Going Further in Hardware Tracing

Suchakrapani Datt Sharma

May 5, 2016

École Polytechnique de Montréal

Laboratoire **DORSAL**

Agenda

Introduction

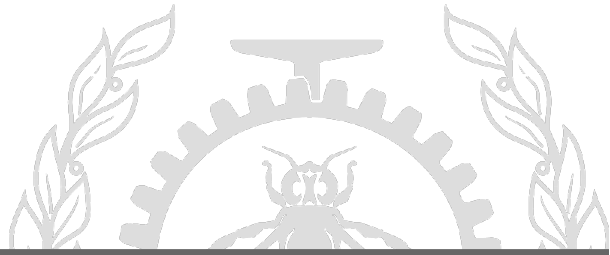
- Hardware Tracing
- Research Updates

New Investigations

- Intel PT
  - Tracing the Tracers
  - VM Analysis!
- Experiments with CoreSight
- ARM CoreSight Internals

Upcoming and in-progress

- Coresight applications and benchmarks

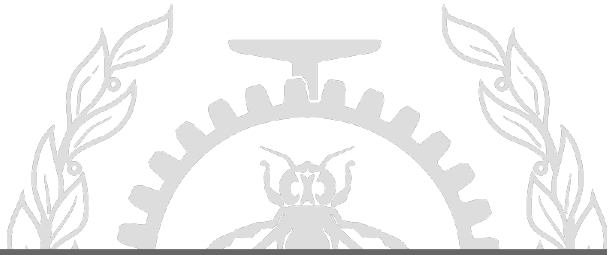


Introduction

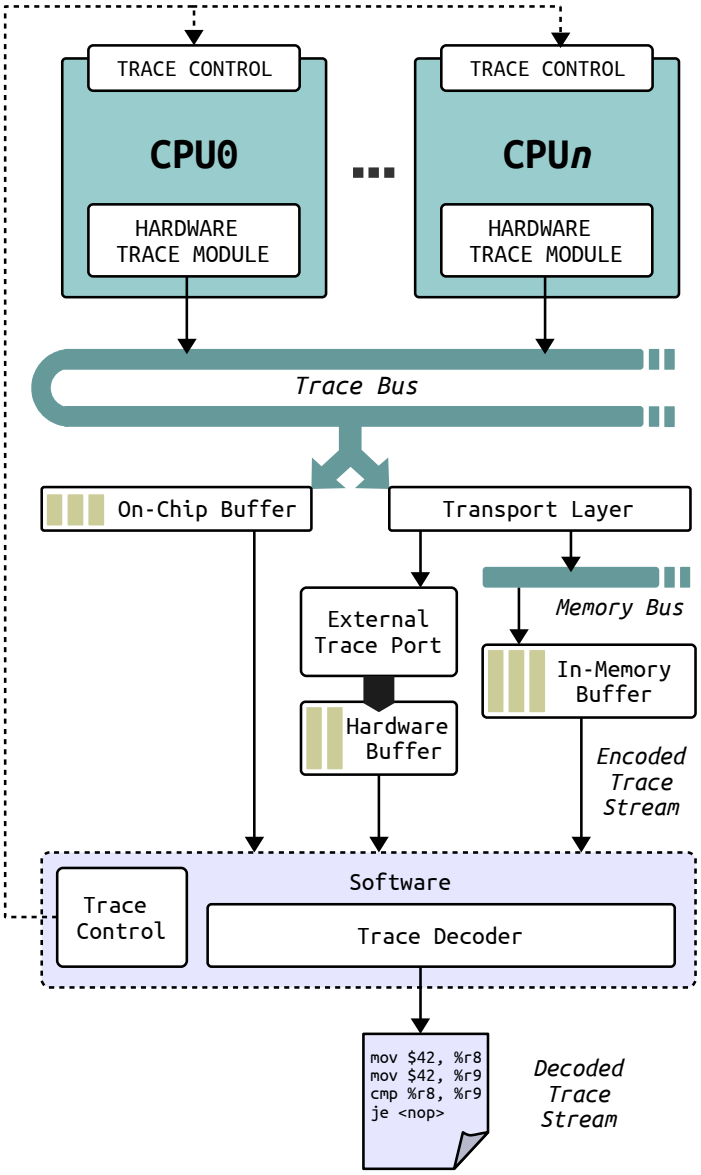
Research Focus : Hardware tracing on Intel **and ARM** for low overhead and high accuracy tracing and profiling

Research Updates

- Intel PT internals and overhead
- Analysis of hardware trace packet decoding
- VM Analysis through PT traces
- Journal paper submitted to JoE (IET)



Hardware Tracing 101



What and Why?

