WADE SHEN

PROGRAM MANAGER
DARPA/MTO
SOFTWARE DEFINED HARDWARE

INTRODUCTION

DISTRIBUTION STATEMENT A. Approved for public release. Distribution is Unlimited.
SOFTWARE DEFINED HARDWARE PROGRAM GOAL

Build runtime reconfigurable hardware and software that enables near ASIC performance (within 10x) without sacrificing programmability for data-intensive algorithms.
THE PROBLEM

PROCESSOR DESIGN TRADES

- Math/logic resources
- Memory (cache vs. register vs. shared)
- Address computation
- Data access and flow

THE PROBLEM: OPTIMAL HW CONFIGURATION DIFFERS ACROSS ALGORITHMS

No one hardware efficiently solves all problems well
**SDH: Runtime Optimization of Software and Hardware for Data Intensive Computation**

**TODAY:**  HW DESIGN SPECIALIZATION

- One chip per algorithm
- Chip design expensive
- Not reprogrammable
- Can’t take advantage of data-dependent optimizations

**TOMORROW:**  RUNTIME OPTIMIZATION OF HARDWARE AND SOFTWARE

- One chip many applications
- One time design cost
- Reprogrammable via high-level languages
- Data-dependent optimization (10-100x)

*DISTRIBUTION STATEMENT A. Approved for public release. Distribution is Unlimited.*
SOFTWARE-DEFINED HARDWARE

High-level program

Dynamic HW/SW compilers for high-level languages (TA2)
1. Generate optimal configuration based on static analysis code
2. Generates optimal code
3. Re-optimize machine code and processor configuration based on runtime data

Reconfigurable processors (TA1)
1. Reconfiguration times: 300 - 1,000 ns
2. Re-allocatable compute resources – i.e. ALUs for address computation or math
3. Re-allocatable memory resources – i.e. cache/register configuration to match data
4. Malleable external memory access – i.e. reconfigurable memory controller

DISTRIBUTION STATEMENT A. Approved for public release. Distribution is Unlimited.
SDH PERFORMERS

**TA1**
- Intel
- Qualcomm

**TA1 + TA2**
- NVIDIA
- Princeton University
- Stanford University
- University of Michigan
- University of Washington

**TA2**
- GT
- STR

DISTRIBUTION STATEMENT A. Approved for public release. Distribution is Unlimited.
LUCA CARLONI

DEPARTMENT OF COMPUTER SCIENCE
COLUMBIA UNIVERSITY

DISTRIBUTION STATEMENT A. Approved for public release. Distribution is Unlimited.
DECADES: DEEPLY-CUSTOMIZED ACCELERATOR-ORIENTED DATA SUPPLY SYSTEMS SYNTHESIS

DISTRIBUTION STATEMENT A. Approved for public release. Distribution is Unlimited.
Modern computer systems are increasingly heterogeneous

- Accelerator-oriented parallelism to meet aggressive performance and power targets

As accelerators have sped up compute performance, the main challenge is data supply

- Key bottlenecks lie in memory and communication overheads associated with supplying data to specialized accelerators
- Different applications have distinct data supply needs
DECADES: A VERTICALLY-INTEGRATED APPROACH

Language and Compiler Support
(M. Martonosi)
- Enhance data locality
- Optimize spatial mapping of threads
- Enable in-memory computing

Very Coarse-Grained Reconfigurable Tile-Based Architecture
(L. Carloni)
- Coarser than CGRA
- Three classes of reconfigurable tiles
- Reconfigurable interconnection network
- Reconfigurable in-memory computing

Multi-Tiered Demonstration Strategy
(D. Wentzlaff)
- Scalable full-system simulation
- Multi-FPGA emulation infrastructure
- 225-tile DECADES chip prototype

DISTRIBUTION STATEMENT A. Approved for public release. Distribution is Unlimited.
Heterogeneity meets coarse reconfigurability

DECADES PLATFORM ARCHITECTURE

DECADES Core Tile
- DECADES Monitor and Run-Time Reconfiguration Shim
- Per-Tile Configurable On-chip Memory
- Configurable Core Pipeline
- L1 Cache
- Data Supply / Compute Threads
- L1 Cache

