a standard MPI over Linux environment. A first necessary validation step was the reproduction of the neural spiking activity and of the synaptic plasticity. A second validation step was the analysis of the strong and weak scaling and the profiling of the computational/communication components of the many-process code. In a first test, we used the benchmark on QuoNG (varying the number of used physical cores from 1 to 128), through the commodity network. Bidimensional grids of columns composed of Izhikevich neurons projected synapses locally and toward first, second and third neighboring columns. The size of the simulated network varied from 6.6 Giga synapses down to 200 K synapses. The code demonstrated to be fast and scalable: 10 wall clock seconds were required to simulate one second of activity and plasticity (per Hertz of average firing rate) of a network composed by 3.2 G synapses running on 128 hardware cores clocked @ 2.4 GHz. For a full description see [Paolucci, 2013b].



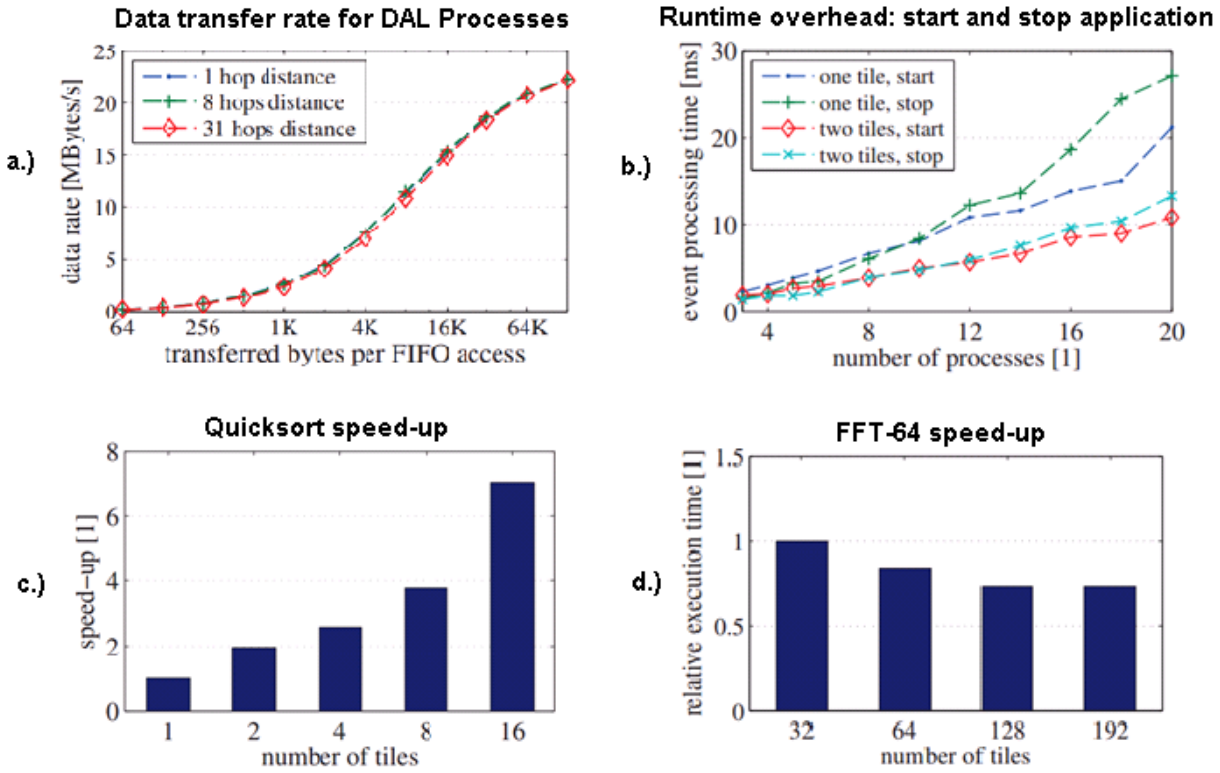
Strong scaling of the DPSNN-STDP mini-app benchmark. For an ideal scaling the execution time should grow proportionally to the number of synapses and to the firing rate, and should reduce proportionally to the number of cores applied to the execution.

The many-process code demonstrated its ability to reproduce the spiking activity and neural plasticity produced by a open-access, open-source sequential single-process code made available by Izhikevich. In particular, we adopted a coding style that allowed to reproduce identical neuro-synaptic activity independently from the number of processes used for the parallelization.

3.15 Experimental results on VEP simulated platform

The VEP has been used to assess some essential non-functional characteristics of the EURETILE software tool-chain. Details of the experiments can be found in [Schor, 2014a]. The figure shown below presents some remarkable results. Part (a) shows the aggregated data transfer rate for process mappings on the VEP when a data token of increasing size passes via 1, 8 and 31 DNP hops. The observed peak data rate over all configurations is 22.3 MBytes/s. Part (b) shows the time that the runtime environment takes to start and stop an increasing number of processes, considering the cases when they are all on the same tile and on different tiles. The time to start and stop the application increases linearly with the number of processes. If the processes are distributed between two tiles, the time to start and stop the application is almost the half of the time for one tile.

Part (c) shows the speed-up achieved when running the DAL quicksort application (see Section 3.15.1.3) on a different number of tiles. The maximum speedup that can be achieved is 7.01x whereby the additional costs to split an array into sub-arrays prevent an even higher speed-up. Finally, part (d) shows the execution time of the FFT-64 application (see Section 3.15.1.1) on 64, 128, or 192 tiles when compared to a reference run on 32 tiles. The granularity of the application is too small to absorb the overhead imposed by the process and channel management, but the execution time is anyway reduced by a factor of 1.36 when going from 32 to 192 processes. This demonstrates low communication latency and low process management overhead of the EURETILE system.



EURETILE Tool-chain characterization using the VEP.

