Labeled RISC-V: A New Perspective on Software-Defined Architecture

Zihao Yu, Bowen Huang, Jiuyue Ma, Ninghui Sun, Yungang Bao

Oct 14th, 2017 @ Boston

Institute of Computing Technology (ICT), Chinese Academy of Sciences (CAS)
Control in Computer Architecture

• Maurice Wilkes proposed microprogramming to design the control unit of a processor in 1951.

• RISC became popular in 1980s.
  – More efforts to datapath
  – Less attentions to control

• But as multicore emerges, weak control leads to a new problem.
Weak Control -> Unmanaged Sharing

- Unmanaged sharing -> resource contentions
- -> performance interference/degradation
- -> bad user experience

Which one is more critical?
Unmanaged Sharing in Datacenter

- SMT, LLC, DRAM, Network

|                | 5%  | 10% | 15% | 20% | 25% | 30% | 35% | 40% | 45% | 50% | 55% | 60% | 65% | 70% | 75% | 80% | 85% | 90% | 95% |
|----------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| **websearch**  | 134%| 103%| 96% | 96% | 109%| 102%| 100%| 96% | 96% | 104%| 99% | 100%| 101%| 100%| 104%| 103%| 103%| 103%|
| LLC (small)    | 134%| 103%| 96% | 96% | 109%| 102%| 100%| 96% | 96% | 104%| 99% | 100%| 101%| 100%| 104%| 103%| 103%| 103%|
| LLC (med)      | 134%| 103%| 96% | 96% | 109%| 102%| 100%| 96% | 96% | 104%| 99% | 100%| 101%| 100%| 104%| 103%| 103%| 103%|
| LLC (big)      | >100%| >100%| >100%| >100%| >100%| >100%| >100%| >100%| >100%| >100%| >100%| >100%| >100%| >100%| >100%| >100%| >100%| >100%|
| DRAM           | >100%| >100%| >100%| >100%| >100%| >100%| >100%| >100%| >100%| >100%| >100%| >100%| >100%| >100%| >100%| >100%| >100%| >100%|
| HyperThread    | >100%| >100%| >100%| >100%| >100%| >100%| >100%| >100%| >100%| >100%| >100%| >100%| >100%| >100%| >100%| >100%| >100%| >100%|
| CPU power      | 190%| 124%| 110%| 107%| 134%| 115%| 115%| 106%| 108%| 102%| 114%| 107%| 105%| 104%| 101%| 105%| 100%| 98% |
| Network        | 35% | 35% | 36% | 36% | 36% | 36% | 37% | 37% | 38% | 39% | 41% | 44% | 48% | 51% | 55% | 58% | 64% | 95% |

Impact on real time

• Hard to satisfy with multicore
  – Disable multicore

Read 4KB data from memory

Single App

Multiple Apps

Labeled von Neumann Architecture (LvNA)

1. Fine-grained object
2. Semantic association
3. Propagation
4. Software-defined control logic

Programmable Architecture for Resourcing-on-Demand

① Add label Reg
② Allocate label to each VM
③ Attach label to each Req
④ Add label-based, programmable control logic
⑤ Abstract label Regs and CLs into files

PARD

Platform Resource Manager (PRM)

Linux-based firmware

/sys/cpa
- cpa0
  - ident
  - type
  - ldoms
  - ldom0
  - ldom1
  - ldom2
  - param1
  - param2
  - statistics
  - trigger
  - cpa1
  - cpa2

/sys/cpa0
- parameter
- ident
- type
- ldoms
- ldom0
- ldom1
- ldom2
- param1
- param2
- statistics
- trigger
- cpa1
- cpa2

Shared Last Level Cache

I/O Chipset

Memory Controller

Disk

NIC

Core

VM0

VM1

VMn

CL

Datapath

Data Plane

Programming Interface

Original MemReq

DS-id

Prl1

Prl2

Prl3

[0,4GB]

4GB,6GB

...
Implementation

- Full-system cycle-accurate simulator **Open Sourced**
- FPGA prototype on Xilinx VC709 evaluation board
- MicroBlaze version **Deprecated**
- RISC-V version **Open Sourced**

* http://github.com/fsg-ict/PARD-gem5
+ http://github.com/fsg-ict/labeled-RISC-V
LvNA + RISC-V = Labeled RISC-V

• Features to add
  • Labels registers after tiles
  • Cache CL
  • Label Converter in TLtoAXI
  • Memory CL
  • PRM

