# Tutorial Outline

<table>
<thead>
<tr>
<th>Time</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>8:30 am – 9:00 am</td>
<td>Introduction</td>
</tr>
<tr>
<td>9:00 am – 10:00 am</td>
<td>Pre-RTL Simulation Framework: Aladdin</td>
</tr>
<tr>
<td>10:00 am – 10:30 am</td>
<td>Break</td>
</tr>
<tr>
<td><strong>10:30 am – 11:00 am</strong></td>
<td><strong>Workload Characterization Tool: WIICA</strong></td>
</tr>
<tr>
<td>11:00 am – 12:00 pm</td>
<td>CAD &amp; Benchmarks: HLS &amp; MachSuite</td>
</tr>
<tr>
<td>12:00 pm – 2:00 pm</td>
<td>Lunch</td>
</tr>
<tr>
<td>2:00 pm – 3:00 pm</td>
<td>Embedded Keynote Talk: Mark Horowitz (Stanford)</td>
</tr>
<tr>
<td>3:00 pm – 3:30 pm</td>
<td>Accelerator Selection Tool: Sigil</td>
</tr>
<tr>
<td>3:30 pm – 4:00 pm</td>
<td>Break</td>
</tr>
<tr>
<td>4:00 pm – 5:00 pm</td>
<td>Hands-on Exercise</td>
</tr>
</tbody>
</table>
WIICA

Workload ISA-Independent Characterization for Applications

Yakun Sophia Shao, Emma Wang, Gu-Yeon Wei, David Brooks
Harvard University
Specialized architectures are decoupled from legacy ISAs.

Spectrum of Specialization:

- General-Purpose CPU
- GPU
- Fixed-Function ASIC

Efficiency:
- Low Efficiency
- High Efficiency

Programmability:
- High Programmability
- Low Programmability
Specialized architectures are decoupled from legacy ISAs.

Spectrum of Specialization:

- **General-Purpose CPU**
- **GPU**
- **Fixed-Function ASIC**

- Low Efficiency ↔ High Efficiency
- High Programmability ↔ Low Programmability
- Tied to a Specific ISA ↔ No ISA
Performance-Counter Based Workload Characterization

• Metrics
  – IPC
  – Cache miss rates
  – Branch mis-prediction rates
  – …

• Microarchitecture-dependent
  – What if there is a bigger cache/a better branch predictor?
  – Not program intrinsic characteristics
ISA impacts program behaviors.

Stack Overhead
- Limited Registers
- Additional Load/Store
ISA impacts program behaviors.

Stack Overhead
- Limited Registers
- Additional Load/Store

Complex Operations
- Memory Operands
- Vector Operations
ISA impacts program behaviors.

Stack Overhead
• Limited Registers
• Additional Load/Store

Complex Operations
• Memory Operands
• Vector Operations

Calling Conventions
WIICA Summary

Goal:

• An analysis tool to characterize workloads ISA-Independent characteristics for specialized architectures

Methods:

• Leverage compiler’s intermediate representation (IR)
• Categorize characteristics into compute, memory, and control

Takeaways:

• ISA-dependent characterization is misleading for specialization.
• ISA-independent characterization allows designers to quickly identify opportunities for specialization.
WIICA Summary

Goal:
- An analysis tool to characterize workloads ISA-Independent characteristics for specialized architectures

Methods:
- Leverage compiler’s intermediate representation (IR)
- Categorize characteristics into compute, memory, and control

Takeaways:
- ISA-dependent characterization is misleading for specialization.
- ISA-independent characterization allows designers to quickly identify opportunities for specialization.
WIICA Summary

Goal:
• An analysis tool to characterize workloads ISA-Independent characteristics for specialized architectures

Methods:
• Leverage compiler’s intermediate representation (IR)
• Categorize characteristics into compute, memory, and control

Takeaways:
• ISA-dependent characterization is misleading for specialization.
• ISA-independent characterization allows designers to quickly identify opportunities for specialization.
Tool Overview

Program ➔ Design of Specialized Architecture
Tool Overview

Program

IR Trace

ISA-Independent

Design of Specialized Architecture

Characterization for Specialized Architecture

Compute
Memory
Control
Tool Overview

Program

IR Trace

x86 Trace

Design of Specialized Architecture

ISA-Independent

Characterization for Specialized Architecture

Compute

Memory

Control

ISA-Dependent
Program Representations

- Program
  - LLVM
    - IR Trace
    - LLVM
      - x86 Trace
Program Representations

LLVM IR

• High-level IR
• Machine- and ISA-independent
• Features:
  – 60 instructions
  – Unlimited registers
  – Only loads/stores access memory
  – Disable vector operations
Program Representations

