Low Level Tracing for Latency Analysis

From Baremetal to Hardware Tracing Blocks

Suchakrapani Datt Sharma & Thomas Bertauld

Oct 12, 2016

École Polytechnique de Montréal
Laboratoire DORSAL
whoami

**Suchakra**

- PhD student, Computer Engineering  
  (Prof Michel Dagenais)  
  DORSAL Lab, École Polytechnique de Montréal - UdeM

- Works on debugging, tracing and trace aggregation (LTTng, eBPF), hardware tracing and VMs.

- Loves poutine, samosas and bikes
Thomas

• MSc student, Computer Engineering
  (Prof Michel Dagenais)
  DORSAL Lab, École Polytechnique de Montréal - UdeM

• Worked on embedded systems tracing, baremetal systems, trace analysis and now in financial-tech domain

• Loves computer games
Agenda

**Latency**
- Introduction
- Tools and techniques

**Hardware Tracing**
- Intel Processor Trace
- ARM CoreSight
- Hardware trace based analysis

**Baremetal Tracing**
- Heterogeneous system challenges
- Low level traces with barectf
Latency
Super Complex Stuff

Latency
Hardware interrupt
Delay between interrupt and do_IRQ()
Delay between interrupt and do_IRQ()

Interrupt Handler Delay

Hardware interrupt

Interrupt Handler

Request task scheduling
Hardware interrupt → Delay between interrupt and do_IRQ() → Interrupt Handler → Request task scheduling → Task Scheduled → Scheduling Delay → Interrupt Handler Delay → Delay between interrupt and do_IRQ() → Hardware interrupt
Delay between interrupt and do_IRQ()

Interrupt Handler

Delay

Scheduling

Delay

Execution

Delay

Hardware interrupt

Interrupt Handler

Request task scheduling

Task Scheduled

Task Exits
Challenges in Measurement

• May be non-deterministic, sporadic and may required long term continuous monitoring
• Measuring techniques may add latency to critical path
• Modern SoCs - multiple processors, multiple architectures
• Sampling techniques have limited viability
Ftrace
Interesting Events

do_IRQ()
events/irq/irq_handler_entry/enable
events/sched/sched_waking/enable
events/sched/sched_wakeup/enable
events/sched/sched_switch/enable
sys_exit

do_IRQ
irq_handler_entry
sched_waking
sched_wakeup
sched_switch
sys_exit
Ftrace

irqsoff/preemptirqsoff Tracer

# irqsoff latency trace v1.1.5 on 4.5.0
# --------------------------------------------------------------------
# latency: 336 us, #4/4, CPU#1 | (M:desktop VP:0, KP:0, SP:0 HP:0 #P:4)
# | task: gnome-shell-4858 (uid:1000 nice:0 policy:0 rt_prio:0)
# -----------------------------

 gnome-sh-4858   1d...  336us : gen8_emit_request <--i915_add_request
 gnome-sh-4858   1d...  337us : trace_hardirqs_on <--i915_add_request
 gnome-sh-4858   1d...  341us : <stack trace>
   => gen8_emit_request
   => __i915_add_request
   => i915_gem_execbuffer_retire_commands
   => intel_execlists_submission
   => i915_gem_do_execbuffer.isra.18
   => i915_gem_execbuffer2
   => drm_ioctl
   => vfs_ioctl
   => do_vfs_ioctl
   => SyS_ioctl
   => entry_SYSCALL_64_fastpath
Ftrace

wakeup/wakeup-rt Tracer

# wakeup latency trace v1.1.5 on 4.5.0
# ---------------------------------------------
# latency: 10 us, #4/4, CPU#1 | (M:desktop VP:0, KP:0, SP:0 HP:0 #P:4)
# -----------------------------
# | task: kworker/1:1H-2312 (uid:0 nice:-20 policy:0 rt_prio:0)
# -----------------------------

