A Common Machine Language for Communication-Exposed Architectures

Bill Thies, Michal Karczmarek, Michael Gordon, David Maze and Saman Amarasinghe

MIT Laboratory for Computer Science

HPCA Work-in-Progress Session, February 2002
A Common Machine Language for Communication-Exposed Architectures

Language Designers Have Been Ignoring Architects

Bill Thies, Michal Karczmarek, Michael Gordon, David Maze and Saman Amarasinghe

MIT Laboratory for Computer Science

HPCA Work-in-Progress Session, February 2002
Back in The Good Old Days...

- Architecture: simple von-Neumann
- "Common Machine Language": C
  - Abstracts away idiosyncratic differences
    - Instruction set
    - Cache configuration
  - Exposes common properties
    - Program counter
    - Monolithic memory
  - Efficient implementations on many machines
  - Portable: everyone uses it
Programming Language Evolution

- C
- C++
- Java
Programming Language Evolution

![Graph showing the evolution of language effectiveness and Moore's Law over time.](image-url)
Languages Have Not Kept Up

• Two choices:
  • Develop cool architecture with complicated, ad-hoc language
  • Bend over backwards to support old languages like C/C++
Evidence: Superscalars

- Huge effort into improving performance of sequential instruction stream
- Complexity has grown unmanageable
- Even with 1 billion transistors on a chip, what more can be done?

- Pipelining
- Out-of-Order Execution
- Renaming
- Branch Prediction
- Prefetching
- Speculative Execution
- Value Prediction
A New Era of Architectures

• Facing new design parameters
  - Transistors are in excess
  - Wire delays will dominate

• “Communication-exposed” architectures
  - Explicitly parallel hardware
  - Compiler-controlled communication
  - e.g. RAW, Smart Memories, TRIPS, Imagine, the Grid Processor, Blue Gene
A New Common Machine Language

• Should expose shared properties:
  - Explicit parallelism (multiple program counters)
  - Regular communication patterns
  - Distributed memory banks
  - No global clock

• Should hide small differences:
  - Granularity of computation elements
  - Topology of network interconnect
  - Interface to memory units

⇒ C does not qualify!
The StreamIt Language

• A high-level language for communication-exposed architectures
• Computation is expressed as a hierarchical composition of independent filters
The StreamIT Language

• A high-level language for communication-exposed architectures
• Computation is expressed as a hierarchical composition of independent filters
• Features:
  - High-bandwidth channels
  - Low-bandwidth messaging
  - Re-initialization
The StreamIt Compiler

• We have a compiler for a uniprocessor
  - Performs comparably to C++ runtime system
The StreamIt Compiler

• We have a compiler for a uniprocessor
  - Performs comparably to C++ runtime system
• Working on a backend for RAW
  - Fission and fusion transformations
  - Many optimizations in progress
The StreamIt Compiler

• We have a compiler for a uniprocessor
  - Performs comparably to C++ runtime system

• Working on a backend for RAW
  - Fission and fusion transformations
  - Many optimizations in progress

• Goal: High-performance, portable language for communication-exposed architectures
For more information, see:
http://cag.lcs.mit.edu/streamit/

Thank you!