- In its current form, I define it as “traceless tracing” (zero-overhead)
- Precise real-time data for instruction level profile and debug
- Trace packets flow from 'processor' to on-chip buffer or external transport
- Traces in range of hundreds of Mbits/s to Gbits/s.
- Being quickly adopted in Linux (Perf/Coresight)

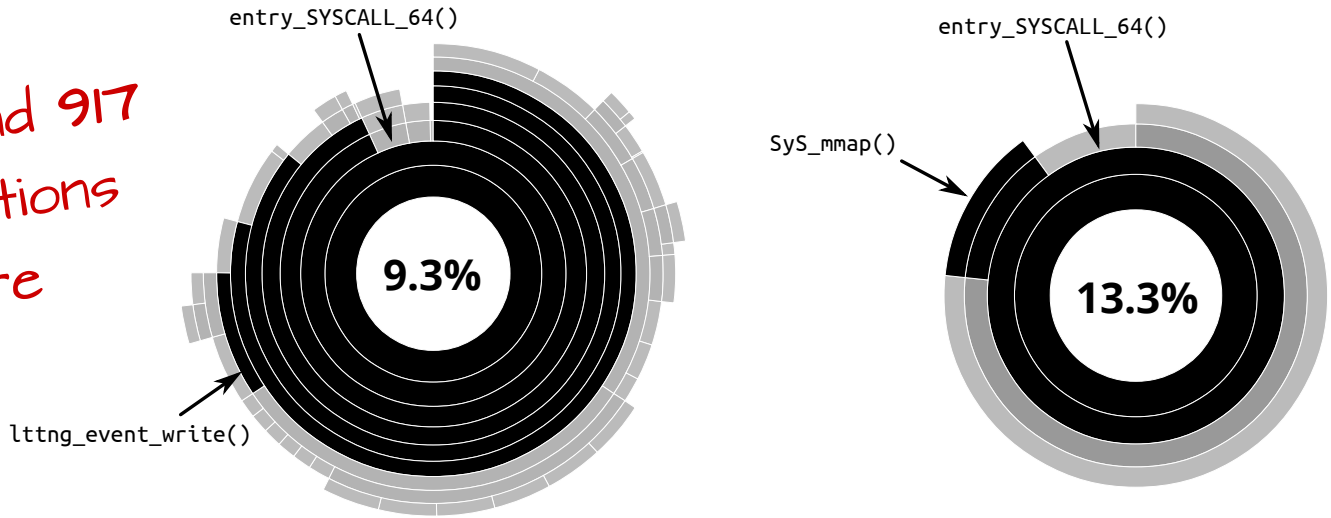
Cite image as, Sharma S., Dagenais M, *Hardware-Assisted Instruction Profiling and Latency Detection,* (pre-print) DORSAL, 2016

Intel PT

Tracing the Tracers

- Targeted snapshot of callstacks - `mmap()` example

173 ns and 917 instructions more



PT Overheads

- 2-3% in common use-cases. Mostly memory related.
- Ftrace vs PT : 63% vs 1% (I/O)

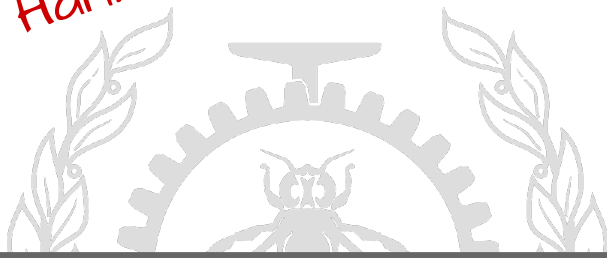
Cite image and data as, Sharma S., Dagenais M, *Hardware-Assisted Instruction Profiling and Latency Detection,* (pre-print) DORSAL, 2016