<idle>-0 1dNs. 1us : 0:120:R + [001] 2312:100:R kworker/1:1H
<idle>-0 1dNs. 2us : ttwu_do_activate.constprop.70 <-try_to_wake_up
<idle>-0 1d... 10us : __schedule <-schedule
<idle>-0 1d... 10us : 0:120:R ==> [001] 2312:100:R kworker/1:1H
Other Tools
Other Tools

**Latency Tracker**

- Many Analyses, single module
  - Wakeup Latency
  - Off-CPU latency
  - Syscall Latency
  - I/O Request Latency
  - Views in Ftrace or LTTng

**eBPF, LatencyTop**
## Other Tools

### Perf sched

```plaintext
# perf sched record sleep 0.01
# perf sched latency
```

<table>
<thead>
<tr>
<th>Task</th>
<th>Runtime ms</th>
<th>Switches</th>
<th>Average delay ms</th>
<th>Maximum delay ms</th>
<th>Maximum delay at</th>
</tr>
</thead>
<tbody>
<tr>
<td>kworker/u8:2:10930</td>
<td>0.161 ms</td>
<td>1</td>
<td>avg: 0.140 ms</td>
<td>max: 0.140 ms</td>
<td>max at: 136693.341916 s</td>
</tr>
<tr>
<td>kworker/2:1:11382</td>
<td>0.058 ms</td>
<td>3</td>
<td>avg: 0.043 ms</td>
<td>max: 0.075 ms</td>
<td>max at: 136693.376069 s</td>
</tr>
<tr>
<td>rcu_sched:7</td>
<td>0.395 ms</td>
<td>8</td>
<td>avg: 0.041 ms</td>
<td>max: 0.155 ms</td>
<td>max at: 136693.283139 s</td>
</tr>
<tr>
<td>rcuos/2:25</td>
<td>0.183 ms</td>
<td>4</td>
<td>avg: 0.039 ms</td>
<td>max: 0.146 ms</td>
<td>max at: 136693.295139 s</td>
</tr>
<tr>
<td>kworker/3:0:11129</td>
<td>0.052 ms</td>
<td>2</td>
<td>avg: 0.023 ms</td>
<td>max: 0.028 ms</td>
<td>max at: 136693.280035 s</td>
</tr>
<tr>
<td>kworker/0:1:11383</td>
<td>0.061 ms</td>
<td>3</td>
<td>avg: 0.020 ms</td>
<td>max: 0.029 ms</td>
<td>max at: 136693.283007 s</td>
</tr>
<tr>
<td>soffice.bin:29258</td>
<td>0.132 ms</td>
<td>1</td>
<td>avg: 0.012 ms</td>
<td>max: 0.012 ms</td>
<td>max at: 136693.298755 s</td>
</tr>
<tr>
<td>nautilus:6071</td>
<td>0.048 ms</td>
<td>1</td>
<td>avg: 0.012 ms</td>
<td>max: 0.012 ms</td>
<td>max at: 136693.310336 s</td>
</tr>
<tr>
<td>chrome:(3)</td>
<td>0.508 ms</td>
<td>4</td>
<td>avg: 0.011 ms</td>
<td>max: 0.012 ms</td>
<td>max at: 136693.316332 s</td>
</tr>
<tr>
<td>SCTP timer:3265</td>
<td>0.324 ms</td>
<td>11</td>
<td>avg: 0.011 ms</td>
<td>max: 0.013 ms</td>
<td>max at: 136693.374016 s</td>
</tr>
<tr>
<td>gmain:(2)</td>
<td>0.186 ms</td>
<td>2</td>
<td>avg: 0.010 ms</td>
<td>max: 0.011 ms</td>
<td>max at: 136693.278787 s</td>
</tr>
<tr>
<td>sleep:11404</td>
<td>1.012 ms</td>
<td>2</td>
<td>avg: 0.009 ms</td>
<td>max: 0.012 ms</td>
<td>max at: 136693.375922 s</td>
</tr>
<tr>
<td>kworker/u8:3:11157</td>
<td>0.012 ms</td>
<td>1</td>
<td>avg: 0.008 ms</td>
<td>max: 0.008 ms</td>
<td>max at: 136693.341930 s</td>
</tr>
<tr>
<td>perf:11403</td>
<td>0.656 ms</td>
<td>1</td>
<td>avg: 0.007 ms</td>
<td>max: 0.007 ms</td>
<td>max at: 136693.376070 s</td>
</tr>
<tr>
<td>rcuos/0:9</td>
<td>0.034 ms</td>
<td>2</td>
<td>avg: 0.006 ms</td>
<td>max: 0.009 ms</td>
<td>max at: 136693.277007 s</td>
</tr>
<tr>
<td>irq/28-iwlwifi:695</td>
<td>0.000 ms</td>
<td>21</td>
<td>avg: 0.005 ms</td>
<td>max: 0.007 ms</td>
<td>max at: 136693.348238 s</td>
</tr>
<tr>
<td>ips-monitor:671</td>
<td>0.020 ms</td>
<td>1</td>
<td>avg: 0.004 ms</td>
<td>max: 0.004 ms</td>
<td>max at: 136693.375995 s</td>
</tr>
<tr>
<td>rcuos/3:32</td>
<td>0.015 ms</td>
<td>2</td>
<td>avg: 0.004 ms</td>
<td>max: 0.004 ms</td>
<td>max at: 136693.283155 s</td>
</tr>
<tr>
<td>migration/3:28</td>
<td>0.000 ms</td>
<td>1</td>
<td>avg: 0.002 ms</td>
<td>max: 0.002 ms</td>
<td>max at: 136693.275081 s</td>
</tr>
</tbody>
</table>

