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
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

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

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 SHAPES1, 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

---

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 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.
The SHAPES software development environment is designed to efficiently execute applications on the SHAPES hardware, minimizing the effort required from the application programmer. The objective is to achieve efficiencies comparable to those obtained by the best assembler programmer. In massive parallel systems, this is a challenging task due to the need to preserve the algorithmic structure and consider architectural parameters like bandwidth, computing capability, pipeline and latencies.

A new design flow is required for applications running on heterogeneous multi-tile DSP architectures, focusing on complexity reduction and time-to-market. The approach must be QoS-aware, continuously improving design through hardware platform, software environment, and application structure. The proposed method is an integrated scalable QoS-aware design process that can be applied to scalable hardware platforms and allows continuous monitoring and optimization.

2. THE SHAPES SOFTWARE DEVELOPMENT ENVIRONMENT

Figure 2 illustrates the interdependences between the building blocks related to the SHAPES software development process. It also presents the overall scalable and QoS-aware design process as 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. It enables the specification of applications from intended domains and contains functional and non-functional information. The Kahn process networks [27] serve as a model-based design trajectory, explicitly modeling components, their interaction, and available parallelism. The model compiler extracts source code for software components and their interaction.

The simulation environment uses this information along with a characterization of the underlying hardware platform to perform simulations on various levels of abstraction, integrating instruction set simulators. Mapping information includes computation and communication on available resources and resource-sharing strategies. As a result, a simulation trace is obtained, which is used to improve the initial guess during mapping.

The mapping phase benefits from trace information, analytic performance analysis [22,26], and run-time monitoring. The mapping strategy is based on multi-objective optimization considering criteria such as throughput, delay, predictability, and efficiency. Scalability is achieved through different accuracy-runtime tradeoffs.

Based on the mapping information, the hardware-dependent software phase generates communication and synchronization primitives along with glue code. The generated software is specifically designed for the application, mapping, and underlying hardware platform.

All generated source code is compiled for the target processors, memory, and communication structure present in multi-processor tiles. Runtime optimizations are performed based on detailed interface scheduling defined by the mapping and the hardware-dependent software layer.

2.1 High-level exploration, mapping, and simulation

The core principle of high-level exploration is to separate different programming concerns [23], e.g., separating computation and communication from application architecture. In SHAPES, application functionality is decoupled from mapping and system architecture. There is 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