Intel PT

VM Analysis

- Yes it is technically possible!
- **VMPT :** https://github.com/tuxology/vmpt
- Decoder + TraceCompass View
- VMCS packets are generated from PT hardware
  - VMCS Base Register (Associated VM)
- Decode [PIP -- PAD (8) -- VMCS -- TSC]
- Bundle decoded associated PIP (CR3 value / NR bit), VMCS base register and TSC packets in XML/JSON
- Analysis requires small kernel patch for now

Talk to Hani



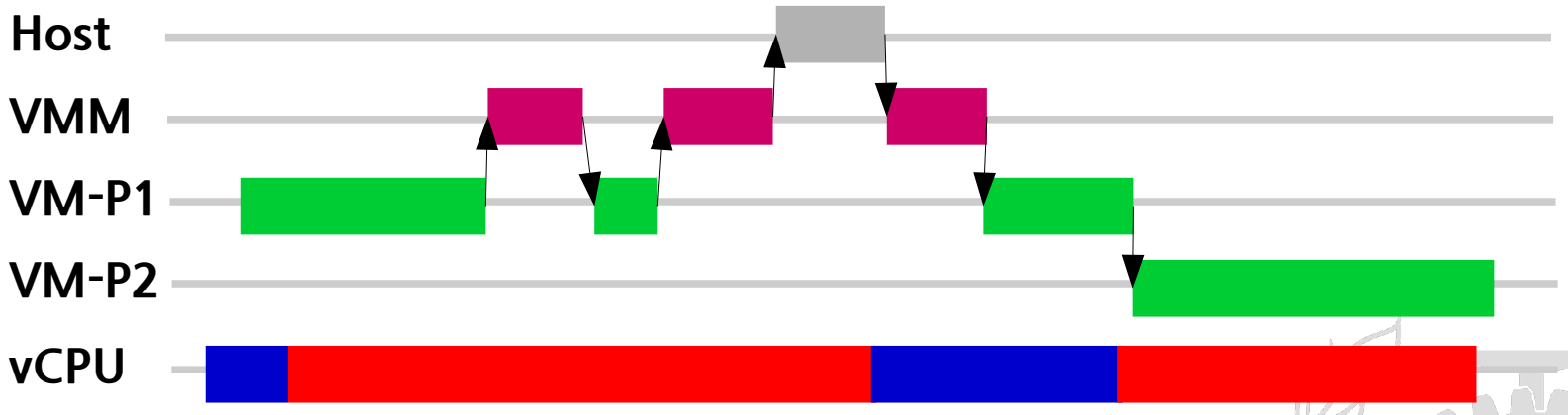
Intel PT

VM Analysis

PT Binary Dump → VMPT →

```
<bundle>
    <PIP> 32423545 </PIP>
    <NR> 1 </NR>
    <VMCS> 243241334 </VMCS>
    <TSC> 2342353646 </TSC>
</bundle>
```