**TOTAL:** 3.857 ms | 71 |
Other Tools

**Latency Tracker**
- Many Analyses, single module
  - Wakeup Latency
  - Off-CPU latency
  - Syscall Latency
  - I/O Request Latency
- Views in Ftrace or LTTng

**eBPF, LatencyTop**
• What happens **before** kernel receives an interrupt?
• Does a tracer affect the test system itself?
• What about regions in kernel that you can't trace?
• Can we observe the system without being in the critical path?
• What happens **before** kernel receives an interrupt?
• Does a tracer affect the test system itself?
• What about regions in kernel that you can't trace?
• Can we observe the system without being in the critical path?

Precise instruction and data flow straight from processor!
Hardware Tracing
Hardware Tracing 101

Debugging embedded systems: using hardware tricks to trace program flow

Stuart R Ball

A trace buffer is a valuable debugging aid. It provides a history of your program’s behavior by recording explanatory “action codes” whenever your program reaches certain key points in its execution (Reference 1). In some small μC-based

If your embedded system is too small to provide even a trace buffer or a serial port, debugging aids, you can use a variety of hardware tricks to observe your program’s behavior.

By writing to unused addresses, your program can make a logic-analyzer store codes that trace the program’s execution flow. Here, a 74ACT138 3-to-8-line decoder generates a signal that clocks the analyzer whenever the program writes to a particular address.

Courtesy, EDN; Apr 23, 1998; 43, 9; ProQuest pg. 163
Mainstream Adoption

• ARM very early introduced EmbeddedICE
  • Direct access to the data and address bus of the CPU
  • External device controls through TAP (JTAG interfaced)
• ETM/PTM → CoreSight
• Intel starts with LBR support and moves on to BTS [8, 2]
• MIPS PDtrace
Hardware Tracing 101

Typical Arrangement

- 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)
- Decoding performed 'offline' after trace recording
- Can be instruction or data flow
Hardware Tracing 101

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)
Intel Processor Trace
Background

Intel LBR and BTS

- **LBR** - Last Branch Record
  - Save last $n$ number of branches
  - LBR Stack MSRs based - limited for comprehensive analysis
  - BTM messages can be sent on system bus