3.16 Experimental results on QUonG hardware platform

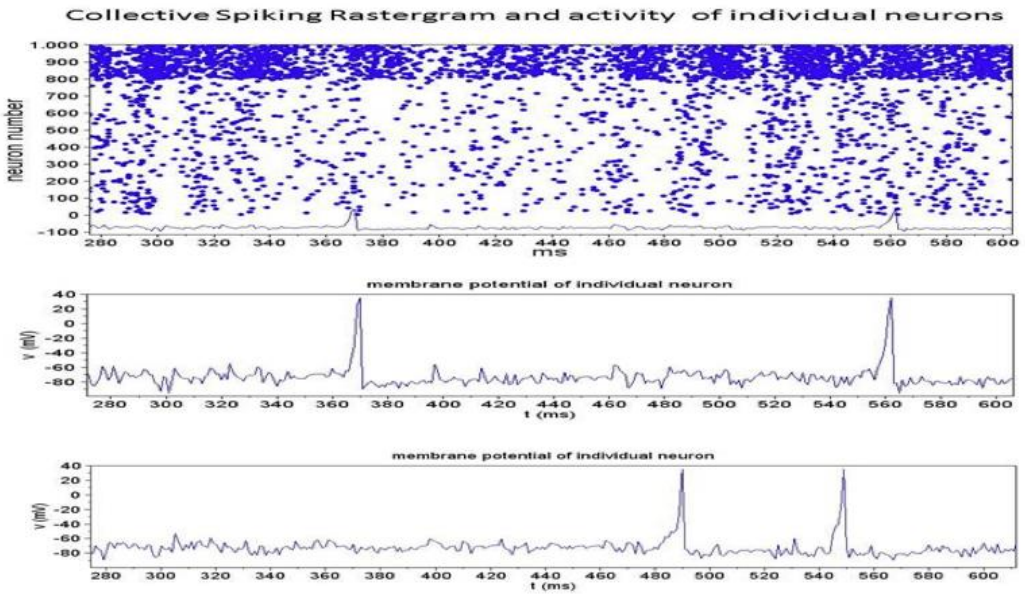
3.16.1 DNA-OS on QUonG

DNA-OS has been ported on the QUonG hardware platform, including all the device features required to generate, start, and run applications. For instance, PCIe driver, DNP driver, IGB driver for the ETHERNET support (in order to start an synchronize all QUonGs) has been validated, as well as support for multi-tile and multi-core. We are today facing the memory limit with a 32 bits version of DNA-OS in case of complex applications which require more than 4 GB (232 Bytes) of memory. A boot sequence for x86 starts each processor. A script downloads all binaries using the IGB driver. Then we start a synchronization process of all QUonG. Finally, we run the application. All this is today fully automatic and does not require any action from the designer.

3.16.2 Executing the DAL version of the neural DPSNN benchmark on QUonG

The DPSNN application has been ported to the DAL programming environment for execution on the EURETILE hardware platform, using DNA-OS and the software tool-chain for the binary generation. In the DAL model of the DPSNN application several parameters may be tuned at each run. For instance, it can be varied the number of processes, the number of neurons per process, the number of synapses per neuron, the kind of interconnection between neurons (network topology), the kind of external thalamic input, etc. A QUonG

cluster of 16 machines is available, for a total of 128 hyperthreaded hardware cores. The designer can provide as well the number of QUonG to be used and how many DAL processes (application processes) are running on each QUonG.



The many-process DPSNN-STDP produces detailed simulation of neural activity and synaptic plasticity. The results of the simulation are independent from the number of C++ process used for the parallelization and from the software tool-chain (e.g. EURETILE DAL on DNA-OS or MPI over Linux). These features simplifie the study of the scaling properties on many-tile execution platforms.

The results produced by a run of the DAL model of the DPSNN benchmark can be plotted in rastergram, a diagram used to describe the spiking activity of a neural network. In the "rastergram" the horizontal axis is the simulation time, the vertical axis is the identifier of individual neurons. Each dot in the rastergram represents a spiking event. In detail, the rastergram of the first two seconds of simulated activity can be considered representative of the evolution of the neural network, both for the dynamic behaviour and for the plastic behaviour. In fact, during each second of simulation the neural dynamic is responsible of the network evolution, while the synaptic plasticity is applied at the end of each second and his effects are visible on the successive second of activity. The second set of results, reported by following sections is about 1- the comparison of the efficiency of DAL on DNA-OS execution vs a "standard" environment (MPI on Linux) and 2- about the scaling on a full QUonG system (16 server with APENet+ interconnect).

3.16.2.1 Efficiency of the EURETILE tool-chain (DAL on DNA-OS) vs "standard" (MPI on Linux)

The EURETILE tool chain provides a set of added values to the developer of a many-process application to be executed on many-tile execution platform. It is important to demonstrate that it is possible to get high-efficiency execution using our tool-chain. To this purpose we measured the execution time and scaling behaviour of the simulation of a neural network on a single multi-core server, for a varying number of DAL (and MPI processes). On a single QUonG multi-core server, inter-process communication channels can be implemented over shared-memory, and each process can be executed by a dedicated (hyperthread) hardware core. The following table demonstrated the efficiency of our software tool-chain, compared with a standard MPI over Linux environment. We run a relatively small neural network, 13.1 Million synapses, active at 12 Hz mean rate, projected by a mix composed of 80% Izhikevich excitatory RS and 20% FS neurons, with a mean of 400 synapses per neuron. Long Term Potentiation/Depression of the synapses is also included in the simulation. Such a "small" network challenges the capability to be efficiently distributed over a high number of processes and hardware resources. The single QUonG server was equipped with (hyperthreaded) cores. As expected, best performances were obtained when allocating a software process per hardware core. Distributing the simulation over a greater number of software processes (overcommiting the available hardware cores) led to slower execution.

number of software processes	1	2	4	8	16	32	64
MPI on Linux (exec seconds)	46.9	27.0	11.8	9.9	8.3	11.0	24.4
DAL on DNA-OS (exec seconds)	46.1	23.0	17.4	10.9	9.6	10.3	-

This table demonstrates that the EURETILE DAL on DNA-OS tool chain is able to get efficiencies comparable to "standard" (MPI on Linux) when executing on an Intel server. The simulation of three seconds of activity of a relatively "small" neural network, composed by 13.1 Million synapses has been executed distributing the problem over an increasing number of software processes. Best execution is obtained with one process per hardware core.

3.16.2.2 DAL on DNA-OS scaling to the full QUonG hardware platform

Another interesting point to demonstrate was the ability of the EURETILE software tool-chain to manage the full QUonG platform, exploiting the APEnet+ interconnect for communications among different servers and shared memories for communications among software processes mapped on the same tile. The following table report the execution time of the simulation of three seconds of activity of a network composed by 52.4 Million synapses, varying the number of software processes mapped on 16 QUonQ servers.

number of software processes (mapped on 16 QUonG servers, APENet+ interconnect)	16	32	64
DAL on DNA-OS (exec seconds)	26.3	21.9	14.0

It is interesting to note that the network size is relatively small and that a 16 node QUonG server would greatly benefit from 64 bit support, to run configurations of adequate size.

3.16.3 APENet+ RDMA applied to DPSNN simulation

In order to test the APEnet+ architecture on QUonG nodes running Linux OS and use the NIC at its best, we ran a DPSNN version where the exchange of the axonal spikes is performed using APEnet+ RDMA API. As this implementation uses MPI for the initialization and DIM exchange phases and RDMA only for the axonal spikes exchange phase, it is actually configured as alternative to the use of the "pure MPI" implementation and we refer to it as ``hybrid MPI/RDMA implementation". With this code we were able to run on QUonG ranging the number of software processes from 16 to 64, up to 16 QUonG nodes, with configurations composed by up to 209 Million synapses. The results are lined up with those obtained in the DAL on DNA-OS execution, but suffer higher variability.

3.16.4 Demonstration of MPI for APENet+

DPSNN can run on QUonG also using the MPI library supporting the apelink btl, i.e. using APEnet+ for internode communication. The advantage is that the code is pure MPI version, but APEnet+ can be selected for internode communication as well as Infiniband or Ethernet, meaning that the code itself contains no lines for any specific network interface and all the low level complexity of the communication is hidden in the MPI library itself. Performance measured with DPSNN shows an overhead due to the presence of the MPI library layers adapted to the APEnet+ peculiarities and the presence of MPI runtime, thus the results we obtain using the RDMA/MPI hybrid code described in the previous section are more promising, at the cost of inserting RDMA communication handling in the application code.

