Towards Accelerator-Rich Architectures and Systems

Zhenman Fang, Postdoc

Computer Science Department, UCLA

Center for Domain-Specific Computing

Center for Future Architectures Research

https://sites.google.com/site/fangzhenman/
The Trend of Accelerator-Rich Chips

Die photo of Apple A8 SoC
[www.anandtech.com/show/8562/chipworks-a8]

Specialized Accelerators, e.g., audio, video, face, imaging, DSP, ...

Increasing # of Accelerators in Apple SoC (Estimated)

Maltiel Consulting estimates
Harvard’s estimates
[Shao, IEEE Micro’15]
The Trend of Accelerator-Rich Cloud

Cloud service providers begin to deploy FPGAs in their datacenters

Field-Programmable Gate Array (FPGA) accelerators

- Reconfigurable commodity HW
- Energy-efficient, a high-end board costs ~25W

2x throughput improvement! [Putnam, ISCA'14]
The Trend of Accelerator-Rich Cloud

Cloud service providers begin to deploy FPGAs in their datacenters

Accelerators are becoming 1st class citizens

- Intel expectation: 30% datacenter nodes with FPGAs by 2020, after the $16.7 billion acquisition of Altera

[Putnam, ISCA'14]
Post-Moore Era: Potential for Customized Accelerators

Accelerators promise 10X -1000x gains of performance per watt by trading off flexibility for performance!

Source: Bob Broderson, Berkeley Wireless group

Source: ISSCC Proceedings
Challenges in Making Accelerator-Rich Architectures and Systems Mainstream

How to characterize and accelerate killer applications?

How to efficiently integrate accelerators into future chips?
- E.g., a naïve integration only achieves 12% of ideal performance [HPCA'17]

“Extended” Amdahl’s law: \( \text{overall\_speedup} = \frac{1}{\frac{\text{kernel\%}}{\text{acc\_speedup}}} + (1 - \text{kernel\%}) + \text{integration} \)

- Accelerator
- CPU
- Integration overhead
Challenges in Making Accelerator-Rich Architectures and Systems Mainstream

How to characterize and accelerate killer applications?

How to efficiently integrate accelerators into future chips?
- E.g., a naïve integration only achieves 12% of ideal performance [HPCA'17]

How to deploy commodity accelerators in big data systems?
- E.g., a naïve integration may lead to 1000x slowdown [HotCloud'16]

How to program such architectures and systems?
Overview of My Research

1. Application Drivers
   - Workload characterization and acceleration

3. Accelerator-Rich Systems
   - Accelerator-as-a-Service (AaaS) in cloud deployment

2. Accelerator-Rich Architectures (ARA)
   - Modeling and optimizing CPU-Accelerator interaction

4. Compiler Support
   - From many-core to accelerator-rich architectures
Dimension #1: Application Drivers

- **image processing [ISPASS'11]**
  - Analysis and combination of task, pipeline, and data parallelism
  - 13x speedup on 16-core CPU
  - 46x speedup on GPU

- **deep learning [ICCAD'16]**
  - Caffeine: FPGA engine for Caffe
  - 1.46 TOPS for 8-bit Conv layer
  - 100x speedup for FCN layer
  - 5.7x energy savings over GPU

- **genomics [D&T'17]**
  - 2.6x speedup for in-memory genome sort (Samtool)
  - Record 9.6GB/s throughput for genome compression on Intel-Altera HARPv2;
  - 50x speedup over ZLib

How do accelerators achieve such speedup?
Dimension #1: Application Drivers

image processing [ISPASS'11]  
deep learning [ICCAD'16]  
genomics [D&T'17]

#1: Caching  
#2: Customized pipeline  
#3: Precision customization  
#4: Double buffer  
#5: DRAM re-organization

Kernel: Convolutional Matrix-Multiplication

E.g., convolutional accelerator on-chip

\[ O[m][i][j] = \sum_{n=0}^{N} \sum_{r=0}^{K_1} \sum_{s=0}^{K_2} W[m][n][r][s] \times X[n][i+r][j+s] \]

Input 0  Input 1

Weight 0  Weight 1  Weight 2  Weight 3

Output 0  Output 1

DRAM

Parallelization

#1: Caching
#2: Customized pipeline
#3: Precision customization
Dimension #1: Application Drivers

- Image processing [ISPASS'11]
- Deep learning [ICCAD'16]
- Genomics [D&T'17]

- Results collected on Alpha Data PCIe-7v3 FPGA board
- Programmed in Xilinx High-Level Synthesis (HLS)

![Chart showing GFLOPS values for various components and configurations.](chart.png)
Dimension #2: Accelerator-Rich Architectures (ARA)

Overview of Accelerator-Rich Architecture
Dimension #2: Accelerator-Rich Architectures (ARA)

ARA Modeling:
✓ PARADE simulator: gem5 + HLS [ICCAD'15]
✓ Fast ARAPrototyper flow on FPGA-SoC [arXiv'16]