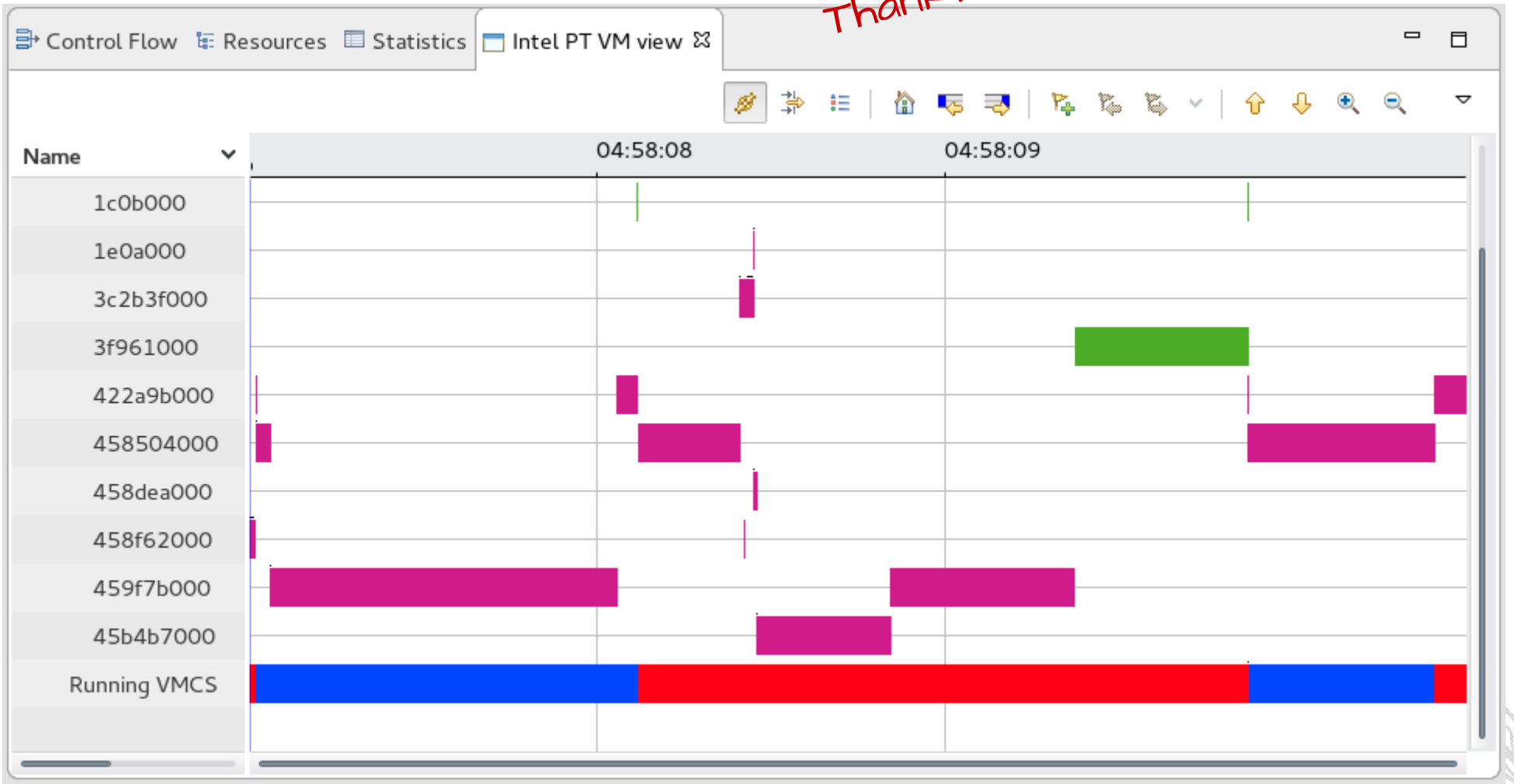
Expectation :



Intel PT

VM Analysis

Thanks to Geneviève!



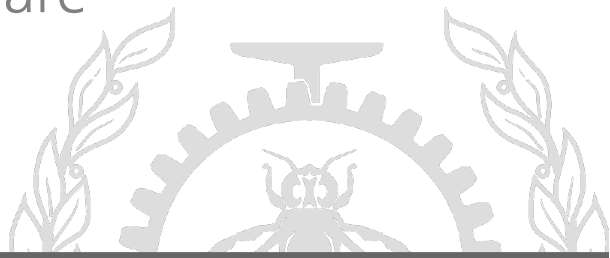
Hardware Tracing

Other Architectures

- ARM CoreSight (Program and Data Flow Trace) [1]
  - Stream trace to external transport or internal buffer
- Intel PT (Program Flow) [2] [3]
- MIPS PDTrace (Program and Data Flow)