3.17 Discussion of limitations and possible related future work

Finally, we discuss possible limitations of the proposed approach, based on 1- experience gained by execution on the QUonG and VEP platform and 2- conceptual limitations understood thanks to the practical integration

work above described and conceptual analysis of possible further integration efforts. We also propose related future work.

3.17.1 Limitations of the Programming Model and its Scalability

The current limitation of the programming model is its static specification at design-time. In fact, the original system configuration can only be adapted if the corresponding change is described in the original system specification. For instance, as all possible use cases of the system are specified as a finite state machine, the current programming model does not allow the modification of the original system configuration, e.g., by installing additional applications. If unknown applications can be installed after deployment, system analysis and mapping decisions must be performed at run-time, requiring fast and accurate heuristics, which do not cause long response times. While low communication latency and low process management overhead have been demonstrated, the speed-up of a DAL application with respect to the number of available tiles is still limited by the intrinsic scalability of the application. In fact, the benchmarks developed in the context of EURETILE have shown two scalability issues of process network topologies. On the one hand, some of these applications are limited by one, or few processes that are most computing intensive. On the other hand, applications like the proposed neuro-synaptic simulator, in spite of their complex global network topology, and time-variant intra-process traffic patterns can be designed using native distributed coding and network topology styles. For such applications, natively designed for scalability, each DAL process interfaces a controllable number of communication channels, even for global process network topologies and traffic patterns, and no single process dominates from a computational point. In such cases, a kind of weak scaling exists also for the communication infrastructure, and it could be conceivable to exploit regularities in the topology to parallelize also the generation of "locally" flattened XMLs and, similarly, a parallelized and distributed APES generation tool could generate local binaries reducing the executable creation and bootstrap time.

3.17.2 Limitations of the Fault Management Approach

The limitations on the current fault tolerance approach stem directly from the proposed design. In its current form, the LO|FA|MO fault detection mechanism could inform the specialized fault tolerance components (i.e., the "splittermanager" and the "join" processes) of any faults in the hardware components. In addition, both splittermanager and the join processes are able to autonomously detect timing faults, but not value faults. Therefore, the design makes an implicit assumption that all faults will either be detected by the LO|FA|MO detection system or will be observed as a timing fault. However, this assumption may not always be true. Specifically, a fault may cause a part of the process network to transmit incorrect values, but none of the hardware and timing checks may show any anomaly in the system. Detection of such value faults at the fault tolerance components may require incorporation of statistical, probabilistic and heuristic algorithms, which may additionally need to be tuned to the application that is being made fault tolerant. Needless to say, such algorithms may impose significant memory and computation costs on the hardware platform, and may require recomputing at least a few, if not all, mapping options. In the framework of EURETILE, the fault-awareness and detection hardware abilities of LO|FA|MO have been tested on both the VEP simulator and QUonG hardware platforms, but systemic reaction has been investigated only conceptually, and not proven neither on the hardware nor on the simulated platform. LO|FA|MO detection mechanism on the other hand is completely integrated with DNP logic, while it relies on the host diagnostic tools to check for host side faults, like memory faults. On QUonG this can be easily done when running Linux OS as it provides a number of such tools. On DNA-OS we do not have specified how this kind of integration can be done.

3.17.3 Limitations of the Runtime System

Due to its 32-bits support, various memory limits are currently faced in DNA-OS. For instance, if one tile of the QUonG platform is used, the maximum number of processes is limited to 32. On the other hand, if multiple tiles are used, the maximum number of processes per tile is limited by the address space that is currently limited to 232. Another limitation of the current tool chain is the size of the input files and the size of the resulting low-level code in terms of number of function calls, which comes from the fact that only flattened XML files are used as input. For instance, when running the DPSNN application on multiple tiles of the QUonG platform, the XML file had about 500'000 lines of code so that parsing and compiling the application took about one hour. Similar compiler limitations were hit in the VEP when compiling the FFT-64 application for 192 tiles. In order to tackle these challenges, different parsing and code generation techniques have been considered in the last

phase of the project so that these limitations could be decreased drastically. In perspective, parallel and distributed generation techniques should be considered.

3.17.4 Limitations of the VEP Simulator

The capability and scalability of the VEP simulator is mainly limited by the used IRISC model version. With the fastest ISS based on Synopsys WARP technology (speed of approx. 160 MIPS in stand-alone configuration), the VEP simulator is currently limited to be compiled in 32-bits, which sets an overall simulation memory limit of 4 GBytes. Simulations running bare metal applications can fit up to maximum 32 tiles and simulations running DNA-OS and DAL applications can fit up to maximum 24 tiles on a single simulation host. There is also an IRISC model that is compatible with 64-bits compilation and can be used to simulate up to 1000 tiles on a single host. However, its speed of approx. 25 MIPS is considerable less than the fastest variant, thus decreasing the overall simulation speed. Architectural features of the IRISC also limit the capability to execute some types of applications in an efficient way. For instance, no floating-point hardware unit is provided so floating-point data types have to be emulated in software by the compiler, and have a considerable impact on application runtime. Also, both the IRISC core and the VEP DNP are limited to a 32-bits address space which in turn limits the execution of applications with huge memory requirements on the VEP. In addition, the maximum data transfer rate of DAL applications is limited by buffering limits of the DNP.

4 Consolidated and Potential Impact

4.1 Industrial Exploitation and Impact of ASIP Design Tools

The new technologies of co-design of hardware architecture and programming tools of Application-Specific Processors that have been developed by TARGET and are being transferred from EURETILE will be an integral part of next-generation products of the leading EDA tools vendor Synopsys.

Euretile's software tools for the design and programming of application-specific processors (ASIPs) were developed by project partner Target Compiler Technologies (TARGET) in a cooperation with INFN. Since its incorporation in 1996, TARGET has continuously invested in R&D, partly based on collaborations within EC and nationally funded projects, which resulted in the company becoming the world's leading vendor of ASIP design tools. Today, TARGET's IP Designer tool-suite is in production use by many of the world's tier-1 and tier-2 semiconductor and system companies, including public references like Broadcom, Conexant, Dialog Semiconductor, Freescale Semiconductor, GN ReSound, Huawei, NXP Semiconductors, Olympus, ON Semiconductor, Sanyo, Silicon Labs, STMicroelectronics, and Texas Instruments. We estimate that more than 150 unique chips are in the market containing IP Designer-made ASIPs.

Already before the completion of the Euretile project, TARGET initiated the commercial exploitation of certain technologies developed in the project. The current exploitation status is as follows:

- After an initial trial period by key customers, the new integrated development environment (IDE) named "ChessDE" has been released as the standard IDE in the commercial IP Designer product.
- The new multicore IDE named "ChessMP" has been tested by key customers and is currently being rolled out commercially.
- After an initial trial period by key customers, the new methodology and tools for modeling the behavior of the ASIP's datapath operators and program control unit, based on bit-accurate C code (so-called "PDG" language), have been released as the standard method in the commercial IP Designer product.
- The subsequent extension of the behavioral modeling methodology in the PDG language towards modeling of I/O interfaces has been tested by key customers and is currently being rolled out commercially. Additionally TARGET is extending the initial set of example I/O interface models that were developed as demonstrators in the project, into a complete library of I/O interface models that will be shipped to commercial IP Designer users.

A continuation of this exploitation work is expected in a period of 2-3 years following the completion of Eurtile.

