MAPS (MPSoC Application Programming Studio)

Weihua Sheng*, Jianjiang Ceng*, Jeronimo Castrillon*, Anastasia Stulova*, Rainer Leupers*
Arif Ullah Khan†, Tsuyoshi Isshiki†

* Institute for Integrated Signal Processing Systems, RWTH Aachen University, Germany
† Dept. of Communications and Integrated Systems, Tokyo Institute of Technology, Japan

Abstract

MAPS project is a prototype tool-set to address a typical problem setting of SW development for embedded systems running multiple applications simultaneously that partially compete for the resources.

1. Introduction

Heterogeneity of the processing elements and communication architectures in modern embedded MPSoCs, as well as real-time requirements and multi-application usage scenarios make the sole reuse of existing multicore programming technologies difficult. Thus, MPSoC programming practices are currently under rapid evolution and more novel research is required.

The MAPS (MPSoC Application Programming Studio) project is part of RWTH Aachen’s Ultra high speed Mobile Information and Communication (UMIC) research cluster [1]. It is a research prototype tool-set for semi-automatic code parallelization and task-to-processor mapping, inspired by a typical problem setting of SW development for wireless multimedia terminals, where multiple applications and radio standards can be activated simultaneously and partially compete for the same resources. The overview of the MAPS work-flow is shown in Figure 1.

Figure 1: MPSoC Software Design-flow by MAPS

Applications can be specified either as sequential C code or in the form of pre-parallelized processes, where real-time properties such as latency and period as well as preferred PE types can be optionally annotated. MAPS uses advanced dataflow analysis to extract the available parallelism from the sequential codes (see [2] for a more detailed description of code partitioning) and to form a set of fine-grained task graphs based on a coarse model of the target architecture. The resulting partitioning/mapping can be exercised and refined with a fast, high-level SystemC based simulation environment (MAPS Virtual Platform, MVP), which has been put in practice to evaluate different software settings in a multi-application scenario. After further refinement, a code generation phase translates the tasks into C codes for compilation onto the respective PEs with their native compilers and OS primitives. The SW can then be executed, depending on availability, either on the real HW or on a cycle-approximate virtual platform incorporating instruction-set simulators. MAPS is also developing a dedicated task dispatching ASIP (OSIP, operating system ASIP) in order to enable higher PE utilization via more fine-grained tasks and low scheduling overhead. Early evaluation case studies exhibited great potential of the OSIP approach in lowering the task-scheduling overhead, compared to an additional RISC performing scheduling, in a typical MPSoC environment.

MAPS is an on-going research activity and is covered in the Multicore Applications Special Day of DATE 2009. In the University Booth we will demonstrate some results of the MAPS project, namely the code partitioning framework (MAPS-TCT framework) and the high-level SystemC-based simulation environment (MVP). In the following sections, technical details on the proposed demonstrations will be illustrated.

2. MAPS-TCT Framework

The MAPS-TCT framework is an integrated framework developed for assisting MPSoC programmers with C application parallelization. Unlike a fully autonomous parallel compiler, it provides a set of tools which guides the parallelization process. MAPS has been coupled with the TCT [3] as back-end which utilizes a so-called Tightly-Coupled-Thread (TCT) programming model [4] for its MPSoC silicon prototype. The overview of the MAPS-TCT framework is shown in Figure 2.

Figure 2: MAPS-TCT framework

The parallelization framework starts with the sequential C code analysis – the tool-set provides high level profiling information to help the designer to understand the application execution. Both static and dynamic approaches
are applied here to produce e.g. call-graphs, source-level profiling and control/data-dependency information. The results are summarized in so-called Weighted Statement Control/Data-Flow Graph (WSCDFG), which is the compiler IR graph with the annotations of detailed program analysis results. The second step is to partition the C source code into coarse-grained tasks. An iterative heuristic analysis results. The second step is to partition the C source code into coarse-grained tasks. An iterative heuristic analysis results. The second step is to partition the C source code into coarse-grained tasks. An iterative heuristic analysis results. The second step is to partition the C source code into coarse-grained tasks. An iterative heuristic analysis results. The second step is to partition the C source code into coarse-grained tasks. An iterative heuristic analysis results. The second step is to partition the C source code into coarse-grained tasks. The partitioning results, i.e. tasks, are delivered to the programmer as suggestions on how to decompose the application. TCT programming model is used (Fig. 3) as a drastically simple way to specify system partitioning on the C code.