- **BTS** - Branch Trace Store
  - When enabled, BTM can save data in BTS buffer, generate interrupt when full and save
  - 24 bytes per branch [FROM(64);TO(64);PREDICTED(1/64)]
  - Heavy penalty. Sometimes 40x overheads! [8]
  - Designed for debugging scenarios primarily
Intel Processor Trace

- Control Flow Tracing
  - Record branches to deduce program flow
  - Configure MSRs, setup buffer & generate trace packets
  - Save Packets to buffer or send to 'transport layer'
- Trace Packets Overview
  - **PSB** (Packet Stream Boundary) : Heartbeat, every 4K packets
  - **PIP** (Paging Information Packet) : CR3 change
  - **TSC, OVF** (Overflow), **CBR** (Core:Bus) Packets
- Control Flow
  - **TNT** (Taken Not-Taken), except unconditional
  - **TIP** (Target IP) at branches, **FUP, MODE**
Intel PT

**Intel Processor Trace**

- **Cycle Accurate Mode**
  - **CYC** Packets: Cycle counter data to keep tab on instruction count, IPC, tracking wall-clock time

- **Timing**
  - **MTC** (Mini Timestamp Counter): More frequent, based on CTC (crystal clock counter) value (8 bit). Can be frequency adjusted. Used with TSC to get accurate timestamps for less cost. [TSC → TMA → MTC, MTC, MTC → TSC]
  - Decoder finds out accurate time offline through elaborate calculation (Refer Chapter 36, Intel Manual)
Using Intel PT

Configure and Enable PT → CPU → Intel PT Hardware → Intel PT Software Decoder → Reconstructed Execution Flow

Runtime Data → Intel PT Packet Data

Based on, Andi Kleen’s Presentation (TracingSummit 2015)
Using Intel PT

- Perf support
  - Perf driver configures and controls PT hardware
  - Generated trace data is dumped to an AUX buffer on top of Perf's buffer
  - Userspace Perf counterpart can decode the perf data
  - Accurate instruction profiling data
- GDB support
  - Accurate reverse debugging.
- Andi Kleen's simple-pt
## Hardware Trace Packets (Perf)