On February 7, 2014, TARGET was successfully acquired by Synopsys, Inc., the world's leading vendor of design tools for the semiconductor industry, headquartered in Mountain View, California. The acquisition underscores Synopsys' strong belief in the future of ASIP tools and in TARGET's IP Designer product in particular. The following statements are quoted from Synopsys' press release announcing the acquisition [Synopsys, 2014b]:

"Today's SoCs rely more on heterogeneous multi-core architectures. Designers are turning to ASIPs to implement their unique data plane and digital signal processing requirements. (…) Target's leading IP Designer and MP Designer software tools perfectly complement Synopsys' offerings for ASIP developers, enabling design teams to develop ASIPs that meet their performance, power, and flexibility requirements more efficiently and with less risk. (…) The acquisition of Target strengthens Synopsys' existing ASIP tools portfolio while bringing a world-class team of ASIP experts into the company."

Nine months after the acquisition, the following can be said:

- Synopsys is combining TARGET's IP Designer and MP Designer products with its pre-existing product called "Processor Designer", in order to build an even stronger next-generation tool-suite for ASIP-based system design. The new technologies that have been or are being transferred from Euretile (as described above) will be an integral part of this next-generation product.
- Synopsys has kept TARGET's entire R&D team based in Leuven (Belgium), as well as the former Processor Designer R&D team based in Leuven (Belgium), Aachen (Germany) and Noida (India),

completely intact. As a matter of fact, these two teams have been merged without any reduction of personnel, into a new sizable R&D group for ASIP-based system design technologies, lead by Gert Goossens (TARGET's former CEO and now R&D Group Director in Synopsys). This ASIP R&D group is also cooperating with Synopsys' R&D group for DSP platforms based in Eindhoven (Netherlands). As a result, the Leuven, Aachen and Eindhoven sites now form the center of gravity for ASIP and DSP technologies in the worldwide Synopsys organization. They also form one of the strongest concentrations of Synopsys R&D engineers in system-level methodologies outside of the US.

- Synopsys continues to invest in new R&D activities to expand the functionality of its ASIP-based system design tools, bootstrapping on TARGET's R&D activities in Euretile. In particular, new related focus areas include compilation support for high code-density and for additional programming languages such as C++ and OpenCL (used for parallel programming). Also we will explore cooperation options with Euretile partner RWTH Aachen, which previously already had an active relationship with Synopsys.

4.2 Industrial Exploitation and Impact of Simulation and Debugging Technologies

The simulation and parallel debugging technologies created by RWTH to support the development of the EURETILE system have been transferred to third-party organizations via different means, and are being used in academia and industry in different contexts. During early project stages, the hybrid simulation (HySim) technology and the parSC SystemC kernel were used by Huawei Technologies Co. Ltd. and Synopsys Inc., respectively. HySim and parSC were adapted to their needs and evaluated in an industrial setting for a period of approx. 1 year. Both technology transfer projects ended up with satisfaction from the partners and yielded the expected result of higher simulation speed for electronic system-level (ESL) simulation.
Besides these activities, RWTH acts as an external services provider of simulation technologies for the Chist-Era GEMSCLAIM (GreenEr Mobile System by Cross LAyer Integrated energy Management, http://www.gemsclaim.eu/) project. The SCope SystemC kernel currently supports the GEMSCLAIM system simulator and the software development activities performed on it, which include the development of an energy-aware optimizing and parallelizing compiler (OpenMP+) and an energy efficient OS kernel. Currently, the University of Innsbruck (AT), Queen's University Belfast (IE) and the Polytechnic University of Timisoara (RO) are users of the SCope technology through the GEMSCLAIM simulator.
Finally, cooperation plans are being studied with other projects and organizations that have shown interest for our simulation and debugging technologies. In particular, RWTH is currently in conversations with Synopsys Inc. and Sigma Designs Inc. regarding technology transfer agreements. Additionally, HySim, parSC, SCope, WSDB and SWAT are being or planned to be used by other internal RWTH projects as well as in several PhD and Master theses.

4.3 Exploitation of the neuro-synaptic DPSNN-STDP simulator and of the QUonG platform in the CORTICONIC FET Project

Starting from October 2014, the techniques of distributed simulation of neural activity and synaptic plasticity developed by INFN in EURETILE will be adapted and improved to support the simulations planned by the CORTICONIC project, in cooperation with ISS (Istituto Superiore di Sanità). The CORTICONIC projects is about "Computations and Organization of Retes Through the Interaction of Computational, Optical and Neurophysiological Investigations of the Cerebral cortex" and is funded by the FET Grant Agreement 600806. The DPSNN-STDP application will be the starting point for the development and the QUonG platform will be used to perform the large scale simulations required by the CORTICONIC project.

4.4 Public release of DAL many-process dynamic development environment

The distributed application layer (DAL) framework [Schor, 2012], which has been developed by ETH Zurich as part of the EURETILE project, significantly affected the activities of the Computer Engineering Group at ETH Zurich in the past few years. Since 2013, the DAL framework has been made available to interested industrial and academic institutions and since summer 2014, the DAL framework is public available for download at http://www.dal.ethz.ch. Since then, the DAL framework has been successfully applied in several academic and industrial projects, both as front-end and back-end. In the following, we describe the most important activities.

The DAL framework is freely available for download at http://www.dal.ethz.ch. Besides the ability to download the DAL framework, various tutorials are provided to help the interested user to use the framework. Furthermore, a set of advanced embedded system benchmarks are available for download including a distributed implementation of a ray-tracing algorithm, a video-processing application, and an H-264 decoder and encoder. Finally, DALipse, the graphical software development environment for the DAL framework, is available for download at the website.
At ETH Zurich, two separate projects have adopted the DAL framework. The EU FP 7 project CERTAINTY (http://www.certainty-project.eu) is using the DAL framework as a baseline to develop their own software management framework for mixed-criticality systems. The UltrasoundToGo project (http://www.nano-tera.ch/projects/359.php), which is founded by the Swiss National Science Foundation, uses the DAL framework to prototype software for embedded medical devices. In particular, the DAL framework has recently been used to demonstrate the ability to use process networks in order to develop complex medical applications [Tretter, 2014]. Furthermore, the DAL framework has been the basis for various PhD, Master, and Bachelor theses at ETH Zurich.
Besides internal projects, the DAL framework has been applied by other academic and industrial users to prototype embedded streaming applications. For instance, the DAL framework has been used by Huawei Technologies Co. Ltd. and by Broadcom Corporation for prototyping new design methodologies. In addition, the following academic institutions have been using the DAL framework for either teaching or research activities: the McGill University, the Halmstad University, the University of Amsterdam, and the University of Oulu.
Among these collaborations, the integration of the DAL framework into the Open RVC CAL (ORCC) compiler is the most elaborative one. The CAL actor language is a popular dataflow language to specify complex multi-processor systems. The ORCC compiler is available to convert CAL applications into conventional software languages such as C, C++, and Verilog. In order to automatically generate code for the DAL framework, the University of Oulu recently added a code generator back-end to the ORCC compiler [Boutellier, 2014] as part of the Finnish Academy of Science project "Dataflow oriented automated design toolchain (DORADO)", which is carried out between 2011 and 2015 (http://www.cse.oulu.fi/CMV/Research/DORADOproject). The current plan is to extend this interface during an additional research project that will be carried out between 2015 and 2019.

