SHAPES: a tiled scalable Software Hardware Architecture Platform for Embedded Systems

Pier S. Paolucci
ATMEL Roma and INFN
INFN Dip. Fisica Univ. Roma
ATMEL Roma, Italy
+39 329 6315069
Pier.Paolucci@roma1.infn.it

Ahmed A. Jerraya
TIMA INPG
46 Av. Félix Viallet
38031 Grenoble cedex (France)
+33 47 6574759
Ahmed.Jerraya@imag.fr

Rainer Leupers
Inst. for Integrated Signal Proc. Syst.
Aachen Univ. of Technology
Germany
+49 241 80-28301
Leupers@iss.rwth-aachen.de

Lothar Thiele
Computer Eng. and Networks Lab.
Swiss Federal Institute of Technology
(ETH) Zürich, Switzerland
+41 44 632 7031
Thiele@tik.ee.ethz.ch

Piero Vicini
INFN
c/o Dip. Fisica Univ. Roma "La Sapienza"
P.le Aldo Moro 5, 00185 Roma, Italy
+39 06 49914423
Piero.Vicini@roma1.infn.it

ABSTRACT
Nanoscale systems on chip will integrate billion-gate designs. The challenge is to find a scalable HW/SW design style for future CMOS technologies. Tiled architectures suggest a possible path: "small" processing tiles connected by "short wires". A typical SHAPES tile contains a VLIW floating-point DSP, a RISC, a DNP (Distributed Network Processor), distributed on chip memory, the POT (a set of Peripherals On Tile) plus an interface for DXM (Distributed External Memory). The SHAPES routing fabric connects on-chip and off-chip tiles, weaving a distributed packet switching network. 3D next-neighbours engineering methodologies is adopted for off-chip networking and maximum system density. The SW challenge is to provide a simple and efficient programming environment for tiled architectures. SHAPES will investigate a layered system software, which does not destroy algorithmic and distribution info provided by the programmer and is fully aware of the HW paradigm. For efficiency and QoS, the system SW manages intra-tile and inter-tile latencies, bandwidths, computing resources, using static and dynamic profiling. The SW accesses the on-chip and off-chip networks through a homogeneous interface.

Categories and Subject Descriptors
C.1.4 [**Processor Architectures**]: Parallel Architectures; C.0 [**Computer Systems Organization]:** General - *system architectures, hardware/software interfaces, modeling of computer architecture;* C.3 [**Special-purpose and Application-Based Systems**]: *real-time and embedded systems, signal processing systems.;* B.7.2 [**Integrated Circuits]** Design Aids – *simulation;* F.2.2 [**Analysis of Algorithms and Problem Complexity]** Nonnumerical Algorithms and Problems – *sequencing and scheduling;*

General Terms
Design, Performance, Experimentation, Languages.

Keywords
Tiled Parallel Architectures, Embedded Systems, Model Based Design, Application Mapping, Network of Processes, Binding, Scheduling, Simulation, Hardware dependent Software, Retargetable Compiler, RISC, VLIW, Distributed Network Processors, MP-SOC

CODES+ISSS'06, October 22–25, 2006, Seoul, Korea.

1. INTRODUCTION
1.1 The tiled HW Architecture of SHAPES
A key objective for deep sub-micron technologies is the minimization of wire delay problems[1-4]. A second target is to define the appropriate strategy to manage the design complexity, with several hundreds of million gates.We propose a tiled[5-6] architectural strategy that employs building blocks that are scalable on future technology nodes. In SHAPES[1], a typical DSP oriented tile (RDT: Figure 1) is composed of a RISC, a VLIW DSP, a DNP (Distributed Network Processor), on-tile memories and a set of on-tile peripherals (POT). Each tile can be equipped

[1] www.shapes-p.org: a European Project (FET-FP6-2004-IST-4.2.3.4(viii)- Advanced Comp. Arch. started Jan 2006.

with a Distributed eXternal Memory (DXM). The tile is the evolution of Atmel Diopsis[7], a multiprocessor SoC which includes a RISC + the floating-point VLIW mAgic DSP[11].

