Litmus Testing at Rack Scale
We're Going to Build a Large Program Collider

Collide instructions at 0.99c, and observe the decay products.
Programmers Once (Thought They) Understood Computer Architecture

Figure 1.4
Hardware organization of a typical system. CPU: Central Processing Unit, ALU: Arithmetic/Logic Unit, PC: Program counter, USB: Universal Serial Bus.

systems, but all systems have a similar look and feel. Don’t worry about the complexity of this figure just now. We will get to its various details in stages throughout the course of the book.
Symmetric Multiprocessors Were *Fairly* Simple
Concurrent Code Makes Architecture Visible

- Consider message passing.
  - Pretty much the simplest thing you can do with shared memory.
  - Systems like Barrelfish rely on it.
- When are barriers required?
- You can't write good code, without sufficiently understanding the hardware.
- We're combining components in new ways.
Message Passing with Shared Memory

CPU

Write: *x = 42
Write: *y = 1

RAM

*x = 02  *y = 0

CPU

Read: *y = 1
Read: *x = 42
Message Passing with a Write Buffer

Write: \( *x = 42 \)
Write: \( *y = 1 \)
\( *x = 42 \)
\( *y = 1 \)

Read: \( *y = 1 \)
Read: \( *x = 0 \)
\( *x = 0 \)
\( *y = 0 \)
Message Passing with a Barrier

CPU

Write: \( *x = 42 \)
Write: \( *y = 1 \)

WB

\( *x = 42 \)
\( *y = 1 \)

RAM

\( *x = \emptyset \)
\( *y = \emptyset \)

CPU

Read: \( *y = 1 \)
Read: \( *x = 42 \)
Of Course, CPUs Aren't *That* Simple

9 hops
seL4 was verified *modulo* a hardware model.

The Cortex A8 has bugs:
- Cache flushes don't work.
- As of today, these “errata” are still not public.
- We rediscovered these by accident.
- Non-coherent memory is coming.

Source: Chip Errata for the i.MX51, Freescale Semiconductor
And Then There's Rack Scale...
There's a Lot of Data Available

Program trace

Cache dumps

Port mirroring

Event triggers

Openflow

Backhaul
ARM High-Speed Serial Trace Port

- Streams from the *Embedded Trace Macrocell*.
- Cycle-accurate control flow + events @ 6GiB/s+
- Compatible with FPGA PHYs.
- Well-documented protocol.
- Available on ARMv8
The HSSTP Hardware

- The official tool is CHF10,000 per core.
- The cable run is maximum 15cm.
- It's PHY-compatible with common FPGAs
- A CHF6k FPGA could easily handle 10 – 15x cheaper!
- We're working with the D-ITET DZ on an interface board.
- *If you like soldering, let us know!*

---

*Image of HSSTP Hardware components*
Fancy Triggering and Filtering

- The ETM has sophisticated filtering e.g. Sequencer.
- $Bn$ and $Fn$ can be just about any events on the SoC.
- States can enable/disable trace, or log events.
- A powerful facility for pre-filtering
Filtering and Offload in an FPGA

- We'll need to intelligently filter high-rate data.
- We're using an FPGA for the physical interface already.
- How much processing could we do?
- We have expertise in the group with FPGA query offloading
  - Zsolt and I are writing a joint Master's project proposal on this.
Hardware Tracing for Correctness

Are HW operations right?

- $\exists va. va \rightarrow pa$
- $\forall va. va \rightarrow pa$

- Real time pipeline trace on ARM.
- Can halt and inspect caches.
- HW has “errata” (bugs).
- Check that it actually works!
- Catch transient and race bugs.

- Log & process offline
- Filter at line rate
- Check temporal assertions

5Gb/s
Hardware Tracing for Performance

- Should see N coherency messages.
- Do we?
  - The HW knows!

Is URPC optimal?

Log & process offline

Filter at line rate

5Gb/s

Core 0

INEVAL(0)

READ(1)

...
Properties to Check: Security

- Runtime verification is an established field.
- Lots of existing work to build on.
- What properties could we check efficiently?
- How could we map them to the filtering pipeline?

/* A very simple TESLA assertion. */
TESLA_WITHIN(example_syscall, 
  previously(security_check(ANY(ptr), 
    o, op) == 0));

http://www.cl.cam.ac.uk/research/security/ctsrd/tesla/
That's a lot of data, how can we process it?

This is what rack-scale systems are for!

Andrei is starting on this as his Master's project.
Properties to Check: Memory Management

- Could we check this?
- We don't have data values \((a \& b)\).
- We can play clever tricks with the hardware!
- Shows what we could do with data trace.

```c
void *a = malloc();
...
free(b);
{a = b}
```

In a VMSA implementation, the CONTEXTIDR bit assignments are:

<table>
<thead>
<tr>
<th>Bit Position</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>PROCID</td>
</tr>
<tr>
<td>8-7</td>
<td>ASID</td>
</tr>
<tr>
<td>0</td>
<td>PROCID</td>
</tr>
</tbody>
</table>

\[ \text{PROCID} = b[15:0]; \]
\[ \text{PROCID} = b[31:16]; \]

\[ \text{CID: } B[15:0] \text{ ++ ASID} \]
\[ \text{CID: } B[31:16] \text{ ++ ASID} \]
A Streaming Verification Engine

- **Sources**
  - HSSTP Packet Capture
  - ETM Sequencer
  - FPGA Capture

- **Capture**
  - ETM Sequencer
  - FPGA Capture

- **Processing**
  - Dataflow Engine
  - FPGA Offload

- **Properties**
  - TESLA
    - malloc()
    - pairing
  - Coherence correctness

- **Constraints**
  - Requirements
Offloading Example: LTL to Büchi

- LTL(-ish) formula: A store on core 1 is eventually visible on core 2.
- Think regular expressions for infinite streams.
- As for REs, we compile a checking automaton.
- Run the automaton in real time and look for violations.
- FPGAs are good at state machines.
An Instrumented Rack-Scale System

- 64 SoCs x 5Gb/s = 320Gb/s trace output.
- Online checkers (e.g. automata) will be essential at this scale.
- We're going to build this:
  - A rack of ARMv8 cores & FPGAs.
- We're starting a fortnightly reading group to get up to speed on the Runtime Monitoring literature – feel free to join.

https://code.systems.ethz.ch/project/view/55/

rack-tracing@lists.inf.ethz.ch