4.5 Public Release and Industrial Impact foresees by TIMA

The capabilities of DNA-OS have been proved in this project, targeting both embedded and HPC domains. Both targeted architectures are really specific and can not be exploited directly. Nevertheless, the port of DNA-OS for x86, its multi-core and multi-tile supports, IGB driver and some other features could be re-used in the context of industrial transfer to third-party, as specific service developments, or all-in-one for a multi-core or multi-tile architecture. This work opens new industrial and research activities, as DNA-OS is delivered with our own simulation environment (ARM-based architecture for embedded domain). For instance, a work with the Kalray company in Grenoble has started one month ago to port DNA-OS on the Kalray processor.

The results obtained concerning task migration are new and validated only by simulation. This could have an industrial impact when it runs on the hardware, following publications of these results. The proposed methodology, that does not require any modification in the OS (even for a light OS without virtual memory or dynamic loading support) is an interesting innovation.

The port of DNA-OS for x86 is freely available and could be used with our own simulation environment also freely available. This could be used by any other designers.
The task migration methodolgy will be published to be used. But all the developments are quite specific and depend on the specification, and on the architecture (memory map, I/O, drivers, …) making this difficult to be used as it is.

This work opens perspectives in the HPC domain. DNA-OS has been developed for the embedded domain and research activities. The work done in the EURETILE project and the results highlight that there is no restriction in the use of DNA-OS, and the HPC domain could be targeted. The added value of task migration for critical tasks (load balancing or thermal management) is of prime interest in this HPC domain. Contributions to H2020 project could be a perspective.

4.6 From EURETILE HPC to High-Energy Physics real-time applications

The GPGPU computational paradigm, i.e. General-purpose computing on graphics processing units, is nowadays well estabilished in the High Performace Computing arena: several GPU accelerated clusters are present in the highest ranks of top500 list since few years. Virtually any Computational Physics simulation application, such as Lattice Quantum Chromo-Dynamics or Fluid Dynamics, have been adapted to take advantage of GPUs processing power exploiting their fine grained parallelism and show significant speedups in their execution times. Similar efforts are ongoing in several fields of Experimental Physics, ranging from Radio Astronomy to High Energy Physics. These contexts are often characterized by real-time constraints in processing data streams coming from experimental apparatuses.
GPUs show a stable processing latency once data are available in their own internal memories, so a deterministic latency data transport mechanism becomes crucial when implementing a GPGPU system with real-time constraints. The NaNet dissemination activity aimed at designing a Network Interface Card (NIC) implementing such deterministic latency data transport of experimental data to processor or GPU memories. NaNet reuse several IPs developed for the APEnet+ board in the framework of the EURETILE project, retailoring the design to integrate it in experimental setups, i.e. adding support for several standard and custom link technologies and developing dedicated modules to ensure the real-time characterization of the system. The result is a modular design of a low-latency PCIe Gen2 X8 RDMA NIC supporting standard 1GbE (1000BASE-T) and 10GbE (10Base-R), besides custom 34 Gbps APElink [Ammendola, 2013a] and 2.5 Gbps deterministic latency optical KM3link [Aloisio, 2011] links. The design includes a network stack protocol offload engine yielding a very stable communication latency, a feature making NaNet suitable for use in real-time contexts; NaNet GPUDirect P2P/RDMA capability, inherited from the APEnet+ design, extends its realtime-ness into the world of GPGPU heterogeneous computing. NaNet has been employed, with different design configurations and physical device implementations, in two different High Energy Physics (HEP) experiments: the NA62 experiment at CERN and the KM3Net underwater neutrino telescope.

The NA62 experiment at CERN aims at measuring the Branching Ratio of the ultra-rare decay of the charged Kaon into a pion and a neutrino-antineutrino pair. The NA62 goal is to collect ~100 events with a signal to background ratio 10:1, using a novel technique with a high-energy (75~GeV) unseparated hadron beam decaying in flight. The trigger system plays a crucial role in any HEP experiment by deciding whether a particular event observed in a detector should be recorded or not, based on limited and partial information.
Since every experiment features a limited amount of network bandwidth and storage for data acquisition, the use of real-time selections is fundamental to make the experiment affordable maintaining at the same time its discovery potential. Such selections reject uninteresting events only, and therefore reduce selectively data throughput.
In the NA62 experiment, to manage the high-rate data stream due to a ~10 MHz rate of particle decays illuminating the detectors, the trigger system is designed as a set of three trigger levels reducing this rate by three orders of magnitude. The low-level trigger (L0), implemented in hardware by means of FPGAs on the readout boards, reduces the stream bandwidth by a factor 10 and is a synchronous real-time system: a decision whether data on readout board buffers are to be sent to higher levels or not has to be made within 1 ms to avoid loss of data. The upper trigger levels (L1 and L2) are software-implemented on a commodity PC farm for further reconstruction and event building. In the baseline implementation, the FPGAs on the readout boards compute simple trigger primitives on the fly, such as hit multiplicities and rough hit patterns, which are then timestamped and sent to a central trigger processor for matching and trigger decision.
A pilot project within NA62 is investigating the possibility of using a GPGPU system as L0 trigger processor (GL0TP): this system processes unfiltered data from detectors, exploiting GPU computing power to implement more selective trigger algorithms. The first prototype of the GL0TP has recently been deployed in the experiment. The protoype integrates NaNet-1, a version of NaNet design configured to receive data from readout boards over 1GbE link with UDP data protocol and implemented on Altera Stratix IV development board. The GL0TP will operate in parasitic mode with respect to the main L0 trigger processor and, at least in the initial phases of the study, will process data from only one detector (the RICH - Ring Imaging CHerenkov detector). Altough the single 1GbE link of NaNet-1 has not enough bandwidth to cope with the bandwidth requirements of the experiment, in the order of tens of Gbps, we are using this reduced-bandwidth setup to assess the feasibility of our approach and to collect latency, bandwidth and throughput measurements that are driving the development of the multi-port 10GbE version of the board (NaNet-10). Preliminary results of this activity, available in [twepp2014-ArXiV], show that the NaNet design fits with the real-time requirements of the system,

