Stories, Not Words: Abstract Datatype Instruction Sets

Martha Kim
Columbia University

Workshop on New Directions in Computer Architecture
6/5/2011
The Utilization Wall

- Exponential decrease in percentage of transistors that can be operated at full frequency.

- In 45nm TSMC process, 7% of 300mm² die can operate at full frequency

- In 32nm, 3.5%

Specialization Is a Promising Approach

- R. Hameed et al., “Understanding sources of inefficiency in general-purpose chips,” ISCA ’10
- G. Venkatesh et al., “Conservation cores: reducing the energy of mature computations,” ASPLOS ’10
- V. Govindaraju, C. Ho, and K. Sankaralingam, “Dynamically Specialized Datapaths for energy efficient computing,” HPCA ’11
- R. Hou et al., “Efficient data streaming with on-chip accelerators: Opportunities and challenges,” HPCA ’11
An Ideal Accelerator System

- High Performance
- Low Energy
- Easy to Program
- Software Portability
Accelerator Design Processes

Application
Accelerator Design Processes

Application

Microarch.
Accelerator Design Processes

Application

Microarch.

Arch.
Accelerator Design Processes

Application → Microarch. → Arch. → Application

Application → Arch. → Microarch. → Application
Accelerator Design Processes

Application → Microarch. → Arch.

Application → Arch. → Microarch.

Application → Arch.
Accelerator Design Processes

Application → Microarch. → Arch.

Application → Arch. → Microarch.

Application → Arch. → Microarch.
Extending Software Abstractions to Hardware

Application
Libraries
Machine Code
Micro-ops
Execution core
Caches
Memory
Extending Software Abstractions to Hardware

- Application
- Libraries
- Machine Code
- Micro-ops
- Execution core
- Caches
- Memory
Extending Software Abstractions to Hardware

Application
Libraries
Machine Code
Micro-ops
Execution core
Caches
Memory

Raise HW/SW interface
Extending Software Abstractions to Hardware

Raise HW/SW interface

Extend interfaces from libraries to hardware

Application
Libraries
Machine Code
Micro-ops
Execution core
Caches
Memory
Extending Software Abstractions to Hardware

- Application
- Libraries
- Machine Code
- Micro-ops
- Execution core
- Caches
- Memory

Raise HW/SW interface

Extend interfaces from libraries to hardware

Exploit interfaces with specialized hardware
Abstract Datatype Processing

SW

Arch

UArch
Abstract Datatype Processing

class HashTable

put(k,v)  v get(k)

class HashTable

SW

Arch

UArch

Sunday, June 5, 2011
Abstract Datatype Processing

class HashTable

put(k,v)  v get(k)

put $h, $k, $v  get $h, $k, $v
Abstract Datatype Processing

class HashTable

put(k,v)  get(k)

put $h, $k, $v  get $h, $k, $v

Hash Table Processor

SW

Arch

UArch
Compilation & Execution

Sequence Labeling

SparseVec  HashTable

Dispatch

GP  SV  HT

Sunday, June 5, 2011
The Software Fallback

Dispatch

GP

SV
An Ideal Accelerator System

- High Performance
- Low Energy
- Easy Use - align hardware interfaces with those software is already using
- Portability - software fallback plan
Enforcing Data Encapsulation

CPU

Sparse Vector Accelerator

set $v$, $i$, $x$
get $v$, $i$, $x$
dot $v_1$, $v_2$, $p$
Enforcing Data Encapsulation

Sparse Vector Accelerator

set $v, i, x$

get $v, i, x$

dot $v_1, v_2, p$

CPU

v

i

x
Enforcing Data Encapsulation

Sparse Vector Accelerator

CPU

set $v, i, x$
get $v, i, x$
dot $v1, v2, p$

v i x
Enforcing Data Encapsulation

CPU

Sparse Vector Accelerator

set $v, i, x$
get $v, i, x$
dot $v1, v2, p$

I C D

v i x

I A B C D
Specialized Caching for Sparse Vectors

Hit Rate vs. Storage Capacity (B)

- Standard Cache
- VecStore

Sunday, June 5, 2011
Key Reuse in Hash Tables

Pct. Hash Operations

Number of Keys

LZW Compress
Parser

Sunday, June 5, 2011
Key Reuse in Hash Tables

Number of Keys

Pct. Hash Operations

LZW Compress

Parser
Key Reuse in Hash Tables

LZW Compress  Parser

386 entry table  26% of table  99% of dynamic accesses
Key Reuse in Hash Tables

LZW Compress  Parser

386 entry table  26% of table  99% of dynamic accesses

94K entry table  .1% of table  75% of dynamic accesses

Pct. Hash Operations

Number of Keys

Sunday, June 5, 2011
Exploiting Key Reuse

- Compress HTX-M
- Parser HTX-M Accesses
- Compress HTX-M Entrystore Accesses
- Parser HTX-M Entrystore Accesses

Hash Table Accelerator (HTX)
Exploiting Key Reuse

Hash Table Accelerator (HTX)

- put $h, k, v$
- get $h, k, v$

HTX-M

- Compress HTX-M
- Parser HTX-M Accesses
- Compress HTX-M Entrystore Accesses
- Parser HTX-M Entrystore Accesses

Reduction in HTX-M Accesses vs. Cache Capacity

Sunday, June 5, 2011
Summary

Extend software’s encapsulated datatypes into hardware accelerators

Natural alignment with standard software engineering

Accelerator utility on all applications that use a particular type

A software fallback that ensures portability

Aggressive optimization of computation and data movement
Research Challenges

- What are the appropriate types to target? What is the lower bound in complexity? Is there a max number of types a hardware system can support?

- How do I implement polymorphism efficiently? (e.g., priority queue with arbitrary types and user-defined sort function)

- How do I optimize enforcement of data encapsulation? (copy-on-read is conservative)

- Can the execution model support parallel execution?

- What is type-specific coherence like? Simpler? Uglier?

- What is the appropriate system-level resource allocation between general and specialized? Between different types?
Thank You