... Intel Processor Trace data: size 8544 bytes

<table>
<thead>
<tr>
<th>Address</th>
<th>Hexadecimal Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000000</td>
<td>02 82 02 82 02 82 02 82 02 82 02 82 02 82 02 82</td>
<td>PSB</td>
</tr>
<tr>
<td>00000010</td>
<td>00 00 00 00 00 00</td>
<td>PAD</td>
</tr>
<tr>
<td>00000016</td>
<td>19 ba 39 4d 7b 89 5e 04</td>
<td>TSC 0x45e897b4d39ba</td>
</tr>
<tr>
<td>0000001e</td>
<td>00 00 00 00 00 00 00 00</td>
<td>PAD</td>
</tr>
<tr>
<td>00000026</td>
<td>02 73 57 64 00 1c 00 00</td>
<td>TMA CTC 0x6457 FC 0x1c</td>
</tr>
<tr>
<td>0000002e</td>
<td>00 00</td>
<td>PAD</td>
</tr>
<tr>
<td>00000030</td>
<td>02 03 27 00</td>
<td>CBR 0x27</td>
</tr>
<tr>
<td>00000034</td>
<td>02 23</td>
<td>PSBEND</td>
</tr>
<tr>
<td>00000036</td>
<td>59 8b</td>
<td>MTC 0x8b</td>
</tr>
<tr>
<td>00000038</td>
<td>59 8c</td>
<td>MTC 0x8c</td>
</tr>
<tr>
<td>00000039</td>
<td>02 03 27 00</td>
<td>CBR 0x27</td>
</tr>
<tr>
<td>00000040</td>
<td>02 03 27 00</td>
<td>CBR 0x27</td>
</tr>
<tr>
<td>00000044</td>
<td>00 00</td>
<td>PAD</td>
</tr>
<tr>
<td>00000046</td>
<td>59 8b</td>
<td>MTC 0x8b</td>
</tr>
<tr>
<td>00000048</td>
<td>59 8c</td>
<td>MTC 0x8c</td>
</tr>
<tr>
<td>00000049</td>
<td>02 03 27 00</td>
<td>CBR 0x27</td>
</tr>
<tr>
<td>0000004c</td>
<td>02 03 27 00</td>
<td>CBR 0x27</td>
</tr>
<tr>
<td>0000004e</td>
<td>00 00</td>
<td>PAD</td>
</tr>
<tr>
<td>00000050</td>
<td>59 8b</td>
<td>MTC 0x8b</td>
</tr>
<tr>
<td>00000052</td>
<td>59 8c</td>
<td>MTC 0x8c</td>
</tr>
<tr>
<td>00000053</td>
<td>02 03 27 00</td>
<td>CBR 0x27</td>
</tr>
<tr>
<td>00000056</td>
<td>02 03 27 00</td>
<td>CBR 0x27</td>
</tr>
<tr>
<td>00000058</td>
<td>00 00</td>
<td>PAD</td>
</tr>
<tr>
<td>0000005a</td>
<td>59 8b</td>
<td>MTC 0x8b</td>
</tr>
<tr>
<td>0000005c</td>
<td>59 8c</td>
<td>MTC 0x8c</td>
</tr>
<tr>
<td>0000005d</td>
<td>02 03 27 00</td>
<td>CBR 0x27</td>
</tr>
<tr>
<td>0000005f</td>
<td>02 03 27 00</td>
<td>CBR 0x27</td>
</tr>
<tr>
<td>00000061</td>
<td>00 00</td>
<td>PAD</td>
</tr>
<tr>
<td>00000063</td>
<td>59 8b</td>
<td>MTC 0x8b</td>
</tr>
<tr>
<td>00000065</td>
<td>59 8c</td>
<td>MTC 0x8c</td>
</tr>
<tr>
<td>00000067</td>
<td>02 03 27 00</td>
<td>CBR 0x27</td>
</tr>
<tr>
<td>00000069</td>
<td>02 03 27 00</td>
<td>CBR 0x27</td>
</tr>
<tr>
<td>0000006b</td>
<td>00 00</td>
<td>PAD</td>
</tr>
<tr>
<td>0000006d</td>
<td>59 8b</td>
<td>MTC 0x8b</td>
</tr>
<tr>
<td>0000006f</td>
<td>59 8c</td>
<td>MTC 0x8c</td>
</tr>
<tr>
<td>00000071</td>
<td>02 03 27 00</td>
<td>CBR 0x27</td>
</tr>
<tr>
<td>00000073</td>
<td>02 03 27 00</td>
<td>CBR 0x27</td>
</tr>
<tr>
<td>00000075</td>
<td>00 00</td>
<td>PAD</td>
</tr>
<tr>
<td>00000077</td>
<td>59 8b</td>
<td>MTC 0x8b</td>
</tr>
<tr>
<td>00000079</td>
<td>59 8c</td>
<td>MTC 0x8c</td>
</tr>
<tr>
<td>0000007b</td>
<td>02 03 27 00</td>
<td>CBR 0x27</td>
</tr>
<tr>
<td>0000007d</td>
<td>02 03 27 00</td>
<td>CBR 0x27</td>
</tr>
<tr>
<td>0000007f</td>
<td>00 00</td>
<td>PAD</td>
</tr>
<tr>
<td>00000081</td>
<td>59 8b</td>
<td>MTC 0x8b</td>
</tr>
<tr>
<td>00000083</td>
<td>59 8c</td>
<td>MTC 0x8c</td>
</tr>
<tr>
<td>00000085</td>
<td>02 03 27 00</td>
<td>CBR 0x27</td>
</tr>
<tr>
<td>00000087</td>
<td>02 03 27 00</td>
<td>CBR 0x27</td>
</tr>
<tr>
<td>00000089</td>
<td>00 00</td>
<td>PAD</td>
</tr>
<tr>
<td>0000008b</td>