Figure 3: (a) A Sample Code Using TCT Programming Model; (b) TCT MPSoC

The TCT tools are integrated as a back-end for binary code generation, instruction-set simulation and parallel performance profiling. Its target PE is a RISC processor inside an MPSoC [4] (Fig. 4) where 6 PEs are connected with a full crossbar. The PEs are also connected with their host-CPU through an AHB bus. Each PE has a dedicated module for high speed communication (4-6 cycles setup, 4Bytes/cycle burst transfer, 1 cycle PE-to-PE latency).

The TCT SW toolkit consists of the following tools. TCT Compiler which takes C codes with THREAD annotation as input, analyzes inter-procedural dependence flows including pointer dereferences, inserts thread communication instructions for thread activation, data transfer and data synchronization, and finally generates partitioned thread executable binaries. TCT Simulator which consists of a set of cycle-accurate instruction-set simulators for parallel thread execution. PE communication on the full crossbar interconnect is also modeled at cycle accuracy. TCT Trace Scheduler which is a fast performance estimation tool. It uses an abstract computation/communication model to quickly report the estimated execution time with accuracy of few % error from the TCT Simulator.

In the complete exploration flow, TCT tools examine the speedup brought by the MAPS partitioning. The programmer gets important feedback like thread schedule, sequential/parallel performance, and other characteristics. Based on this information, the programmer can decide whether further improvement is necessary to exploit the potential of the parallel platform. The user interface of the MAPS-TCT framework is built upon the C/C++ development tools of the Eclipse environment, with a set of plug-ins developed to control the MAPS-TCT tools conveniently and visualize the result intuitively.

3. MVP

The MVP provides an environment for the user to quickly prototype multi-application/task scenarios at high-level abstraction. The C functional models can be fully reused, and for inter-application/task communications, a small number of API functions are provided by the MVP, which are designed to be friendly for target migration. The MVP uses SystemC based Virtual Processing Elements (VPEs) to model computation units. Multiple applications can be loaded and executed in one VPE, and multiple VPEs can be instantiated in the MVP to run parallelized applications, examine the functionality and check the relative performance of the parallel tasks. Multitasking is supported through the schedulers implemented in the VPE. This SystemC concurrent simulation layer is completely transparent to the user's C functional models by a source instrumenter in the MVP SW toolchain. Figure 4(a) gives an overview of the framework, which consists of two parts: the MVP software toolchain and the MVP simulator. The former supports the C application development and compiles the source code into native shared libraries to be loaded by the simulator. The latter is a SystemC based simulator which uses VPE instead of ISS to model processing elements. Since the C application and the simulator are completely decoupled, the developer, as user, does not need to care about the underlying SystemC details, and thus can focus more on the programming. A GUI (Fig. 4(b)) is provided for the user to configure the virtual platform, perform task-to-VPE mapping and observe the system simulation.

Figure 4: (a) MVP Overview; (b) MVP GUI

4. Conclusion

This paper summarizes the on-going research project MAPS. The technical details of the tools that are demonstrated in the MAPS Booth of DATE 2009 are presented. This work has been supported by the UMIC Research Centre, RWTH Aachen University and the EU Networks of Excellence “ArtiStDesign” and “HiPEAC”.

5. References

[1] UMIC (Ultra high speed Mobile Information and Communication), http://www.umic.rwth-aachen.de/