<table>
<thead>
<tr>
<th>DS-id</th>
<th>Base</th>
<th>Len</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0x0000</td>
<td>0x4000</td>
</tr>
<tr>
<td>2</td>
<td>0x8000</td>
<td>0x8000</td>
</tr>
</tbody>
</table>
Overheads

• 16 lines of chisel code to add labels into RocketChip
  – Add dsid member in the Bundle of TileLink2
  – Attach labels at the TileLink2 masters of core tiles
• < 5% resource overheads for CLs
  – Much less with complex cores, e.g. BOOM
• No performance overheads for critical apps according to the timing report
Demo 1 - NoHype

PARD Server

Isolate resources (address space, device) by CLs

Push the software hypervisor down to LvNA

Traditional Server
Demo 2 - Memory Bandwidth Control

• Use labeled token buckets to protect the bandwidth from attacker

<table>
<thead>
<tr>
<th>Function</th>
<th>Best Rate MB/s</th>
<th>Avg time</th>
<th>Min time</th>
<th>Max time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copy:</td>
<td>53.8</td>
<td>0.074376</td>
<td>0.074305</td>
<td>0.074430</td>
</tr>
<tr>
<td>Scale:</td>
<td>46.7</td>
<td>0.086195</td>
<td>0.085691</td>
<td>0.087494</td>
</tr>
<tr>
<td>Add:</td>
<td>50.1</td>
<td>0.119921</td>
<td>0.119842</td>
<td>0.120034</td>
</tr>
<tr>
<td>Triad:</td>
<td>48.2</td>
<td>0.124578</td>
<td>0.124473</td>
<td>0.124679</td>
</tr>
</tbody>
</table>

Solo

<table>
<thead>
<tr>
<th>Function</th>
<th>Best Rate MB/s</th>
<th>Avg time</th>
<th>Min time</th>
<th>Max time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copy:</td>
<td>11.7</td>
<td>0.344165</td>
<td>0.340619</td>
<td>0.347746</td>
</tr>
<tr>
<td>Scale:</td>
<td>12.1</td>
<td>0.341771</td>
<td>0.331738</td>
<td>0.356621</td>
</tr>
<tr>
<td>Add:</td>
<td>8.1</td>
<td>0.748021</td>
<td>0.737555</td>
<td>0.755204</td>
</tr>
<tr>
<td>Triad:</td>
<td>8.1</td>
<td>0.751058</td>
<td>0.740938</td>
<td>0.760358</td>
</tr>
</tbody>
</table>

interfered

<table>
<thead>
<tr>
<th>Function</th>
<th>Best Rate MB/s</th>
<th>Avg time</th>
<th>Min time</th>
<th>Max time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copy:</td>
<td>53.7</td>
<td>0.074583</td>
<td>0.074555</td>
<td>0.074617</td>
</tr>
<tr>
<td>Scale:</td>
<td>46.6</td>
<td>0.085922</td>
<td>0.085853</td>
<td>0.086026</td>
</tr>
<tr>
<td>Add:</td>
<td>50.0</td>
<td>0.120866</td>
<td>0.119926</td>
<td>0.122406</td>
</tr>
<tr>
<td>Triad:</td>
<td>48.1</td>
<td>0.125066</td>
<td>0.124837</td>
<td>0.125717</td>
</tr>
</tbody>
</table>

isolated
A lot to explore!

- **Theory**: How does LvNA impact on RAM, PRAM, LogP models?
- **Hardware/Arch**: How to implement LvNA at CPU pipeline/SMT, memory, storage, networking? How to correlate LvNA and SDN by labels?
- **OS/Hypervisor**: How to correlate labels with VMs, containers, processes, threads? How to abstract programming interfaces for labels?
- **Programming Model and Compilers**: How to express users’ requirements and propagate to the hardware via labels? How to make compilers support labels?
- **Distributed systems**: How to correlate labels with distributed resources? How to manage distributed systems with label mechanisms?
- **Measurement/Audit**: How to leverage labels to gauge and audit resource usages?

- Finished
- On-going
- Have ideas
- Feature work
Summary

• **LvNA**: a model of software-defined architecture
• **PARD**: a proof of concept of LvNA
• **Labeled RISC-V**: an implementation of LvNA
Thanks
Q & A

yuzihao@ict.ac.cn

Labeled RISC-V: A New Perspective on Software-Defined Architecture