The SHAPES routing fabric connects on-chip and off-chip tiles, weaving a distributed packet switching network. 3D next-neighbours toroidal engineering methodologies will be adopted for off-chip networking and maximum system density, leveraging

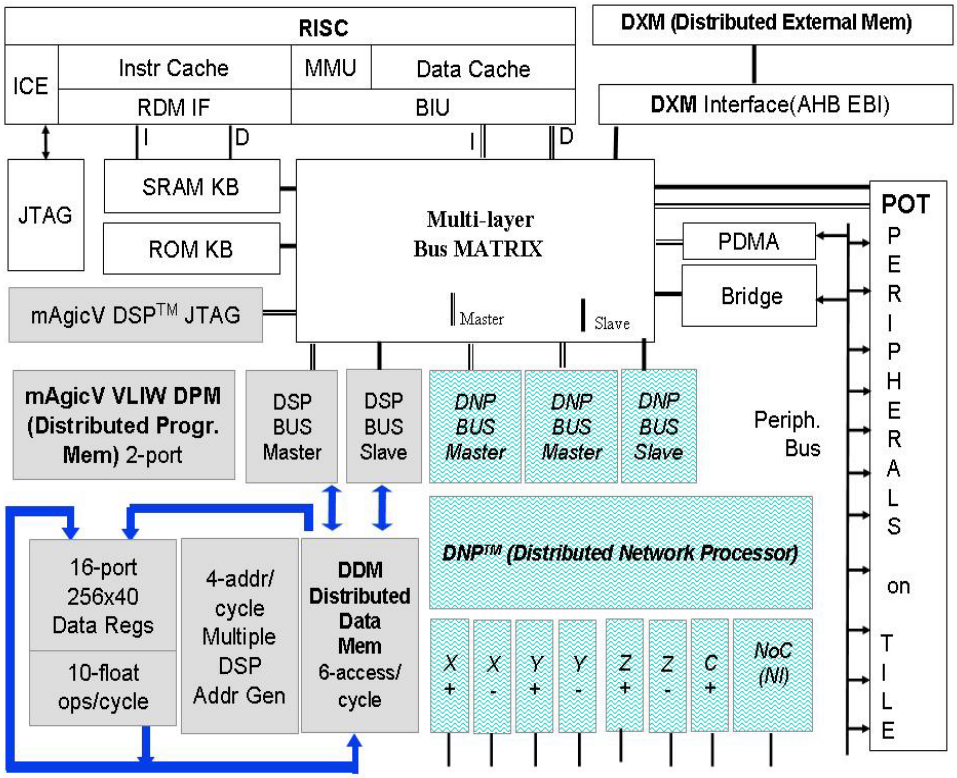

Figure 1. RDT: The RISC + VLIW + DNP elementary tile of a SHAPES parallel system.

on the background developed during several generations of INFN Ape projects[12-14,28], dedicated to the design and development of massive parallel processors for numerical physical simulations (Lattice QCD) based on custom VLIW Processors.

The DNP uses 3D next-neighbors connections for off-chip interconnections (X+, X-, Y+, Y-, Z+, Z- in Figure 1 bottom-right) and interfaces a NoC (SpidergonTM, developed by ST Microelectronics) to integrate multiple tiles on a single chip. A separate link is dedicated to collective communications.