yielding low and stable communication latencies between readout boards and GPU internal memory, making possible the usage of a GPGPU system as low level trigger for the NA62 experiment.
NaNet3 represents the customization of the NaNet design for KM3Net, an underwater experimental apparatus for the detection of high energy neutrinos in the TeV/PeV range based on the Cherenkov technique. The detector measures the visible Cherenkov photons induced by charged particles propagating in sea water at speed larger than that of light in the medium, and consists of a 3D array of photomultipliers (PMT). The charged particle track can be reconstructed measuring the time of arrival of the Cherenkov photons on the PMTs, whose positions is continuosly monitored with a precision better than 40 cm.
The KM3Net detection unit is called Tower and consists of 14 floors vertically spaced 20 meters apart. The floor arms are about 8 m long and support 6 glass spheres called Optical Modules (OM): 2 OMs are located at each floor end and 2 OMs in the middle of the floor; each OM contains one 10 inches PMT and the front-end electronics needed to digitize the PMT signal, format and transmit the data. Each floor hosts also two hydrophones, used to reconstruct in real-time the OM position and oceanographic instrumentation to monitor site conditions relevant for the detector. All data produced by OMs, hydrophones, and instruments, are collected by an electronic board contained in a vessel at the centre of the floor; this board, called Floor Control Module (FCM) manages the communication between the on-shore laboratory and the underwater devices, also distributing the timing information and signals. Timing resolution is fundamental in track reconstruction, i.e. pointing accuracy in reconstructing the source position in the sky. An overall time resolution of about 3 ns yields an angular resolution of 0.1 degrees for neutrino with energies greater than 1 TeV.
This requirement implies that the readout electronics, which is spatially distributed, have a common timing with a known delay with respect to a fixed reference. The described constraints hinted to the choice of a synchronous link protocol which embeds clock and data with a deterministic latency; due to the distance between the apparatus and shoreland, the transmission medium is forced to be an optical fiber.
Data produced by the OMs, the hydrophones and other devices in a single floor are collected by the FCM board, packed together and transmitted through a optical bidirectional point-to-point link to one port of a NaNet3 board hosted in a server in the on-shore laboratory. A single NaNet3 board manages four optical links, and hence four floors, sending GPS timing data and slow control commands to off-shore devices and receiving experimental and devices slow control data from them. For this particular NaNet application we implemented a synchronous protocol with deterministic latency at link physical level in order to ensure the correct event timestamping, along with a data level offload module handling the Time Division Multiplexing protocol adopted for the data transport. NaNet3 is implemented on a Terasic DE5-Net board, which is based on a Altera Stratix V GX FPGA device with straight connections to four external 10G SFP+ modules and a PCIe x8 edge connector, fully supporting the four optical KM3link channels and PCIe Gen2 X8 I/O interface of the design. The GPUDirect P2P/RDMA module is also included in the NaNet3 design, paving the way for the already started development of a GPU-based trigger for the experiment. At the present stage of development, a single optical KM3link has been thoroughly tested, with 48 hours of continuous error-free data acquisition of OM data; slow control functionalities has been tested as well. Further details on the activities carried on until now are reported in [twepp2014-ArXiV]. The deployment of the first KM3Net Tower is expected within November 2014, NaNet3 will be used to perform data acquisition and off-shore devices monitoring and control tasks since first day of operation.

4.7 EURETILE Exploitation via TETRACOM technology transfer project

A more general, yet important, path to technology transfer is enabled via the TETRACOM FP7 project (www.tetracom.eu). TETRACOM, coordinated by Rainer Leupers from RWTH, provides a key instrument to improve industrial uptake of research results via so-called Technology Transfer Projects (TTPs). TTPs provide a novel and systematic incentive for small to medium scale TT at European level. As an important support measure, the TTPs are backed by Technology Transfer Infrastructures (TTIs), such as regular workshops, trainings, and TT consultation services by experts which will are widely announced via the HiPEAC network of excellence.
All computing systems community members can apply for TTP funding in TETRACOM using a very efficient proposal scheme. In short, the “TTP algorithm” works as follows:
All TTPs are based on bilateral academia-industry TT partnerships. One academic partner A teams up with one industry partner B, who is interested in taking up a specific technology or IP developed by A for internal use, evaluation, or productization.

The total volume of the intended TTP is between 10k-200k Euros, and the total TTP duration is between 3-12 months.
Partner A, assisted by B, submits a lightweight three-page TTP proposal to TETRACOM that will be efficiently evaluated by experts according to several well-defined and public criteria. Following a positive evaluation, TETRACOM can provide funding of up to 50% of the total TTP volume. This funding will be received only by partner A, but it will indirectly also benefit partner B of course. Three calls for TTP proposals are being issued in TETRACOM over the project duration. The 2nd call is open during Nov 15-Dec 31, 2014.
While TETRACOM, as an open coordination action, naturally cannot provide any preference for specific projects, the EURETILE-TETRACOM link has been used to make all partners aware of the TTP opportunities regarding their newly developed foreground technologies. In fact, the first "EURETILE TTP" (INFN collaborating with EUROTECH) has already been accepted.

4.8 Workshop on Multi-core debugging

Regarding debugging technologies, the Multicore Application Debugging workshop (MAD, http://www.mad-workshop.de) was created by RWTH (in cooperation with the Technical University of Munich) in 2013 largely inspired on the challenges, opportunities and experience derived from the EURETILE debugging activities. MAD was created to foster the exchange of ideas, approaches, solutions and requirements between industry and academia on the field of debugging for parallel embedded systems. The community response to the MAD workshop has been exceptional as it targets a critical need which is highly underrepresented in research. The first edition of MAD was held in Munich (DE) on Nov. 14-15, 2013, with approx. 50 participants largely from industry. The second MAD workshop was organized in Athens (GR) in the context of the HiPEAC Fall Computing Systems Week (Oct. 8-10, 2014). MAD'14 had approx. 90 registered participants but it was accessible overall to approx. 200 people attending the HiPEAC event. In its two editions, MAD has had involvement of companies like Bosch, Intel, Infineon, Samsung, ARM, Freescale, Synopsys, Lauterbach and ST as well as academic institutions like ETH Zurich, Barcelona Supercomputing Center, University of Grenoble, TU Vienna, University of Luebeck, National University of Singapore, and many others. The success of MAD'13 and MAD'14 calls for a new edition in 2015, which is already under plans. Furthermore, a possible merge with the S4D (System, Software, SoC and Silicon Debug) conference from ECSI is being studied by the MAD organizers.

4.9 Workshop on computing on heterogeneous many-tile computing systems

The International Workshop "Perspective of GPU Computing in Physics and Astrophysics" (http://www.roma1.infn.it/conference/GPU2014/) was held in Rome, 15-17 September 2014.
The aim of the meeting was to present and discuss some of the modern applications in Physics and Astrophysics (and related subfields) of hybrid computational systems based on multicore CPU governing a set of Graphic Processing Units (GPUs) acting as number crunchers while a significant amount of time has been specifically devoted to the hardware and software aspects involved.
The meeting was organized by INFN jointly with several Italian research agencies, "Sapienza" University of Rome and primary industrial partners (Nvidia, HP, E4-Intel,…). More than 80 attendants participated from Europe, Asia and USA with very-well known (at international level) keynote and invited speakers from academia and industry. Piero Vicini was the workshop co-chair and EURETILE team was (minimally) involved in workshop organization. The EURETILE achievements were presented with 2 plenary talks and 1 poster.

5 Acknowledgements

The EURETILE project has been funded by the European Commission Grant Agreement no. 247846 Call: FP7-ICT-2009-4 Obj. FET-ICT-2009.8.1.
The authors of this Final Publishable Report acknowledge the contribution of other members of the EURETILE project, that participated during previous years, including:

- ETH Zurich: Hoeseok Yang
- RWTH: Christoph Schumacher, Jan H. Weinstock, Robert Buecs, Jovana Jovic
- TIMA: Etienne Ripert, Ikbel Belaid, Julian Michaud, Mohamad Jabber, Alexandre Chagoya-Garzon and Xavier Guérin

6 References

[Aloisio, 2011] Aloisio A., Ameli F., D'Amico A., Giordano R., Izzo V. and Simeone, F., "The NEMO experiment data acquisition and timing distribution systems", IEEE Nuclear Science Symposium Conference Record: 147-152, 2012.

