# Optimal and Heuristic Code Generation for Explicitly Parallel Processors

Gergö Barany, Alexander Jordan,
Viktor Pavlu, Andreas Krall[1,2]

*Institute of Computer Languages, Vienna University of Technology,*
*Argentinierstrasse 8/E185, 1040 Wien, Austria*

**ABSTRACT**

**We describe our ongoing research within the EPICOpt project aimed at optimal and near-optimal code generation for explicitly parallel processors. On the one hand, this includes the application of techniques from the operations research domain to decrease the solver time for integer linear programming formulations of code generation problems. On the other hand, we want to investigate when and why established heuristics fail, and to develop efficient approximation algorithms and near-optimal techniques that remain computationally feasible even for large problems. To aid our investigation of the performance of various approaches to code generation, we are also investigating novel ways of efficient processor simulation using just-in-time compilation techniques.**

KEYWORDS: code generation; integer linear programming; simulation; just-in-time compilation

## 1 Introduction

Embedded systems have become a prevalent part of our everyday life and it is very unlikely that this trend is going to decline anytime soon. Traditional superscalar techniques require for a 2–3× speedup in performance very roughly about an increase of 80× in area and, maybe even more important, about 12× in power consumption. For numerous mobile applications with critical energy and cost requirements, this is a cost too high to bear. Embedded system designers thus frequently employ application specific accelerators and simple explicit parallel RISC architectures. The latter architectural paradigm—Very Large Instruction Word (VLIW)—originated in the supercomputing domain and gradually found

[1]Authors' e-mail addresses: {gergo, ajordan, vpavlu, andi}@complang.tuwien.ac.at
[2]The work described here is supported by the Austrian Science Fund (Fonds zur Förderung der wissenschaftlichen Forschung) under contract P21842, *Optimal Code Generation for Explicitly Parallel Processors*, http://www.complang.tuwien.ac.at/epicopt/.

its way into today's embedded system architectures, e. g., ST2xx, TI C6xxx, LSI-Starcore. In contrast to superscalar designs, the burden to effectively discover and exploit instruction-level parallelism (ILP) is left to the compiler. The key benefit of this approach is the ability to support large amounts of parallelism using a simple control unit.

Our work builds upon the freely available LLVM compiler framework. LLVM is an industrial strength set of compiler components that has been successfully used to develop various components such as highly optimizing compiler backends, dynamic just in time compilers (JIT), and simulators. Several analyses and transformations that are critical for ILP code generation such as alias analysis, loop unrolling, and loop dependence testing are readily available or currently under active development.

# 2 Code generation

## 2.1 Clustered VLIW code generation

Based on an existing prototype for a VLIW backend in LLVM, we are working on cluster assignment techniques that especially suit architectures with low inter-cluster-communication cost. The first implementation binds instructions to clusters using a straight-forward heuristic that takes place before scheduling. In the coming months we plan to construct an integrated approach that combines cluster assignment, instruction scheduling and spilling minimization. State of the art heuristics shall be contrasted with our own integer linear programming formulation of this problem.

## 2.2 Combined register allocation and instruction scheduling

We are working on a register allocator which can reschedule instructions to reduce register pressure during allocation. We aim to extend this approach from the heuristic linear-scan allocator to the near-optimal PBQP-based register allocator, and to reformulate the rescheduling heuristics as an optimization problem to obtain near-optimal register allocation with (near-)optimal (superblock) rescheduling. We are also investigating optimal register allocation based on integer linear programming, and its interaction with instruction scheduling.

# 3 Simulator Generation

We plan to extend an existing generator for cycle-accurate simulators. Currently, the simulators use a mixed approach based on interpretation and dynamic compilation via the LLVM just-in-time compiler. We are building traces of simulated instructions to increase the performance of generated simulators. We are currently also working on specializing LLVM's existing JIT compilation framework for AMD64 to enable faster instruction-level simulation on that architecture.

# 4 Project status

The EPICOpt project started in October 2009 and is scheduled to last until 2012. The work described above is therefore still in progress, and it is too early to report results.