The tile includes also a VLIW floating-point DSP (bottom-left in Figure 1). In a conventional processor architecture, which detects parallelism at execution time and pushes the clock speed to the limit, the die area required for control logic overhead frequently dwarfs that which is required by the functional units. For a discussion of the efficiency of DSPs relying on parallelism detected before execution and exploited by VLIW and appropriate software scheduling see[9]. In our opinion and experience, moderate clock speeds are an ideal complement to VLIW architectures because they reduce pipeline depth, bypass logic, and speculation correction logic. This choice simplifies the task of high-level language compilers. A moderate clock speed also allows simpler clock tree management and lower supply voltages. The classical drawback of VLIW architectures is that the longer instruction words require more memory. While conventional code compression /decompression schemes mitigate this problem[10-11], they also increase control logic overhead. We solved this problem by developing in hardware a small footprint (18 Kgate) VLIW Dynamic Program memory Decompression system (DyProDe). Control logic can take up as much a majority of the die area dedicated to the logic of superscalar cores. We demonstrated that the mAgic VLIW architecture dedicates to control less than 25% of the logic area.

An appropriate balancing of instruction level parallelism (ILP), memory size, and frequency provides a sound control of potential wire delay problems within the individual tile. We based our tile design on a simple dimensional analysis, which describes a multiprocessor scaling strategy, based on multiple tiles, over Deep Sub Micron (DSM) technologies (for an introduction to the scaling strategy see[7]).

Interconnection wires are confined inside each tile or link each tile to its next neighbors. The residual number of wires at global die level are kept to an absolute minimum, routed on thicker wires or managed by the insertion of repeaters.

The target of SHAPES is to design a Teraflops board scalable to Petaflops systems. The SHAPES architecture should scale from low end single module hosting 4-8 tiles for mass market applications, covering classic digital signal processing systems like radar and medical equipments (2 K tiles), and scaling to high-end systems requiring massive numerical computation (32 K tiles).

1.2 The Software Challenge of SHAPES

The characteristics of the tiled architecture impose several challenges on the software development process, and only by breakthroughs the tremendous potential of tiled architectures can be fully exploited by the application. For example, typical issues are long delays between distant tiles, hot spots in the communication, and limited parallelism that is exposed in the application. If those problems are not managed by the application development tools, the computing power can not be exploited. Today, in lack of those tools, the architecture tends to be over-designed or do not fulfill the requirements.

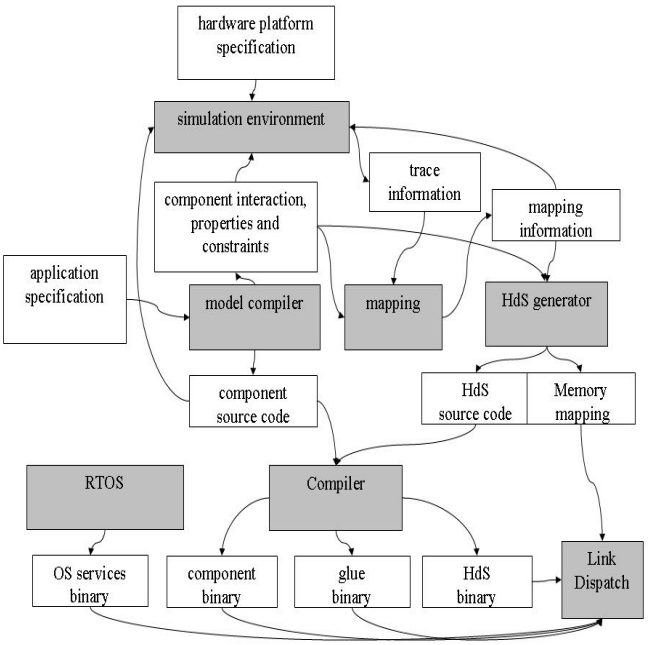

Figure 2. The SHAPES software development environment.

The objective of the SHAPES system software is to obtain efficient execution of applications on the SHAPES hardware, minimizing the effort required to the application programmer. We aim to reach efficiencies comparable to what would be obtained by the best assembler programmer.On massive parallel systems this is an ambitious target. In first place, the key is to avoid destroying the information about the algorithmic structure. In second place, the system software must be fully aware of key architectural parameters like the ratios among bandwidth, computing capability, pipeline and latencies.

