Methodologies and Tools for Development of Signal Processing Software on Multicore Platforms

Jerker Bengtsson*and Bertil Svensson†
Centre for Research on Embedded Systems
Halmstad University
PO Box 823, SE-301 18 Halmstad, Sweden

Research Focus To be able to handle the rapidly increasing programming complexity of multicore processors, we argue that *domain specific development tools are needed*. The signal processing required in radio base stations (RBS), see figure 1, is naturally highly parallel and described by computations on streams of data. Each module in the figure encapsulates a set of functions, further exposing more pipeline-, data- and task level parallelism as a function of the number of connected users. Many radio channels have to be processed concurrently, each including fast and adaptive coding and decoding of digital signals. Hard real-time constraints imply that parallel hardware, including processors and accelerators is a prerequisite for coping with these tasks in a satisfactory manner.

One candidate technology for building baseband platforms is multicores. Based on the specified timing constraints and different types and levels of parallelism in this specific type of application, we are especially interested in multicores where cores:

- are many to the number
- are tightly coupled via a decentralized interconnection network and offer low core send- and receive occupancy
- have individual instruction execution ability
- allow the programmer to orchestrate the transactions between local and global memories

*contact: Jerker.Bengtsson@hh.se
†contact: Bertil.Svensson@hh.se

Parallel Models of Computation One of the requirements from industry in order to fully adopt commercial multicore technology – in favor of in-house hardware solutions – is that the application software, the tools and the programming models are portable. Streaming models of computation (MoC) offer a good match both when expressing signal processing applications and as input when generating parallel code for multicore processors. One good example is synchronous dataflow (SDF) [1]. Using SDF it is possible to describe different types and granularities of parallelism, while at the same time abstract away physical mapping details related to memory allocation, scheduling of communication and synchronization.

Research has demonstrated efficient compiler heuristics for programming languages based on streaming MoC, achieving good speedup and high throughput for parallel benchmarks [2]. However, even though a compiler can generate optimized code the programmer is left with very little control of how the source program is transformed and mapped on the cores. This means that if the resulting code output does not comply with the system timing requirements, the only choice is trying to restructure the source program.

We argue that *experienced application programmers must be able to direct and specialize the parallel mapping strategy by giving directive tool input.*

Flexible Mapping Strategies For complex real-time systems, such as baseband processing platforms, we see a need for tunable code parallelization- and

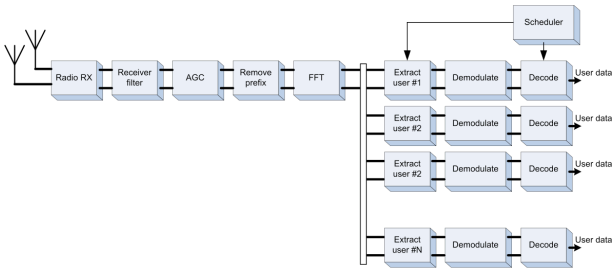

Figure 1: A simplified modular view of the principal functions of the baseband receiver in long term evolution (LTE) RBS [3].

mapping tools, allowing programmers to take the system's real-time properties into account during the optimization process. Therefore, complementary to fully automatized multicore compilers, we are proposing an iterative code parallelization- and mapping tool flow that allows the programmer to tune mapping by:

- analyzing the result of a parallel code map using performance feedback
- giving timing constraints, clustering and core allocation directives as input the tool

Such a tool would allow the programmer early in the development process to explore the run time behavior of the system and to find successively better mappings. We believe that this iterative, machine assisted workflow helps keeping the application software portable and supports the user to make tradeoffs concerning throughput, latency and compliance with real-time constraints on different platforms.

Tool Chain for Iteratively Tuned Code Generation We are designing a tool chain that allows the programmer to specify a multicore architecture (using a configurable machine model), to describe a model of the application (using SDF) and to obtain a parallel intermediate representation (IR) that can be evaluated [4].

To enable portability and code generation for a multitude of processor targets, there is an obvious need for a parallel machine abstraction. A research challenge here is that this model needs to be general enough to capture the architectural structure of a set of processors, while still be detailed enough to be useful for realistic performance analysis. The difficult part to model is contention on shared resources, mainly the network and the global memory. By focusing on software controlled cache machines this problem can be delimited to a network modeling problem. We have developed such a machine model which is subject to experimental evaluation.

To be able to evaluate performance we need an IR that is also easily abstractly computable. In our work, we have proposed one such parallel IR. The IR is a process networks graph in which nodes (cores) and edges (logical communication) are associated with processing costs derived from the application- and the machine model. Interesting research questions here are what kind of feedback information is useful and how the developer can and should interact with the mapping tool using this information.

References

[1] E. A. Lee and D. G. Messerschmitt, "Static Scheduling of Synchronous Data Flow Programs for Signal Processing," *IEEE Transactions on Computers*, vol. C-36, January 1987.

[2] M. I. Gordon, W. Thies, and S. Amarasinghe, "Exploiting Coarse-Grained Task, Data, and Pipeline Parallelism in Stream Programs," in *International Conference on Architectural Support for Programming Languages and Operating Systems*, (San Jose, CA), 2006.

[3] H. Sahlin, "Introduction and overview of LTE Baseband Algorithms." Powerpoint presentation, Baseband research group, Ericsson AB, February 2007.

[4] J. Bengtsson, "A Model Set for Manycore Performance Evaluation Through Abstract Interpretation of Timed Configuration Graphs," Tech. Rep. IDE0850, School of Information Science, Computer and Electrical Engineering, 2008.