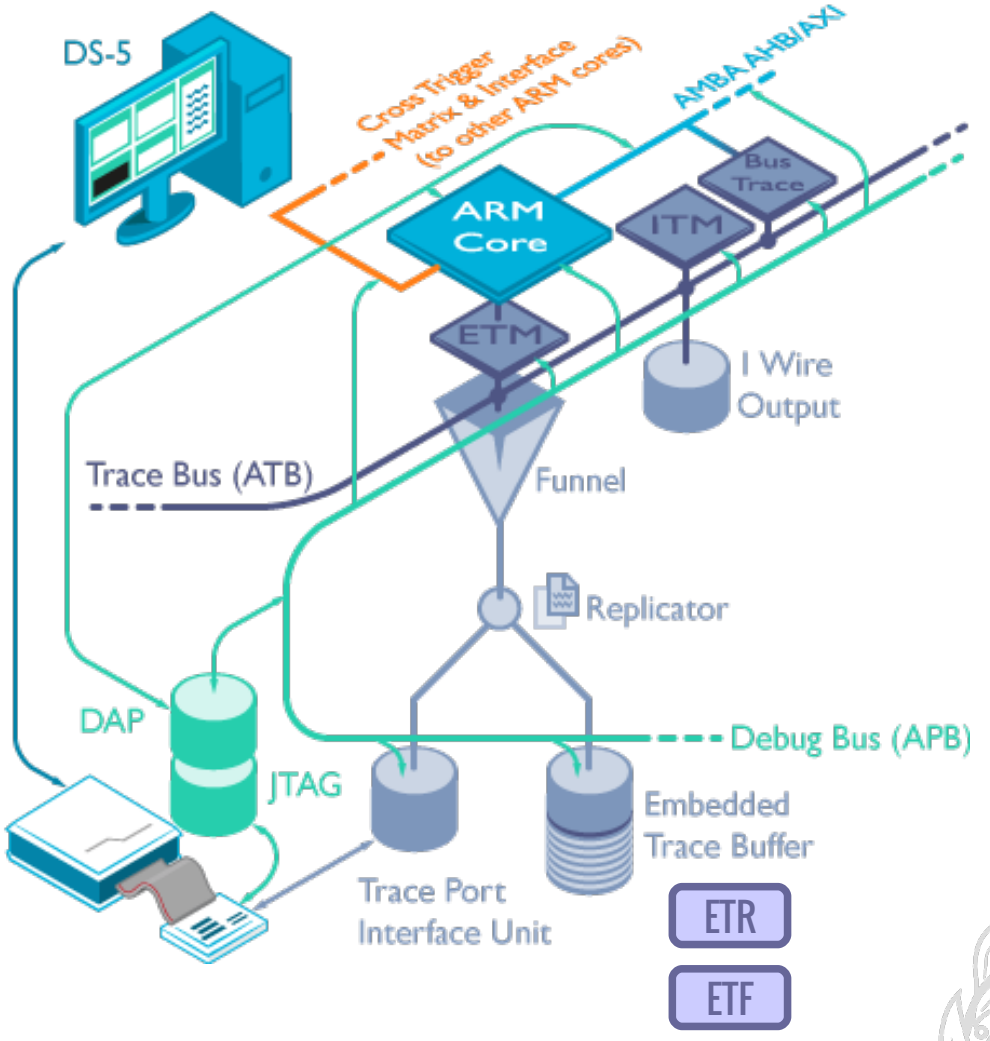
ARM CoreSight

- Program Flow Trace and Data Trace
  - PE → Trace Router → System Bus → System RAM
  - PE → Trace FIFO → TPIU → External Hardware
- Can be configured as desired on silicon