In any case, a new design flow for applications to be run on heterogeneous multi-tile DSP architectures is required. It must cope with the complexity and shorten time-to and time-in market. It must be QoS-aware, and must support a continuous and iterative design taking into account the hardware platform, the software environment and the application structure.

The proposed approach can be summarized as an integrated scalable QoS-aware design process that (a) can be applied to scalable hardware platforms, and (b) allows continuous monitoring and guarantees the required quality and real-time constraints in a systematic manner.

The individual components of this approach are related to compiler and simulation, hardware dependent software and distributed operation.

2. THE SHAPES SOFTWARE DEVELOPMENT ENVIRONMENT

Figure 2 serves to clarify the interdependences between the different building blocks related to the SHAPES software development environment. In addition, it also describes the overall scalable and QoS-aware design process as will be developed in SHAPES. Note that not all links are shown in order not to overload the figure.

The major starting point is the application specification shown on the left hand side. It should enable the specification of applications from the intended domains and contain functional and non-functional information. In particular, we envision Kahn process networks [27] as starting points for our model-based design trajectory in order to explicitly model (a) components, (b) their interaction and (c) the available degree of parallelism. The model compiler extracts relevant information such as the source code for the individual software components and information about their interaction.

The simulation environment uses this information together with a characterization of the underlying hardware platform and the mapping of components to it in order to perform a simulation on various levels of abstraction, thereby integrating also instruction set simulators. The mapping information comprises not only the mapping of computation and communication on the available resources but also resource sharing strategies such as arbitration and scheduling. As a result of the simulation, we obtain a simulation trace which will be used in the mapping phase in order to improve the initial guess. Therefore, it becomes apparent that there is a critical exploration cycle that involves simulation and mapping.

What is not shown in the figure is that the mapping phase will not only be based on (a) trace information from the simulation but also from (b) analytic performance analysis [22,26] and (c) run-time monitoring. The mapping strategy and exploration will be based on multi-objective optimization taking into account the various conflicting criteria such as throughput, delay, predictability and efficiency. Its scalability is achieved by using the estimation of non-functional properties with different accuracy-runtime tradeoffs and by using an initial application specification that exposes the available degree of parallelism.

Based on the mapping information and the interaction between the different software components, the hardware dependent software phase will generate the necessary dedicated communication and synchronization primitives, together with glue code. This way, the generated software is specialized towards the application, the mapping and the underlying hardware platform.

All generated source code will be compiled towards the target processors, memory and communication structure present in the multi-processor tiles. In the re-targetable compiler, optimizations will be done based on the detailed interface scheduling as defined by the mapping and the hardware-dependent software layer.

2.1 High-level exploration, mapping, and simulation

The basic principle of high-level exploration is to separate different programming concerns [23], e.g. separating computation from communication and separating application from architecture. In SHAPES, the specification of application functionality, i.e. the application programming, is fully decoupled from mapping and system architecture. There is also a clear separation between computation and communication within the application specification. A programmer's view of the

hardware architecture is abstracted, which can be regarded as a virtual execution platform seen by the application programmer. A separate mapping process is used to bind application to architecture. Based on these, model-based performance evaluation of different application-to-architecture mappings is feasible. Such an evaluation can be based on abstract analytic [22,26] or simulation models or on combinations thereof. The performance evaluation results are then used as a basis for mapping decisions in a manual or semi-automatic design space exploration process.

As mentioned before, the SHAPES HW platform is highly flexible and scalable. Using a set of heterogeneous processing tiles which have different computational characteristics as basic building blocks, the high level design space of the SHAPES hardware system can be searched in two dimensions: the number and the type of the tiles. Additionally, taking the complexity of different tile types and the inter-tile connection into account, SHAPES poses a big challenge to the hardware design methodology and tools. In the high-level exploration of SHAPES, simulation technique plays an important role.