Configurable Interconnect Shims

DECADES Monitor and Run-Time Reconfiguration Shim
- Specialized, Configurable Data Supply / Compute Accelerator
- Configurable Interconnect Shims

DECADES Intelligent Storage
- Across-Chip Configurable On-Chip Memory
- Near-Memory Computation
- Configurable Pattern-Based Prefetcher
- Configurable Interconnect Shims

DECADES Accelerator Tile

DECADES Core Tile
- DECADES Monitor and Run-Time Reconfiguration Shim
- Per-Tile Configurable On-chip Memory

DECADES Accelerator Tile

DECADES Core Tile
- DECADES Monitor and Run-Time Reconfiguration Shim
- Per-Tile Configurable On-chip Memory

DECADES Accelerator Tile

DECADES Core Tile

DECADES Accelerator Tile

DECADES Core Tile

DECADES Accelerator Tile

DECADES Core Tile

FPGAs: Off-Chip Memory System with In-Memory Computation

DISTRIBUTION STATEMENT A. Approved for public release. Distribution is Unlimited.
Application-Specific Memory Hierarchy for Bandwidth-Bound Graph Analytics

- Customized memory hierarchy to minimize off-chip memory access traffic \([3x \text{ reduction}]\)
- Ease the design/use of accelerators
- Dataflow pipeline based on high-level abstraction eases the programming and enables hardware reuse for different graph applications
- Specialized HW accelerator for graph analytics successfully achieves \(~3x\) speedup and 50x+ energy saving compared to state-of-the-art software framework on 32-core CPU

[Ham et al., MICRO-49, 2016. IEEE Micro Top Picks Honorable Mention]
• Computations mapped onto core tiles or available accelerator tiles
  • Each tile is wrapped in monitor/reconfiguration shim
• Dynamic reconfiguration of supply-compute decoupling, power-performance tradeoffs, and interconnect
INTELLIGENT DATA MANAGEMENT

- Specialization #1: Map applications onto mix of compute tiles and intelligent storage (IS) tiles
- Specialization #2: Select and configure appropriate storage features within IS
  - Configurable memory banks + address and prefetching features
  - Simple near-SRAM ALU

DECADES Intelligent Storage Tile

DISTRIBUTION STATEMENT A. Approved for public release. Distribution is Unlimited.
PRIOR WORK: COHERENCE DOMAIN RESTRICTION ON FLEXIBLE MEMORY

- Flexible memory system on top of cache coherent system
  - Enables the exact minimal communication needed
  - Build incoherent coherent domains
- Restriction on application- or page-level
  - Improves performance
    - Shorter network on-chip distances
    - Less interfering memory coherence traffic
  - Reduces energy
    - Fewer on-chip network links need to be transited
    - Less area dedicated to tracking cache line sharers
  - Reduces area
    - Track fewer sharers on large configurations

[Fu et al, MICRO 2015]
- Compiler Analysis and Support for memory hierarch specialization
  - Bandwidth optimizations through cache optimizations and locality/granularity tailoring
  - Latency tolerance through decoupling
- Build on DeSC LLVM compiler infrastructure

[Ham/Aragon/Martonosi, MICRO-48, 2015]
PRIOR WORK: EMBEDDED SCALABLE PLATFORMS

- **Flexible Tile-Based Architecture**
- **System-Level Design Methodology**

SoC Design Productivity

\[\text{In the span of 1 month:}\]

- 21: student teams
- 661: improved designs
- 32: avg. number of improved designs per team
- 1.5: avg. number of new designs per team/day
- 99: Pareto curve changes
- 11: final number of Pareto-optimal designs
- 26x: performance range
- 10x: area range

[Carloni, DAC 2016]
**EMULATION & PROTOTYPING**

- Take DECADES architecture to FPGA
- Continued design refinement throughout program

- Prototype chip to de-risk architecture
- Recent 25-core manycore system built by our team

- Multi-FPGA emulation infrastructure

DISTRIBUTION STATEMENT A. Approved for public release. Distribution is Unlimited.
SUMMARY & IMPACT

Language/Compiler/Runtime:
• Latency: >4X per thread performance benefits from memory data supply decoupling
• Bandwidth: Granularity management and multiplicative outer-loop parallelism up to bandwidth limit
• Total of 50X over single-thread from software