ARM CoreSight

SoC View

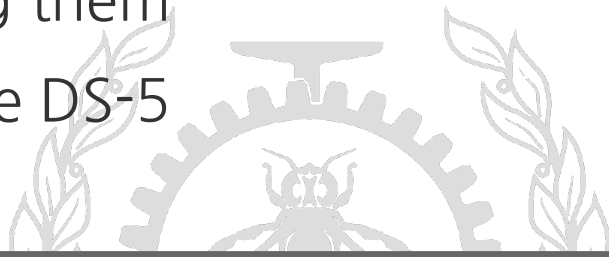


Courtesy, ARM

ARM CoreSight

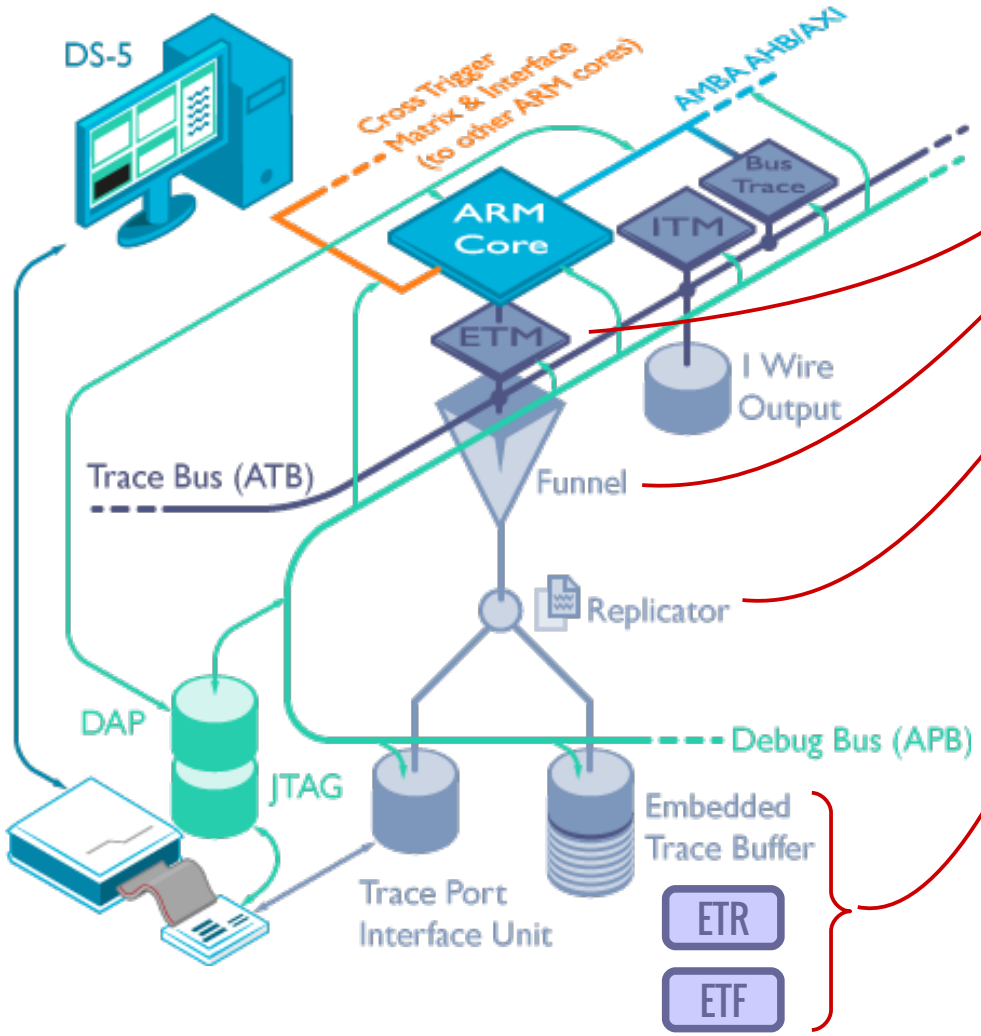
ETMv4

- Major revision [4], highly configurable
  - Insn only for A family. Data+Insn only for M and R family
    - P0 (Insn), P1 and P2 (Data) elements, Other elements
    - **P0 :** Atom elements (E/N), Q elements (cycle count)
    - **P0 :** Branch, Synchronization, Exceptions, TimeStamp, Conditional (C), Result (R), Mispredict etc.
- Analyzer decodes same way as libipt (Intel PT decoder lib)
  - **Trace Control** with CSAL
  - Expose configuration registers by mmaping them
  - Trace start and stop. **Decoding** needs to use DS-5