[Ammendola, 2011] Roberto Ammendola, Andrea Biagioni, Ottorino Frezza, Francesca Lo Cicero, Alessandro Lonardo, Pier Stanislao Paolucci, Davide Rossetti, Francesco Simula, Laura Tosoratto, Piero Vicini "QUonG: A GPU-based HPC system dedicated to LQCD computing. In Application Accelerators in High-Performance Computing (SAAHPC), 2011 Symposium on, pages 113–122, 2011.

[Ammendola, 2013] R. Ammendola, A. Biagioni, O. Frezza, G. Lamanna, A. Lonardo, F. Lo Cicero, P. S. Paolucci, F. Pantaleo, D. Rossetti, F. Simula, M. Sozzi, L. Tosoratto, P.Vicini "NaNet: a flexible and configurable low-latency NIC for real-time trigger systems based on GPUs. In JINST, Journal of Instrumentation, Proceedings of Topical Workshop on Electronics for Particle Physics (TWEPP) 2013. IOP Publishing, 2013.

[Ammendola, 2013a] R Ammendola, A Biagioni, O Frezza, A Lonardo, F Lo Cicero, P S Paolucci, D Rossetti, F Simula, L Tosoratto, and P Vicini. APEnet+ 34 Gbps data transmission system and custom transmission logic. Journal of Instrumentation, 8(12):C12022, 2013.

[Ammendola, 2013b] Roberto Ammendola, Massimo Bernaschi, Andrea Biagioni, Mauro Bisson, Massimiliano Fatica, Ottorino Frezza, Francesca Lo Cicero, Alessandro Lonardo, Enrico Mastrostefano, Pier Stanislao Paolucci, Davide Rossetti, Francesco Simula, Laura Tosoratto, and Piero Vicini. GPU Peer-to-Peer Techniques Applied to a Cluster Interconnect. In Parallel and Distributed Processing Symposium Workshops PhD Forum (IPDPSW), 2013 IEEE 27th International, pages 806–815, 2013.

[Ammendola, 2013c] Roberto Ammendola, Andrea Biagioni, Ottorino Frezza, Francesca Lo Cicero, Alessandro Lonardo, Pier Stanislao Paolucci, Davide Rossetti, Francesco Simula, Laura Tosoratto, Piero Vicini, "Virtual-to-Physical Address Translation for an FPGA-based Interconnect with Host and GPU Remote DMA Capabilities.," in Field-Programmable Technology (FPT), 2013 International Conference on, 2013.

[Ammendola, 2013d] R. Ammendola, A. Biagioni, O. Frezza, A. Lonardo, F. Lo Cicero, P.S. Paolucci, D. Rossetti, F. Simula, L. Tosoratto and P. Vicini, "APEnet+ 34 Gbps data transmission system and custom transmission logic," in JINST, Journal of Instrumentation, Proceedings of Topical Workshop on Electronics for Particle Physics (TWEPP) 2013, IOP Publishing, 2013.

[Ammendola, 2014] R. Ammendola, A. Biagioni, O. Frezza, F. Lo Cicero, A. Lonardo, P. S. Paolucci, D. Rossetti, F. Simula, L. Tosoratto and P. Vicini "LO|FA|MO: Fault Detection and Systemic Awareness for the QUonG computing system", to be published, IEEE Proceedings (SRDS) International Symposium on Reliable Distributed Systems, Nara, Japan, October 6-9, 2014

[Boutellier, 2014] Jani Boutellier. User Guide for Orcc DAL Backend. Available: https://github.com/orcc/orcc/wiki/User-guide-for-Orcc-DAL-backend, 2014.

[Culler, 1999] David Culler, Jaswinder Pal Singh, and Anoo Gupta, Parallel Computer Architecture: A Hardware/Software Approach. Morgan Kaufmann, 1999.

[Guerin, 2009] Xavier Guerin and Frederic Petrot, A System Framework for the Design of Embedded Software Targeting Heterogeneous Multi-core SoCs, in Proc. Int'l Conf. on Application-specific Systems, Architectures and Processors, pages 153-160, 2009.

[Jones, 2009] D. Jones and N. Topham. High Speed CPU Simulation Using LTU Dynamic Binary Translation. In Proc. of International Conference on High Performance Embedded Architectures and Compilers (HiPEAC). Paphos, Cyprus, 2009.

[Jovic, 2012] J. Jovic, S. Yakoushkin, L.G. Murillo, J. Eusse, R. Leupers and G. Ascheid. Hybrid simulation of extensible processor cores. In Proc. Design, Automation & Test in Europe (DATE). Dresden, Germany. Mar. 2012.

[Jung, 2014] Hanwoong Jung, Chanhee Lee, Shin-Haeng Kang, Sungchan Kim, Hyunok Oh, and Soonhoi Ha. Dynamic Behavior Specification and Dynamic Mapping for Real-Time Embedded Systems: HOPES Approach. ACM Transactions on Embedded Computing Systems (TECS) 13, no. 4s (2014): 135.

[Kahn, 1974] Gilles Kahn. The Semantics of a Simple Language for Parallel Programming. In IFIP Congress, vol. 74, pages. 471–475, 1974.

[Kang, 2012] Shin-Haeng Kang, Hoeseok Yang, Lars Schor, Iuliana Bacivarov, Soonhoi Ha, and Lothar Thiele. Multi-objective mapping optimization via problem decomposition for many-core systems. In Embedded Systems for Real-time Multimedia (ESTIMedia), 2012 IEEE 10th Symposium on, pp. 28-37. IEEE, 2012.

[Kraemer, 2007] S. Kraemer, L. Gao, J.H. Weinstock, R. Leupers, G. Ascheid and H. Meyr: HySim: a fast simulation framework for embedded software development. In Proc. of Intl. Conference on Hardware/Software Codesign and System Synthesis (CODES/ISSS). Salzburg, Austria. 2007

[Kwon, 2008] Seongnam Kwon, Yongjoo Kim, Woo-Chul Jeun, Soonhoi Ha, and Yunheung Paek. A retargetable parallel-programming framework for MPSoC. ACM Transactions on Design Automation of Electronic Systems (TODAES) 13, no. 3 (2008): 39.

[Murillo, 2011] L.G. Murillo, W. Zhou, J. Eusse, R. Leupers and G. Ascheid. Debugging Concurrent MPSoC Software with Bug Pattern Descriptions. In Proc. System, Software, SoC and Silicon Debug Conference (S4D). Munich, Germany, Oct. 2011.

[Murillo, 2012a] L.G. Murillo, J. Eusse, J. Jovic, S. Yakoushkin, R. Leupers and G. Ascheid. Synchronization for hybrid MPSoC full-system simulation. In Proc. of Design Automation Conference. San Francisco, USA, Jun. 2012.

[Murillo, 2012b] L.G. Murillo, J. Harnath, R. Leupers and G. Ascheid. Scalable and Retargetable Debugger Architecture for Heterogeneous MPSoCs. In Proc. System, Software, SoC and Silicon Debug Conference (S4D). Vienna, Austria, Oct. 2012.