PARADE is open source: 
http://vast.cs.ucla.edu/software/parade-ara-simulator

ARA Optimization:
✓ Sources of accelerator gains [FCCM'16]
✓ CPU-Acc co-design: address translation for unified memory space, 7.6x speedup, 6.4% gap to ideal [HPCA'17 best paper nominee]
✓ AIM: near memory acceleration gives another 4x speedup [Memsys'17]

Multicore Modeling:
✓ Transformer simulator [DAC'12, LCTES'12]

More information in ISCA'15 & MICRO'16 tutorials:
http://accelerator.eecs.harvard.edu/micro16tutorial/
Dimension #3: Accelerator-Rich Systems

Big data application developer (e.g., Spark)

Cloud service provider w/ accelerator-enabled cloud

Accelerator designer (e.g., FPGA)

Easy and efficient accelerator invocation and sharing

Blaze prototype: 1 server w/ FPGA
~3 CPU servers

Accelerator-as-a-service

Blaze works with Spark and YARN and is open source:
https://github.com/UCLA-VAST/blaze

CPU-FPGA platform choice [DAC'16]:
1) mainstream PCIe, or
2) coherent PCIe (CAPI), or
3) Intel-Altera HARP (coherent, one package)

[HotCloud'16, ACM SOCC'16]
Dimension #4: Compiler Support

mem system improvement

[ICS'14, TACO'15, ongoing]

source-to-source compiler for coordinated data prefetching:
1.5x speedup on Xeon Phi many-core processor

Future work

Accelerator-Rich Architectures & Systems
Overview of My Research

Application Drivers
- image processing [ISPASS'11]
- deep learning [ICCAD'16]
- genomics [D&T'17]

Compiler Support
- mem system improvement [ICS'14, TACO'15, ongoing]

Accelerator-Rich Systems
- Spark + FPGA
- AaaS: Blaze [HotCloud'16, ACM SOCC'16]
- PARADE [ICCAD'15]
- ARAPrototyper [arXiv'16]
- Transformer [DAC'12, LCTES'12]

Tool: System-Level Automation

Accelerator-Rich Architectures tutorials [ISCA'15 & MICRO'16]

CPU-FPGA: PCIe or QPI? [DAC'16]

Sources of gains [FCCM'16]

CPU-Acc address translation [HPCA'17 best paper nominee]

Near mem acceleration [Memsys'17]
Chip-Level CPU-Accelerator Co-design: Address Translation for Unified Memory Space

[HPCA'17 Best Paper Nominee]

Better programmability and performance
Virtual Memory and Address Translation 101

Virtual memory and its benefits
- Shared memory for multi-process
- Memory isolation for security
- Conceptually more memory

Address translation
- Translation Lookaside Buffer (TLB): cache address translation results
- Memory Management Unit (MMU): virtual-to-physical address translation

Virtual-to-physical address mapping in page granularity
Inefficiency in Today’s ARA Address Translation

#1 Inefficient TLB Support.
TLBs are not specialized to provide low-latency and capture page locality.

#2 High Page Walk Latency.
On an IOTLB miss, 4 main memory accesses are required to walk page tables.

Today’s ARA address translation using IOMMU with IOTLB (e.g. 32-entries)

IOMMU only achieves 12% performance of ideal address translation
**Accelerator Performance Is Highly Sensitive to Address Translation Latency**

Performance Relative to Ideal Address Translation

Translation latency (cycles)

- Must provide efficient address translation support
Characteristic #1: Regular Bulk Transfer of Consecutive Data (Pages)

- Opportunities for relatively simple TLB and page walker designs

TLB miss behavior of BlackScholes benchmark

Access of consecutive pages of one large memory reference

Opportunities for relatively simple TLB and page walker designs
Characteristic #2: Impact of Data Tiling – Breaking a Page into Multiple Accelerators

Page 31

Page 16

Page 15

Page 1

Page 0

Acc 2 Acc 3

Acc 0 Acc 1

Original: 32 * 32 * 32 data array

Rectangular tiling: 16 * 16 *16 tiles

Each tile is mapped to a different accelerator for parallel processing.

But 1 page is split into 4 accelerators!

A shared TLB can be very helpful
Our Two-Level TLB Design

- 32-entry private TLB
- 512-entry shared TLB

Utilization wall limits the number of simultaneously powered accelerators

Still only achieves half the ideal performance => need to improve page walker design
Page Walker Design Alternatives

#1 Improve the IOMMU design to reduce page walk latency
   - Need to design a more complex IOMMU, e.g., GPU MMU with parallel page walker [Power, HPCA'14]

#2 Leverage host core MMU that launches accelerators
   - Very simple and efficient as host core has MMU cache & data cache

Page Table
Base Address

CR3

One data cache line

Prefetch entries to 8 consecutive pages

4-Level Page Walk in 64-bit Virtual Address

MMU cache
Final Proposal: Two-Level TLB + Host Page Walk [HPCA'17 Best Paper Nominee]

On average: 7.6X speedup over naïve IOMMU design, only 6.4% gap between ideal translation

Performance Relative to Ideal Address Translation

- IOMMU
- Private TLB
- Two-level TLB
- Two-level TLB + hostPageWalk

Applications:
- Medical Imaging
- Commercial
- Vision
- Navigation

Deblur, Denoise, Regist., Segment, Black, Stream, Swapt, DispMap, LPCIP, EKFSAM, RobLoc, gmean
Datacentor-Level: Deploying FPGA Accelerators at Cloud Scale
Deploying Accelerators in Datacenters

How to program with your accelerators...?