<td>59 8b</td>
<td>MTC 0x8b</td>
</tr>
<tr>
<td>0000008d</td>
<td>59 8c</td>
<td>MTC 0x8c</td>
</tr>
<tr>
<td>0000008f</td>
<td>02 03 27 00</td>
<td>CBR 0x27</td>
</tr>
<tr>
<td>00000091</td>
<td>02 03 27 00</td>
<td>CBR 0x27</td>
</tr>
<tr>
<td>00000093</td>
<td>00 00</td>
<td>PAD</td>
</tr>
<tr>
<td>00000095</td>
<td>59 8b</td>
<td>MTC 0x8b</td>
</tr>
<tr>
<td>00000097</td>
<td>59 8c</td>
<td>MTC 0x8c</td>
</tr>
<tr>
<td>00000099</td>
<td>02 03 27 00</td>
<td>CBR 0x27</td>
</tr>
<tr>
<td>0000009b</td>
<td>02 03 27 00</td>
<td>CBR 0x27</td>
</tr>
<tr>
<td>0000009d</td>
<td>00 00</td>
<td>PAD</td>
</tr>
<tr>
<td>0000009f</td>
<td>59 8b</td>
<td>MTC 0x8b</td>
</tr>
<tr>
<td>000000a1</td>
<td>59 8c</td>
<td>MTC 0x8c</td>
</tr>
<tr>
<td>000000a3</td>
<td>02 03 27 00</td>
<td>CBR 0x27</td>
</tr>
<tr>
<td>000000a5</td>
<td>02 03 27 00</td>
<td>CBR 0x27</td>
</tr>
<tr>
<td>000000a7</td>
<td>00 00</td>
<td>PAD</td>
</tr>
<tr>
<td>000000a9</td>
<td>59 8b</td>
<td>MTC 0x8b</td>
</tr>
<tr>
<td>000000ab</td>
<td>59 8c</td>
<td>MTC 0x8c</td>
</tr>
<tr>
<td>000000ad</td>
<td>02 03 27 00</td>
<td>CBR 0x27</td>
</tr>
<tr>
<td>000000ae</td>
<td>02 03 27 00</td>
<td>CBR 0x27</td>
</tr>
<tr>
<td>000000b0</td>
<td>00 00</td>
<td>PAD</td>
</tr>
<tr>
<td>000000b2</td>
<td>59 8b</td>
<td>MTC 0x8b</td>
</tr>
<tr>
<td>000000b4</td>
<td>59 8c</td>
<td>MTC 0x8c</td>
</tr>
<tr>
<td>000000b6</td>
<td>02 03 27 00</td>
<td>CBR 0x27</td>
</tr>
<tr>
<td>000000b8</td>
<td>02 03 27 00</td>
<td>CBR 0x27</td>
</tr>
<tr>
<td>000000ba</td>
<td>00 00</td>
<td>PAD</td>
</tr>
<tr>
<td>000000bc</td>
<td>59 8b</td>
<td>MTC 0x8b</td>
</tr>
<tr>
<td>000000bd</td>
<td>59 8c</td>
<td>MTC 0x8c</td>
</tr>
<tr>
<td>000000be</td>
<td>02 03 27 00</td>
<td>CBR 0x27</td>
</tr>
<tr>
<td>000000c0</td>
<td>02 03 27 00</td>
<td>CBR 0x27</td>
</tr>
<tr>
<td>000000c2</td>
<td>00 00</td>
<td>PAD</td>
</tr>
<tr>
<td>000000c4</td>
<td>59 8b</td>
<td>MTC 0x8b</td>
</tr>
<tr>
<td>000000c6</td>
<td>59 8c</td>
<td>MTC 0x8c</td>
</tr>
<tr>
<td>000000c8</td>
<td>02 03 27 00</td>
<td>CBR 0x27</td>
</tr>
<tr>
<td>000000ca</td>
<td>02 03 27 00</td>
<td>CBR 0x27</td>
</tr>
<tr>
<td>000000cc</td>
<td>00 00</td>
<td>PAD</td>
</tr>
<tr>
<td>000000ce</td>
<td>59 8b</td>
<td>MTC 0x8b</td>
</tr>
<tr>
<td>000000d0</td>
<td>59 8c</td>
<td>MTC 0x8c</td>
</tr>
<tr>
<td>000000d2</td>
<td>02 03 27 00</td>
<td>CBR 0x27</td>
</tr>
<tr>
<td>000000d4</td>
<td>02 03 27 00</td>
<td>CBR 0x27</td>
</tr>
<tr>
<td>000000d6</td>
<td>00 00</td>
<td>PAD</td>
</tr>
<tr>
<td>000000d8</td>
<td>59 8b</td>
<td>MTC 0x8b</td>
</tr>
<tr>
<td>000000da</td>
<td>59 8c</td>
<td>MTC 0x8c</td>
</tr>
<tr>
<td>000000dc</td>
<td>02 03 27 00</td>
<td>CBR 0x27</td>
</tr>
<tr>
<td>000000ed</td>
<td>02 03 27 00</td>
<td>CBR 0x27</td>
</tr>
<tr>
<td>000000f0</td>
<td>00 00</td>
<td>PAD</td>
</tr>
<tr>
<td>000000f2</td>
<td>59 8b</td>
<td>MTC 0x8b</td>
</tr>
<tr>
<td>000000f4</td>
<td>59 8c</td>
<td>MTC 0x8c</td>
</tr>
<tr>
<td>000000f6</td>
<td>02 03 27 00</td>
<td>CBR 0x27</td>
</tr>
<tr>
<td>000000f8</td>
<td>02 03 27 00</td>
<td>CBR 0x27</td>
</tr>
<tr>
<td>000000fa</td>
<td>00 00</td>
<td>PAD</td>
</tr>
<tr>
<td>000000fc</td>
<td>59 8b</td>
<td>MTC 0x8b</td>
</tr>
<tr>
<td>000000fe</td>
<td>59 8c</td>
<td>MTC 0x8c</td>
</tr>
</tbody>
</table>
## Timing

