XLOOPS: Explicit Loop Specialization

Shreesha Srinath, Berkin Ilbeyi, Mingxing Tan, Gai Liu, Zhiru Zhang, and Christopher Batten
School of Electrical and Computer Engineering, Cornell University, Ithaca, NY
{ss2783,bi45,mt453,gl387,zhiruz,cbatten}@cornell.edu

Hardware specialization is becoming an increasingly common technique to enable improved performance and efficiency in spite of the diminished benefits of technology scaling. Meanwhile, computer architects have long realized the importance of focusing on the key loops that often dominate application performance. These two trends have led to a diverse array of loop-level specialized hardware, such as SIMD engines, vector processors, GPUs, and custom accelerators. **A key research challenge for these heterogeneous engines involves creating clean hardware/software abstractions that are highly programmable, yet still enable efficient execution on both traditional and specialized microarchitectures.**

To address this challenge, this poster will present ongoing work on a new approach called *explicit loop specialization (XLOOPS)* based on the idea of elegantly encoding loop dependence patterns in the instruction set [1]. The XLOOPS instruction set is formed by extending a general-purpose instruction set with a few new instructions to encode the notion of a parallel loop body and the inter-iteration control- and data- dependence patterns. *Inter-iteration control-dependence patterns* include loops that terminate based on comparing an induction variable to a loop-invariant fixed bound or a data-dependent exit condition. XLOOPS can also handle more challenging control-dependence patterns found in irregular worklist-based algorithms. In such algorithms, a loop induction variable is compared to a dynamic bound that is monotonically increased during the loop execution. *Inter-iteration data-dependence patterns* include loops with no inter-iteration dependences and loops with inter-iteration dependences encoded through registers and/or memory. XLOOPS can also handle more challenging data-dependence patterns often found in graph algorithms where each iteration manipulates a shared data structure with data-dependent conflicts. In such algorithms, the iterations can be executed in any order as long as their updates to memory appear atomic to the other iterations.

The XLOOPS hardware/software abstraction requires only lightweight changes to a general-purpose compiler. The compiler front-end uses explicitly inserted pragma annotations to determine which loops to encode using the XLOOPS instruction set. Compiler algorithms for mid-level optimization passes, and back-end algorithms for instruction scheduling, register allocation, and code generation can work out-of-the-box. The XLOOPS compiler includes analysis passes to determine the type of inter-iteration data-dependence pattern. Register-dependence testing is implemented by analyzing the use-definition chains through the PHI nodes and memory-dependence testing is implemented using well-known dependence techniques such as ZIV/SIV/MIV tests.

XLOOPS binaries can be executed efficiently on: (1) *traditional microarchitectures* with minimal performance impact, (2) *specialized microarchitectures* to improve performance and/or energy efficiency, and (3) *adaptive microarchitectures* that can seamlessly migrate loops between traditional and specialized execution to dynamically trade-off performance vs. energy efficiency. Traditional execution on general purpose processors (GPPs) can be supported by simply treating an XLOOPS instruction as conditional branch instruction. Specialized execution is supported by augmenting a GPP with a loop-pattern specialization unit (LPSU). An LPSU contains a lane management unit (LMU) and a number of decoupled lanes for executing iterations in parallel. The LMU is responsible for interacting with the GPP to receive work and dynamically distribute this work across all lanes. Each lane is similar to an inorder processor with a few important differences: a lane index queue for storing pending loop indices from the LMU, a small instruction buffer for storing the loop body, and dynamic arbitration to share long-latency functional units and the data memory port across all lanes. Specialized execution occurs in two phases: a scan phase where the GPP configures the LPSU instruction buffers and registers, and an execution phase where the LPSU executes the parallel loop to completion. The XLOOPS abstraction allows for adaptive execution that enables applications that perform worse with specialized execution to automatically migrate to the GPP for increased performance at reduced energy efficiency.

We are evaluating XLOOPS using a vertically integrated research methodology. We have implemented an LLVM-based compiler framework that can compile pragma annotated application kernels drawn from MiBench, PolyBench, Problem Based Benchmark Suite, and our own custom benchmarks. We have modified the gem5 simulation framework integrated with McPAT-1.0 energy models, to model both in-order and out-of-order processors augmented with an LPSU. We have also implemented a simple register-transfer-level XLOOPS microarchitecture capable of specialized execution for unordered concurrent loops. We have used this model and a commercial ASIC CAD toolflow to estimate area, energy, and timing. Using specialized execution, XLOOPS is able to achieve 2.5× or higher speedup at similar or better energy efficiency for most application kernels compared to a simple single-issue in-order processor with only 40% area overhead. Compared to aggressive two- and four-way out-of-order processors, XLOOPS is able to achieve 1.5-3× improvement in energy efficiency while having speedups in the range of 1.25–2.5× on most application kernels.

XLOOPS is an elegant approach that unifies ideas from transactional memory, hardware task-scheduling, and data parallel accelerators with a novel abstraction that can be mapped to traditional, specialized, and adaptive architectures.

References

[1] S. Srinath, B. Ilbeyi, M. Tan, G. Liu, Z. Zhang, and C. Batten. Architectural Specialization for Inter-Iteration Loop Dependence Patterns. *Int'l Symp. on Microarchitecture (MICRO)*, Dec 2014 (to appear).