**x86 Trace**
- Used for ISA-dependent analysis
- Semantically equivalent to the IR code
- Collected with Pin instrumentation
Tool Overview

Program

IR Trace

x86 Trace

Design of Specialized Architecture

Characterization for Specialized Architecture

ISA-Independent

ISA-Dependent

Compute
Memory
Control
ISA-Independent Workload Characteristics

- **Compute**
  - Opcode Diversity
  - Static Instructions (I-MEM)

- **Memory**
  - Memory Footprint (D-MEM)
  - Memory Entropy
  - Locality Score

- **Control**
  - Branch Instruction Counts
  - Branch Entropy
ISA-Independent Workload Characteristics

- **Compute**
  - Opcode Diversity
  - **Static Instructions (I-MEM)**

- **Memory**
  - Memory Footprint (D-MEM)
  - Memory Entropy
  - Locality Score

- **Control**
  - Branch Instruction Counts
  - Branch Entropy
Compute::Static Instructions

![Bar chart showing the number of unique static instructions for various benchmarks. The x-axis represents the benchmarks, and the y-axis represents the number of instructions. The chart compares x86 and IR versions of the benchmarks.]
ISA-Independent Workload Characteristics

**Compute**
- Opcode Diversity
- Static Instructions (I-MEM)

**Memory**
- Memory Footprint (D-MEM)
- **Memory Entropy**
- Locality Score

**Control**
- Branch Instruction Counts
- Branch Entropy
Memory::Entropy

Entropy: a measure of the randomness

\[
\text{Entropy} = - \sum_{i=1}^{N} p(x_i) \log_2 p(x_i)
\]
Memory::Entropy

Entropy: a measure of the randomness

\[ Entropy = - \sum_{i=1}^{N} p(x_i) \cdot \log_2 p(x_i) \]

Case 1:
X is always a constant.

\[ p(X) = 1 \]
\[ \log_2 p(X) = 0 \]
\[ Entropy = 0 \]

Case 2:
N possible outcomes of X occur equally.

\[ p(X) = \frac{1}{N} \]
\[ \log_2 p(X) = \log_2 N^{-1} \]
\[ Entropy = -N \cdot \frac{1}{N} \cdot \log_2 N^{-1} \]
\[ Entropy = \log_2 N \]
Memory::Global Address Entropy

Temporal Locality

Address Stream A
(less temporal locality)

0 0 0 0
0 0 0 1
0 0 1 0
0 0 1 1

Entropy = 2

Address Stream B
(more temporal locality)

0 0 1 1
0 0 1 1
0 0 1 1
0 0 1 1

Entropy = 0

Yen, Draper, and Hill. Notary: Hardware Techniques to Enhance Signatures. MICRO 08
Memory::Global Address Entropy

Temporal Locality

Address Stream A
(less temporal locality)

0 0 0 0
0 0 0 1
0 0 1 0
0 0 1 1

Address Stream B
(more temporal locality)

0 0 1 1
0 0 1 1
0 0 1 1
0 0 1 1

Entropy = 2

Entropy = 0

Yen, Draper, and Hill. Notary: Hardware Techniques to Enhance Signatures. MICRO 08
Memory::Global Address Entropy

Temporal Locality

Address Stream A (less temporal locality)

0 0 0 0
0 0 0 1
0 0 1 0
0 0 1 1

Entropy = 2

Address Stream B (more temporal locality)

0 0 1 1
0 0 1 1
0 0 1 1
0 0 1 1

Entropy = 0

Yen, Draper, and Hill. Notary: Hardware Techniques to Enhance Signatures. MICRO 08
ISA-Independent Workload Characteristics

- **Compute**
  - Opcode Diversity
  - Static Instructions (I-MEM)

- **Memory**
  - Memory Footprint (D-MEM)
  - Memory Entropy
  - Locality Score

- **Control**
  - Branch Instruction Counts
  - **Branch Entropy**

*Yokota, et al., Introducing Entropies for Representing Program Behavior and Branch Predictor Performance, 07*
Control::Branch Entropy
ISA-Independent Workload Characteristics

- **Compute**
  - Opcode Diversity
  - Static Instructions (I-MEM)

- **Memory**
  - Memory Footprint (D-MEM)
  - Global Address Entropy
  - Local Address Entropy

- **Control**
  - Branch Instruction Counts
  - Branch Entropy
Workload Characterization
Tool Overview

Program

IR Trace

x86 Trace

ISA-Independent

ISA-Dependent

Design of Specialized Architecture

Characterization for Specialized Architecture

Compute
Memory
Control
Tutorial References


- [http://vlsiarch.eecs.harvard.edu/accelerators](http://vlsiarch.eecs.harvard.edu/accelerators)