The SHAPES simulation environment will employ much higher abstraction levels than Register Transfer Level (RTL). To simulate the processing elements in SHAPES, mainly two abstraction levels can be explored, namely Instruction Accurate (IA) or Cycle Accurate (CA). CA simulation models contain much more micro architecture details than IA models do, which means better accuracy. However, IA simulators provide much faster simulation speed, which leads to shorter exploration cycles. Balance needs to be achieved by the SHAPES simulation platform. When the system scales up, it might be the case that a simulation system built with IA processor simulators cannot provide satisfactory simulation speed. Then, it will be necessary to push the abstraction level even higher than the IA level by parameterizing the processing elements [20-21]. This level of abstraction could be desired at early design phases, when the system configuration is still open, and a lot of design variants need to be explored.

Similar to the processing element modelling, different levels of abstraction also exist for the modelling of inter-connections. The level of abstraction can vary from very detailed pin-accurate level to highly abstracted functional level without modeling timing or protocol details[19]. For SHAPES, especially interesting is Transaction Level Modeling (TLM). Compared to the lower pin-accurate level, TLM abstracts away the number of events and amount of information that have to be processed during simulation to the minimum required. Instead of driving the individual signals of a bus protocol, TLM only exchanges the data payload which is really necessary. By this approach, the simulation speed can be kept acceptable when the system scales up.

Except for the design space exploration of SHAPES, the simulation environment is also usable for the software development. Due to the complexity of SHAPES, it is not likely that a full-fledged HW prototype could be available in the early stages of the project. The SHAPES simulation platform then acts as a virtual prototype[24]. Equipped with instruction set simulators the platform enables software developers to try their software out as if it is on the real hardware, which implies shorter development cycles. Moreover, there is some information desired by software developers which cannot be derived from HW easily or for free, e.g. execution traces, profiles. However, since the SHAPES software development flow will involve a mapping procedure which explores different mappings of tasks to processing elements as well as inter-task communications to HW/SW resources, it is important to use trace and profiling information to evaluate the quality the mappings. Compared to HW, it is much easier for a simulation model to collect such information and feed it back to software developers. This is another place in SHAPES, where simulation techniques could help. In short, the simulation environment could be used in SHAPES for three purposes, high-level design space exploration, software development and performance estimation.

2.2 Hardware-dependent software, operating system and compiler

A well-defined platform-specific HdS allows to build efficient hardware-software interfaces and thus to minimize performance penalties. In fact, it can be fully aware of architecture specific resource constrains, such as inter-tile latencies, available bandwidths, static profiling, and dynamic profiling. Hardware-dependent software also enables a HW paradigm-aware programming model. The HdS generally represents a more or less thick software layer that may completely hide the underlying hardware through an application programmer interface (API). Furthermore, since it generally already implements many design decisions (policies), such an abstraction layer is tightly coupled to the operating system.

In our view, the HdS encapsulates the operating system (OS), specific communication software and underlying hardware abstraction layer (HAL). The operating system and specific I/O software are responsible for providing the necessary services to manage and share resources. The services include the scheduling of application threads on top of the available processing elements, inter-task communication, external communication, and all other kinds of resources management and control. At OS level, the hardware dependency is kept functional, i.e. it concerns only high level aspects of the hardware architecture like the type of available resources, and makes use of a hardware abstraction layer programming model (HAL_API). Low level details about how to access the available resources are abstracted by the hardware abstraction layer (HAL).

The federative HdS term underline the fact that, in an embedded context, we are concerned with application specific implementations of these functionalities that strongly depend on the target hardware architecture.

Novel retargetable compilation technologies, which are aware of the low-latency intra-tile and inter-tile communication interfaces, will be developed for the computational tiles of SHAPES, extending the capability of the Chess/Checkers tool-suite [23]. The feedback from the retargetable compiler, we allow to explore the architectural optimization of the computational tiles. This optimization is vital to ensure high computational efficiency without the need for low-level assembly coding.

3. CONCLUDING REMARKS

There is no processing power ceiling for low consumption, low cost, dense Numerical Embedded Scalable Systems for future embedded audio, video and human-centric applications.