ARM CoreSight

SoC View



Kernel View

Device Tree

CoreSight Driver

Configure Sink/Source

Trace Start/Stop bit

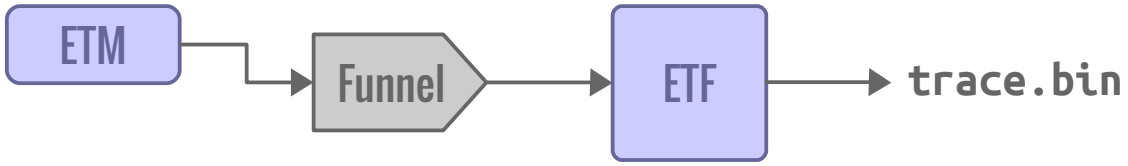
Encoded Trace

Courtesy, ARM

ARM CoreSight

Experiments with Cortex-A53 (ARMv8)

- Qualcomm Snapdragon 410 platform
- Configure ETM as source and ETF as sink with CS driver



- Linaro's kernel, upstream in 4.7 probably
- Decoder issues! `ptm2human` [5] decodes ETMv4 packets but is not as mature as DS-5 or Intel's processor trace library
  - Improve `ptm2human`
  - Own decoder as per research requirements

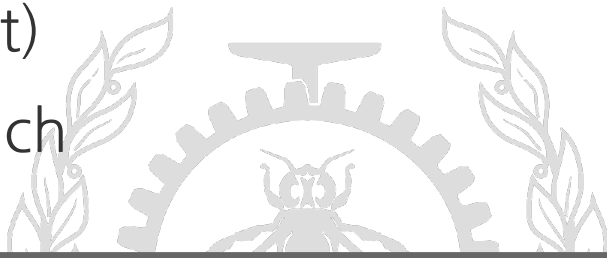
Upcoming

ARM CoreSight

- Try to use ETR to send data to a RAM buffer
- Either improve `ptm2human` or use CASL for tracing
- Benchmarks to compare CoreSight for ARMv8 and Intel PT for Skylake machines
  - Trace granularity
  - Trace size, bandwidth and overheads

Intel PT

- VM Analysis
  - Bigger PT Buffer (Perf or modified simple-pt)
  - Compare with Hani's software-only approach



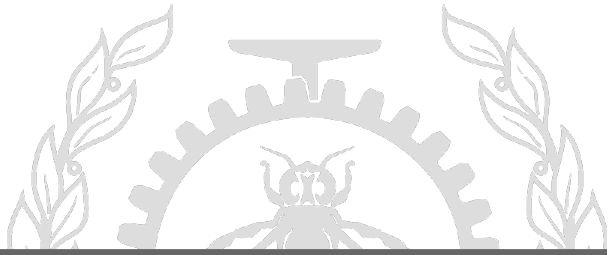
Upcoming

With Hardware Traces

- High level device drivers analysis based on execution profiles
- Rudimentary memory analysis with instruction flow only
  - Such static analysis techniques have been tried [6]. Can we do it with decoded hardware traces?

Misc

- Cycle accuracy of architecture simulators such as PTLSim [7] or Marssx86 [8] vs PT
- eBPF controlled/filtered snapshots - integrate hardware assisted tracing with dynamic tracers



References

[1] Debug and Trace for Multicore SoCs, ARM Whitepaper, 2008

[2] Intel 64 and IA-32 Architecture Software Developer Manual

[3] Adding processor Trace Support to Linux [LWN] http://lwn.net/Articles/648154/

[4] ARM® Embedded Trace Macrocell Architecture Specification (ETMv4.0 to ETMv4.2)

[5] *ptm2human* : https://github.com/hwangcc23/ptm2human

[6] Venkitaraman R, Gupta G, *Static Program Analysis of Embedded Executable Assembly Code*, ACM CASES 2004

[7] Yourst Matt, *PTLsim: A Cycle Accurate Full System x86-64 Microarchitectural Simulator*, 2007 IEEE International Symposium on Performance Analysis of Systems & Software

[8] *MARSSx86*, http://www.marss86.org

Questions?

suchakrapani.sharma@polymtl.ca

suchakra on #lttng