Configurable Hardware Platform:
• Hardware speedups from accelerators for address calculation, memory fetch, or compute
• Fine-grained, low-overhead measurements drive adaptation and module depowering
• 10-20X multiplicative power/performance benefits

DISTRIBUTION STATEMENT
A. Approved for public release. Distribution is Unlimited.
TECHNOLOGY TRANSFER PLANS

- Outputs:
  - Software ecosystem
  - Chip design
  - FPGA emulation system

- Technology transfer plans:
  - Release of software, hardware, and data where possible
  - Commercialization and licensing

- Leverage extensive past experience:
  - Widely-used open-source software (Wattch, *Check tools, scalable QEMU)
  - Patents licensed to major companies (Power-efficient ALUs)
  - Technology transferred from academia to startups (Tilera)
  - Open-Source Hardware (OpenPiton)
TOWARDS A COMPUTER DESIGN RENAISSANCE

• The end of silicon dimensional scaling and the rise of heterogeneous reconfigurable computing bring an opportunity for a Computer Design Renaissance

• ...by supporting the creativity of application developers to realize innovative architectures, chips, systems and products

• ...through richly reconfigurable substrates and intelligent compilation and mapping
MITCHELL: SDH: A DECLARATIVE DATAFLOW FRAMEWORK AND REAL-TIME RESOURCE OPTIMIZATION ENGINE
TEAM

• Systems & Technology Research
  • PI: Brad Gaynor
  • Data-dependent, dynamic resource optimization

• Purdue University
  • Suresh Jagannathan
  • MITCHELL declarative dataflow language & compiler

• Northeastern University
  • Dave Kaeli
  • OpenCL compiler & runtime
OBSERVATIONS

• For many machine learning tasks, getting good answers fast (vs. exact answers eventually) is desirable.

• Need to understand an application’s algorithmic structure and operational semantics to model resource constraints that can direct hardware resource reconfiguration.

• MITCHELL
  • Fine-grained resource-aware compilation framework that
    • Exploits approximation and incrementalization to
    • Dynamically allocates SDH hardware resources at runtime to efficiently compute solutions within user-specified error bounds
EXPOSE COMPONENT-LEVEL CONTROL AND DATA DEPENDENCIES IN ML & GA APPLICATIONS

• A typical dataflow pipeline for a machine learning (ML) application yields component-based asynchronous dataflow-style communication.

• MITCHELL is a declarative dataflow language for efficiently coding ML and graph analytics (GA) applications.

EXAMPLE

```ml
let graph = Mitchell.Graph(Dense)
let O = graph.edge()
let Reader (edge I <input>) = emit (T, Transform input())
let Curate (edge T <t-input>, C <Rank>) = emit (O, Rank (filter Analyze t-input))
let main (input) = emit (C, Mitchell.PageRank);
emit (I, Parse(graph, input))
```

INPUT
- Parse
  - Transform
    - Curate
      - PageRank
        - OUTPUT
EXPLOIT APPROXIMATION WITH ALGORITHMIC OPTIMIZATIONS

- MITCHELL provides a restricted set of native data types appropriate for machine learning and graph analytics applications.

- The MITCHELL compiler generates program variants optimized for different data-dependent regions of operation by exploiting incrementalization and approximation.

- Enlarges possible reconfigurability options by increasing allowed behaviors.

- Variants are substituted at runtime based on the data being processed and cost models.
TARGET THE SDH HARDWARE FABRIC WITH OPENCL

• Compile MITCHELL to OpenCL
  • OpenCL supports a C-like language targeting a range of devices including CPUs, GPUs, DSPs and FPGAs

• The OpenCL compiler performs operational optimizations (e.g., layout, caching, etc.) relevant to SDH reconfiguration parameters

• An OpenCL runtime controls program execution and SDH TA1 hardware configurations
SDH HARDWARE RESOURCE OPTIMIZATION

- Closed-loop, dynamic optimization of the SDH TA1 hardware is based on efficient, runtime characterization of the data being processed
- Applications instrumented with low-overhead data sensors to estimate the data-dependent regions of performance at runtime
- Graph motifs and properties are estimated using sub-graph samplers
- An adaptive reconfigurable control system performs hardware/software resource optimization via the OpenCL runtime.