... Intel Processor Trace data: size 8544 bytes

<table>
<thead>
<tr>
<th>Offset</th>
<th>Data</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000000</td>
<td>02 82 02 82 02 82 02 82 02 82 02 82 02 82 02 82</td>
<td>PSB</td>
</tr>
<tr>
<td>00000010</td>
<td>00 00 00 00 00 00 00 00</td>
<td>PAD</td>
</tr>
</tbody>
</table>
| 00000016 | 19 ba 39 4d 7b 89 5e 04 | TSC 0x45e897b4d39ba |}

<table>
<thead>
<tr>
<th>Offset</th>
<th>Data</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000026</td>
<td>02 73 57 64 00 1c 00 00</td>
<td>TMA CTC 0x6457 FC 0x1c</td>
</tr>
<tr>
<td>0000002e</td>
<td>00 00</td>
<td>PAD</td>
</tr>
<tr>
<td>00000030</td>
<td>02 03 27 00</td>
<td>CBR 0x27</td>
</tr>
<tr>
<td>00000034</td>
<td>02 23</td>
<td>PSBEND</td>
</tr>
<tr>
<td>00000036</td>
<td>59 8b</td>
<td>MTC 0x8b</td>
</tr>
<tr>
<td>00000038</td>
<td>59 8c</td>
<td>MTC 0x8c</td>
</tr>
<tr>
<td>0000003d</td>
<td>f8</td>
<td>TNT TTTTNN (6)</td>
</tr>
<tr>
<td>00000305</td>
<td>06 00 00</td>
<td>TNT T (1)</td>
</tr>
<tr>
<td>00000308</td>
<td>4d e0 3c 6d 9c</td>
<td>TIP 0x9c6d3ce0</td>
</tr>
<tr>
<td>0000030d</td>
<td>1c 00 00</td>
<td>TNT TTN (3)</td>
</tr>
<tr>
<td>00000310</td>
<td>2d f0 3c</td>
<td>TIP 0x3cf0</td>
</tr>
<tr>
<td>00000313</td>
<td>06</td>
<td>TNT T (1)</td>
</tr>
<tr>
<td>00000314</td>
<td>59 2e</td>
<td>MTC 0x2e</td>
</tr>
<tr>
<td>00000316</td>
<td>94</td>
<td>TNT NNTNTN (6)</td>
</tr>
<tr>
<td>00000317</td>
<td>a8</td>
<td>TNT NTNTNN (6)</td>
</tr>
<tr>
<td>00000318</td>
<td>a6</td>
<td>TNT NTNNNT (6)</td>
</tr>
</tbody>
</table>
Conditional Branches

