Modelling Modern Microarchitectures Using CASL

Ed Walters, Eliot Moss, Trek Palmer, Tim Richards, Chip Weems
University of Massachusetts Amherst
Department of Computer Science
ekw, moss, trekp, richards, weems @cs.umass.edu
Motivation

What do these have in common?

- Intel terascale research chip
- Cell BE
- GPU
- TRIPS
Motivation

- What do these have in common?
  - Intel terascale research chip
  - Cell BE
  - GPU
  - TRIPS
  - Radically unlike existing architectures
Motivation

- What do these have in common?
  - Intel terascale research chip
  - Cell BE
  - GPU
  - TRIPS
- Radically unlike existing architectures
- Attempt to exploit more parallelism
Motivation

- What do these have in common?
  - Intel terascale research chip
  - Cell BE
  - GPU
  - TRIPS
- Radically unlike existing architectures
- Attempt to exploit more parallelism
- Designed in the absence of a compiler
Why?

- Compilers are expensive and time-consuming to build
- Architects need to explore design space without waiting for compiler on each iteration
- Most changes to the hardware simulator should have compiler optimizations adjusted (not to mention JIT)
CoGenT Overview

- ISA ADL
- \( \mu \text{Arch ADL} \)
- OS Spec

CoGenT
- Parser
- Collector
- Translator
- Generator

- Assem/Disassem
- Optimizing Compiler
- Functional Simulator
- Timing Simulator
- Instrument Harness
Problems

- Architectures becoming more complex, more parallel, more dynamic, harder to evaluate
- Most ADLs describe microarchitecture as hardware units arranged in a graph
  - Hard to express more complex architectures
  - Complicates instruction-focused instrumentation
- ...and treat instructions as data
  - Limits flexibility of operation flow management
  - Obscures opportunities for optimization
- Need higher semantics in structuring pipelines
  - Nonlinear, coordinated parallel, redundant
CASL Overview

- Microarchitecture description language
- Coordinated with CISL ISA DL
- Mixed structural/behavioral ADL
  - For generating simulators, not hardware
  - Describe structure, behavior, timing, power
    - Structure is component-based
    - Behavioral description: integrated, Java-style syntax
      - Timing annotation from simple delay to complex parallel models
- Language specification now fixed
  - Currently parsing to IR, and type checking
Strands

- Operation packets (not just instructions)
  - Carry control through component graph
- Path determined by initial condition of strand and interaction with components
- Can add and remove information in a strand as it proceeds through the graph
  - e.g., decoding, memory requests, predication
  - Also for instrumentation, optimization
- Can be dynamically created and destroyed
  - e.g., instruction fetch, parallel operation, speculation, redundant operations, event factories
- Distributed control across graph
Pipeline Support

- Pipelines are a common microarchitectural idiom
- Higher level notation for specific types of graphs
- Stages, buffers, connectors with specific semantics
- Designed to carry strands
  - Strands visit stages, are stored in buffers, and are routed by connectors
  - Stages provide computational behavior
- Connectors may be 1-1, 1-many, many-1, and can have arbitrary processing to manage routing
- Buffers can have arbitrary processing to manage constraints and capacity
Strand Example

```c
strang sampleDLXInst {
    reg_index_t read_reg1; ...
    word32_t inst_word;
    sampleDLXInst(word32_t raw) ...
    action path {
        Fetch.visit();
        Decode.visit(); ...
        Writeback.visit(); }
}

action prefetch {
    word32_t raw_inst =
        Icache.fetch(PC);
    sampleDLXInst inst =
        new sampleDLXInst(raw_inst);
    ...
}
```
Summary

- CoGenT is designed to address evaluation of next-gen architectures and languages.
- CASL provides a mixed ADL to model microarchitecture separate from ISA.
- Uses modern language constructs to enhance expressiveness.
- Strands enable more flexible control of instruction flow and instrumentation.
- Flexible pipeline support.
Realistic hardware/software simulation is critical for developing next generation systems, and is also very hard.