[Murillo, 2014] L.G. Murillo, S. Wawroshek, J. Castrillon, R. Leupers and G. Ascheid. Automatic Exploration of Software Concurrency Bugs with Event Ordering Constraints. In Proc. Design, Automation & Test in Europe (DATE). Dresden, Germany. Mar. 2014.

[Murillo, 2015] L.G. Murillo, R. Buecs, D. Hincapie, R. Leupers and G. Ascheid. SWAT: Assertion-based Debugging of Concurrency Issues at System Level. In Asia South Pacific Design Automation Conference (ASP-DAC). Chiba, Japan. Jan. 2015 (accepted for publication)

[Nohl, 2004] A. Nohl, G. Braun, O. Schliebusch, R. Leupers, H. Meyr and A. Hoffmann. A universal technique for fast and flexible instruction-set architecture simulation. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2004.

[Paolucci, 2013a] Paolucci, P.S., Bacivarov, I., Goossens, G., Leupers, R., Rousseau, F., Schumacher, C., Thiele, L., Vicini, P., "EURETILE 2010-2012 summary: first three years of activity of the European Reference Tiled Experiment.", (2013), arXiv:1305.1459 [cs.DC] , http://arxiv.org/abs/1305.1459

[Paolucci, 2013b] P.S. Paolucci, R. Ammendola, A. Biagioni, O. Frezza, F. Lo Cicero, A. Lonardo, E. Pastorelli, F. Simula, L. Tosoratto, P. Vicini, “"Distributed simulation of polychronous and plastic spiking neural

networks: strong and weak scaling of a representative mini-application benchmark executed on a small-scale commodity cluster'" (2013) arXiv:1310.8478 [cs.DC], http://arxiv.org/abs/1310.8478

[Paolucci, 2014b] P.S. Paolucci, I. Bacivarov, D. Rai, L. Schor, L. Thiele, H. Yang, E. Pastorelli, R. Ammendola, A. Biagioni, O. Frezza, F. Lo Cicero, A. Lonardo, F. Simula, L. Tosoratto, P. Vicini '"EURETILE D7.3 - Dynamic DAL benchmark coding, measurements on MPI version of DPSNN-STDP (distributed plastic spiking neural net) and improvements to other DAL codes'", (Aug 2014), arXiv:1408.4587 [cs.DC], http://arxiv.org/abs/1408.4587

[Rai, 2014] Devendra Rai, Pengcheng Huang, Nikolay Stoimenov, and Lothar Thiele. An Efficient Real Time Fault Detection and Tolerance Framework Validated on the Intel SCC Processor. In Proceedings of the The 51st Annual Design Automation Conference on Design Automation Conference, pp. 1-6. ACM, 2014.

[Schor, 2012] Lars Schor, Iuliana Bacivarov, Devendra Rai, Hoeseok Yang, Shin-Haeng Kang, and Lothar Thiele. Scenario-Based Design Flow for Mapping Streaming Applications onto On-Chip Many-Core Systems. In Proc. Int'l Conf. on Compilers Architecture and Synthesis for Embedded Systems (CASES), pages 71–80, 2012.

[Schor, 2013a] Lars Schor, Hoeseok Yang, Iuliana Bacivarov and Lothar Thiele. Expandable Process Networks to Efficiently Specify and Explore Task, Data, and Pipeline Parallelism. In Proc. Int'l Conf. on Compilers Architecture and Synthesis for Embedded Systems (CASES), pages. 1-10, 2013.

[Schor, 2013b] Lars Schor, Iuliana Bacivarov, Hoeseok Yang, and Lothar Thiele. Efficient Worst-Case Temperature Evaluation for Thermal-Aware Assignment of Real-Time Applications on MPSoCs. Journal of Electronic Testing 29, no. 4 (2013): 521-535.

[Schor, 2014a] L. Schor, I. Bacivarov, L.G. Murillo, P.S. Paolucci, F. Rousseau, A. El Antably, R. Buecs, N. Fournel, R. Leupers, D. Rai, L. Thiele, L. Tosoratto, P. Vicini, and J.H. Weinstock. EURETILE Design Flow: Dynamic and Fault Tolerant Mapping of Multiple Applications onto Many-Tile Systems. Proc. Int'l Symp. on Parallel and Distributed Processing with Applications (ISPA), Aug. 2014

[Schor, 2014b] Lars Schor, Iuliana Bacivarov, Hoeseok Yang, and Lothar Thiele. AdaPNet: Adapting Process Networks in Response to Resource Variations. Proc. Int'l Conf. on Compilers, Architecture, and Synthesis for Embedded Systems (CASES), Oct. 2014.

[Schumacher, 2010] C. Schumacher, R. Leupers, D. Petras, A. Hoffmann. parSC: synchronous parallel systemc simulation on multi-core host architectures. In Proc. of Intl. Conference on Hardware/Software Codesign and System Synthesis (CODES/ISSS). Scottsdale (AZ), USA. Oct. 2010.

[Schumacher, 2012a] C. Schumacher, J. H. Weinstock, R. Leupers and G. Ascheid. Scandal: SystemC Analysis for NonDeterminism AnomaLies. In Proc. of Forum on Specification and Design Languages (FDL). Vienna, Austria. Oct. 2012.

[Schumacher, 2012b] C. Schumacher, J. H. Weinstock, R. Leupers and G. Ascheid. Cause and Effect of Nondeterministic Behavior in Sequential and Parallel SystemC Simulators. In Proc. IEEE International High Level Design Validation and Test Workshop (HLDVT). Huntington Beach (CA), USA. Nov. 2012.

[Schumacher, 2013] C. Schumacher, J.H. Weinstock, R. Leupers, G. Ascheid, L. Tossorato, A. Lonardo, D. Petras, T. Groetker. legaSCi: Legacy SystemC Model Integration into Parallel SystemC Simulators. 1st Workshop on Virtual Prototyping of Parallel and Embedded Systems (VIPES), Boston, USA, May. 2013.

[Synopsys, 2014] Synopsys. Synopsys Processor Designer. [Online] http://www.synopsys.com/Systems/BlockDesign/processorDev/Pages/default.aspx. Accessed: Sept. 2014.

[Synopsys, 2014b] Synopsys Acquires Target Compiler Technologies, http://news.synopsys.com/2014-02-07-Synopsys-Acquires-Target-Compiler-Technologies, Feb. 7, 2014.

[Thiele, 2013] Lothar Thiele, Lars Schor, Iuliana Bacivarov, and Hoeseok Yang. Predictability for timing and temperature in multiprocessor system-on-chip platforms. ACM Transactions on Embedded Computing Systems (TECS) 12, no. 1s (2013): 48.

[Tretter, 2014] Andreas Tretter, Harshavardhan Pandit, Pratyush Kumar, and Lothar Thiele. Deterministic Memory Sharing in Kahn Process Networks: Ultrasound Imaging as a Case Study. In Proc. IEEE Symposium on Embedded Systems for Real-Time Multimedia (ESTIMedia), 2014.

[Weinstock, 2014] J. H. Weinstock, C. Schumacher, R. Leupers, G. Ascheid and L. Tosoratto. Time-Decoupled Parallel SystemC Simulation. In Proc. Design, Automation & Test in Europe (DATE). Dresden, Germany. Mar. 2014.