The ratio between shipped embedded processors and general purpose processors is increasing. Therefore, the driving force on computing architectures is shifting from PC platforms to embedded systems. Tiled architecture will cover a significant share of 10+ year embedded applications. SHAPES will set a new density record with multi-Teraops single-board computers and multi-Petaops systems. SHAPES proposes a programmable, high performance, low power, dense systems solution designed for seamless interfacing with reconfigurable logic and signal acquisition and generation systems

Moving from Giga to Tera and then to Peta computing requires a paradigm shift in the software design process. The envisioned underlying hardware platform consists of a highly scalable number of heterogeneous multi-processor computing elements (tiles) that are communicating via On Chip and Off Chip Networks. Neither methods from the general purpose computing area nor the classical HW/SW codesign approaches meet the requirements of software development for such high end performance embedded computing systems. In the area of heterogeneous scalable platforms, new research efforts are still needed for core hardware dependent software technologies, including abstraction of non-functional properties such as memory allocation, real-time constraints, and concurrent optimization of hardware and software. The proposed approach suggests a possible solution. Thanks to this approach, generation after generation, the number of tiles on a single-chip will grow, but the application will be portable. It would be also interesting to include in the tile a fraction of silicon area dedicated to programmable logic. Appropriate, tile oriented manufacturing technique should be investigates. We plan to investigate those two aspects through dedicated project to be launched in the future.

4. ACKNOWLEDGMENTS

We acknowledge fruitful discussions with other partners of the SHAPES project: Marcello Coppola – STMicroelectronics, Luigi Raffo and Gianni Mereu - Università di Cagliari about NoC related issues, Gert Goossens – Target Compiler Technologies, about communication aware compiler scheduling techniques, Thomas Sporer and Sandra Brix – Fraunhofer IDMT about application requirements.

5. REFERENCES

[1] R. Ho, K. Mai and M. Horowitz, “The Future of Wires”, *Proc. IEEE*, 89-4 (2001)490-504.

[2] D. Sylvester and K. Keutzer, “Impact of Small Process Geometries on Microarchitectures in Systems on a Chip”, *Proc. IEEE*, 89-4(2001)467-489.

[3] L.P. Carloni, A.L. Sangiovanni-Vincentelli, "Coping with latency in SOC Design", *IEEE Micro* 22-5 (2002) 24-35.

[4] A. Allan et al., “2001 Technology Roadmap for Semiconductors”, *IEEE Computer* 35-1(2002)42-53.

[5] W.J. Dally and S. Lacy, “VLSI Architectures: Past, Present and Future”, *Proc. Advanced Research in VLSI Conf.*, IEEE Press (1999)232-241.

[6] M.B. Taylor et al., “The Raw Microprocessor: A Computational Fabric for Software Circuits and General-Purpose Programs”, *IEEE Micro* 22-2(2002)25-35.

[7] P.S. Paolucci et al. “Janus: A gigaflop VLIW+RISC Soc Tile”, *Hot Chips 15 IEEE Stanford Conference* (2003). http://www.hotchips.org (note: Janus was the development name of Diopsis. see www.atmelroma.it).

[8] J. Ying Fai Tong et al. “Reducing Power by Optimizing the Necessary Precision Range of Floating Point Arithmetic”, *IEEE Trans. On VLSI Systems*, 8-3 (2000)273-286.

[9] P. Faraboschi, G. Desoli, J.A. Fisher, “The Latest Word in Digital and Media Processing”, *IEEE Signal Processing Mag.* 15-2(1998)59-85.

[10] R.P. Clowell, J. O’Donnell, D.P. Papworth, P.K. Rodman, “Instruction Storage Method with a Compressed Format Using a Mask Word”, *U.S. Patent 5057837*, (Oct 1991).

[11] P.S. Paolucci, P. Kajfasz et al., “mAgic-FPU and MADE: A customizable VLIW core and the modular VLIW processor architecture description environment”, *Computer Physics Communication* 139(2001)132-143.

