XLOOPS: Explicit Loop Specialization  
PolyHS: Polymorphic Hardware Specialization

Abstract

Serious physical design issues are breaking down traditional abstractions in computer systems and motivating computer architects to turn to hardware specialization. Resolving the tension between low-end general architectures and more-efficient specialized architectures, particularly within the context of mainstream computing platforms, is one of the grand challenges facing the computer architecture community. This poster discusses two projects that add hardware specialization funded in part by our NSF XPS Award.

XLOOPS: Explicit Loop Specialization

We are pursuing a single ISA heterogeneous architecture called explicit loop specialization (XLOOPS) that transparently integrates general-purpose processors (GPPs) and specialized loop accelerators. XLOOPS supports a variety of inter-iteration data- and control-dependence patterns for both single and nested loops. The XLOOPS hardware/software abstraction requires only lightweight changes to a general-purpose compiler to generate efficient code for loops in applications specifying these behaviors on (1) traditional multithreaded systems with minimal performance impact, (2) specialized microarchitectures to improve performance or energy efficiency, and (3) adaptive microarchitectures that can seamlessly migrate between traditional and specialized execution.

PolyHS: Polymorphic Hardware Specialization

We are also pursuing a less programmable yet still flexible approach based on an observation of how software engineers balance generality and specialization in the very applications we wish to optimize. Software engineers develop carefully crafted libraries of algorithms and data structures that are restricted in terms of input, output, and stored values. Our key idea is to leverage the efficiency that software engineers have already invested and encapsulate it in a general special-purpose hardware (Poly-HS). A GPP augmented with a Poly-HS processor can be seen as a programmable microprocessor that can execute polymorphic code in addition to traditional code.

XLOOPS Instruction Set

The XLOOPS instruction set is carefully designed to enable efficient execution on both traditional general-purpose processors (GPPs) and specialized microarchitectures. The XLOOPS instructions encode the notion of a parallel loop body and the inter-iteration dependence patterns as shown below.

Code and Assembly Examples

XLOOPS Cycle-Level Evaluation

We modified a g++-LLVM 1.0 simulation framework to model both in-order and out-of-order processors augmented with an LPSU. We compare XLOOPS to three baseline GPPs: a simple single-issue in-order processor (s), a modest 8-way out-of-order superscalar processor (oos2), and an aggressive 8-way out-of-order superscalar processor (ooos4). We augmented each baseline GPP with an LPSU to create three XLOOPS configurations: ooos4, ooos4-0, and ooos4-0.ooos4.

Performance Results

We observe that specialized execution always benefits the in-order processor. For a total of 25 application kernels, specialized execution performs better for 18 kernels compared to ooos4, and performs better for 12 kernels compared to ooos4-0.

PolyHS Project Overview

The following code illustrates a simple polymorphic algorithm for merging two ordered lists.

```
merge(X=0, i=0; i<N)
    if (X<i) { X=i; i=0 }
    if (i<X && i<n) { X=i; i=n }
```

We assume software engineers have already partitioned the application into two parts: (1) a library of polymorphic algorithms and data structures; and (2) the actual application-specific code which uses this library. Our PolyHS design methodology involves partitioning an algorithm into a hardware controller, synthesis, compilation, and time systems.

PolyHS Design Methodology

We assume software engineers have already partitioned the application into two parts: (1) a library of polymorphic algorithms and data structures; and (2) the actual application-specific code which uses this library. Our PolyHS design methodology involves partitioning an algorithm into a hardware controller, synthesis, compilation, and time systems.

PolyHS Architecture

A PolyHS chip contains general-purpose processors (GPPs) along with a diverse selection of heterogeneous Poly-Tiles. Each Poly-Tile includes a customized LPSU, memory system, multiple Poly-ASUs, and multiple Poly-DSUs.

PolyHS Preliminary Case Study

Results for running given application kernel on appropriate algorithm- or data-structure-specific unit. Percentage numbers compare the result to the CP running the application kernel completely in software.

---

Contact: Christopher Batten, 320 Rhodes Hall, Ithaca, NY 14853, cbatten@cornell.edu