Big data application developer (e.g., Spark)

How to acquire accelerator resource ...?

Accelerator designer (e.g., FPGA)

How to install my accelerators ...?

Programming challenges:
- Java/Scala vs OpenCL/C/C++
- Explicit accelerator sharing by multiple threads & apps

Performance challenges:
- JVM-to-accelerator communication overhead
- FPGA reconfiguration overhead
Blaze Proposal: *Accelerator-as-a-Service*

[ACM SOCC'16, C-FAR Best Demo Award 3/49]

Big data applications, e.g., Spark programs

Blaze works with Apache Spark and YARN,
Open source link: [https://github.com/UCLA-VAST/blaze](https://github.com/UCLA-VAST/blaze)

**GAM**  
Global Accelerator Manager  
Accelerator-centric scheduling

**NAM**  
Node Accelerator Manager  
Local accelerator service

RM: Resource Manager  
NM: Node Manager  
AM: Application Master
**Blaze Programming Overview**

**Register Accelerators**
- APIs to add accelerator service to corresponding nodes

**Request Accelerators**
- APIs to invoke accelerators through acc_id
- GAM allocates corresponding nodes to applications

**Diagram**
- Big Data Application (e.g., Spark programs)
- ACC Labels
- Containers
- ACC Invoke
- Input data
- Output data
- Global ACC Manager
- Container Info
- ACC Info
- Node ACC Manager
- FPGA
- GPU
- ACCX
#1 Overlapping (pipelining) computation and communication from multiple threads

#2 Data caching on FPGA device memory

#3 Delayed scheduling: same logical tasks are scheduled to same FPGA to avoid reprogram
CDSC FPGA-Accelerated Cluster

A 22-node cluster with FPGA-based accelerators

1 master / driver
1 10GbE switch
20 workers
1 file server

- 1 master / driver
- 20 workers
- 1 file server
- 1 10GbE switch

Spark:
- Computation framework
- In-memory MapReduce system

HDFS:
- Distributed storage framework

Alpha Data board:
1. Virtex 7 FPGA
2. 16GB on-board RAM

Each node:
1. Two Xeon processors
2. One FPGA PCIe card (Alpha Data)
3. 64 GB RAM
4. 10GbE NIC

file server
10GbE switch
Spark
HDFS
## Programming Efforts Reduction with Blaze

### Computational Genomics

<table>
<thead>
<tr>
<th>Applications</th>
<th>LOC Reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Genome Sequence Alignment [HotCloud'16]</td>
<td>896</td>
</tr>
<tr>
<td>Genome Compression</td>
<td>360</td>
</tr>
</tbody>
</table>

Measured in **Lines of Code (LOC)** reduction for accelerator management
Performance of Single-Accelerator with Blaze

With Blaze, a server with an FPGA can replace 1.7~4.3 CPU servers, while providing the same throughput.
Performance of Multi-Accelerator Scheduling w/ Blaze

Half nodes for LR, half nodes for KM

Static partition

Default scheduling policy for CPU

CPU-sharing

Accelerator-centric delayed scheduling

Blaze-GAM

Theoretical optimal

Normalized throughput

Ratio of LR in the mixed Logistic Regression & KMeans workloads

Static or CPU-style sharing cannot handle dynamic workload distributions; Blaze-GAM performs good in most cases.
Great promise of accelerator-rich architectures and systems

- Orders-of magnitude performance and energy gains in customized chips
- Several folds consolidation of the datacenter size with commodity FPGAs

My contributions for chip-level Accelerator-Rich Architectures

- Developed the open-source ARA simulator PARADE
- Analyzed sources of performance gains for customized accelerators
- Proposed an efficient and unified address translation scheme for ARA

My contributions for datacenter-level accelerator deployment

- Proposed accelerator-as-a-service in the cloud
- Contributed the open-source Blaze system

Lots of opportunities to be explored..
When Internet-of-Things (IoT) Marries Accelerator

IoT devices are very sensitive to power/energy consumption

IoT cloud handles big data for real-time analytics

Trillions of dollars market

Customizable chips

Communication costs more energy than computation in IoT, especially after acceleration

Communication-Efficient Accelerator-Rich IoT (CearIoT)
IoT devices: Local low-power accelerator to preprocess data (e.g., filtering, compression)

Cloud: Large-scale data processing with customized datacenters; near mem/storage computing for big data

Regional edge devices: Simple processing & data aggregation (e.g., genome to variants, image to neural bits, request aggregate)

#1 Architecture support
#2 Programming support
#3 Runtime support
#4 Security support
Lots of opportunities…