[12] A. Bartoloni, P.S. Paolucci et al., “A Hardware Implementation of the APE100 Architecture*”, Int. Journ. Mod. Phys. C* 4(1993)969.

[13] N. Cabibbo and P.S. Paolucci, “SIMD algorithm for Matrix Transposition*”, Int. Journ. Mod. Phys. C* 6(1995)183.

[14] F. Aglietti, P. S. Paolucci, et al. , “The teraflop supercomputer APEmille: architecture, software and project status report” *Computer Physics Communications,* 110,1-3 (May 1998) 216-219

[15] J. Rabaey, A. Chandrakasan, B. Nikolic, *Digital Integrated Circuits*, 2-nd Edition, Prentice-Hall (2003) Chapter 4 and 9.

[16] E. de Kock, G. Essink, W. Smits, P. van der Wolf, J.-Y. Brunel, W. Kruijtzer, P. Lieverse, and K. Vissers. YAPI: Application modeling for signal processing systems. In *Proc. 37th Design Automation Conference (DAC’2000)*, pages 402–405, Los Angeles, CA, June 5-9 2000.

[17] G. E. Erwin de Kock. *Y-Chart Application Programmer’s Interface: The YAPI Programmer’s and Reference Guide* Version 2.0.0.

[18] P. van derWolf, E. de Kock, T. Henriksson,W. Kruijtzer, and G. Essink. Design and Programming of Embedded Multiprocessors: An Interface-Centric Approach. In *Proc. IEEE-ACM-IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS’04)*, Stockholm, Sweden, Sept. 8-10 2004.

[19] T. Kogel, M. Doerper, A. Wieferink, R. Leupers, G. Ascheid, H. Meyr, S. Goossens “A Modular Simulation Framework for Architectural Exploration of On-Chip Interconnection Networks” In *The First IEEE/ACM/IFIP*

International Conference on HW/SW Codesign and System Synthesis, Newport Beach (California USA), Oct 2003

[20] T. Kempf , M. Dörper, R. Leupers, G. Ascheid, H. Meyr "A Modular Simulation Framework for Spatial and Temporal Task Mapping onto Multi-Processor SoC Platforms", In *Proc.of the Conference on Design, Automation & Test in Europe (DATE)*, Munich, Germany, March 2005

[21] T. Kempf, K. Karuri, S. Wallentowitz, G. Ascheid, R. Leupers, H. Meyer " A SW Performance Estimation Framework for Early System-Level-Design Using Fine-Grained Instrumentation" *In Proc.of the Conference on Design, Automation & Test in Europe (DATE)*, Munich, Germany, March 2006

[22] S. Chakraborty, S. Künzli, and L. Thiele. "A general framework for analyzing system properties in platform-based embedded system designs," In *Proc. 6th Design, Automation and Test in Europe (DATE)*, pages 190–195, March 2003.

[23] K. Keutzer, S. Malik, A. Newton, J. Rabaey, and A. Sangiovanni-Vincentelli, "System level design: Orthogonalization of concerns and platform-based design," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 19, no. 12, Dec. 2000.

[24] "Virtual Platform Designer" http://www.CoWare.com

[25] "Chess/Checkers, a retargetable tool-suite for embedded processors", Target Compiler Technologies, http://www.retarget.com/doc/target-whitepaper.pdf.

[26] L. Thiele, S. Chakraborty, and M. Naedele. "Real-time calculus for scheduling hard real-time systems," In *Proc. IEEE International Symposium on Circuits and Systems* (ISCAS), volume 4, pages 101–104, 2000.

[27] G. Kahn, "The semantics of a simple language for parallel programming," in Proc. of the IFIP Congress 74, (1974).

[28] Belletti, F et al. "Computing for LQCD: apeNEXT", *Computing in Science and Engineering*, 8-1, pp. 18-29, Jan/Feb, 2006