... Intel Processor Trace data: size 8544 bytes

- 00000000: 02 82 02 82 02 82 02 82 02 82 02 82 02 82 02 82 PSB
- 00000010: 00 00 00 00 00 00 PAD
- 00000016: 19 ba 39 4d 7b 89 5e 04 TSC 0x45e897b4d39ba
- 0000001e: 00 00 00 00 00 00 PAD
- 00000026: 02 73 57 64 00 1c 00 00 TMA CTC 0x6457 FC 0x1c
- 0000002e: 00 00 PAD
- 00000030: 02 03 27 00 CBR 0x27
- 00000034: 02 23 PSBEND
- 00000036: 59 8b MTC 0x8b
- 00000038: 59 8c MTC 0x8c
- 00000304: f8 TNT TTTTNN (6)
- 00000305: 06 00 00 TNT T (1)
- 00000308: 4d e0 3c 6d 9c TIP 0x9c6d3ce0
- 0000030d: 1c 00 00 TNT TTN (3)
- 00000310: 2d f0 3c TIP 0x3cf0
- 00000313: 06 TNT T (1)
- 00000314: 59 2e MTC 0x2e
- 00000316: 94 TNT NNTNTN (6)
- 00000317: a8 TNT NTNTNN (6)
- 00000318: a6 TNT NTTNTT (6)
### Indirect Branches

... Intel Processor Trace data: size 8544 bytes

<table>
<thead>
<tr>
<th>Address</th>
<th>Bytes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000000:</td>
<td>02 82 02 82 02 82 02 82 02 82 02 82 02 82 02 82 02 82 PSB</td>
<td></td>
</tr>
<tr>
<td>00000010:</td>
<td>00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 PAD</td>
<td></td>
</tr>
<tr>
<td>00000016:</td>
<td>19 ba 39 4d 7b 89 5e 04 TSC 0x45e897b4d39ba</td>
<td></td>
</tr>
<tr>
<td>0000001e:</td>
<td>00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 PAD</td>
<td></td>
</tr>
<tr>
<td>00000026:</td>
<td>02 73 57 64 00 1c 00 00 TMA CTC 0x6457 FC 0x1c</td>
<td></td>
</tr>
<tr>
<td>0000002e:</td>
<td>00 00 CBR 0x27</td>
<td></td>
</tr>
<tr>
<td>00000030:</td>
<td>02 03 27 00 PSBEND</td>
<td></td>
</tr>
<tr>
<td>00000034:</td>
<td>02 23 MTC 0x8b</td>
<td></td>
</tr>
<tr>
<td>00000036:</td>
<td>59 8b MTC 0x8c</td>
<td></td>
</tr>
<tr>
<td>00000038:</td>
<td>59 8c</td>
<td></td>
</tr>
<tr>
<td>000000304:</td>
<td>f8 TNT TTTTNN (6)</td>
<td></td>
</tr>
<tr>
<td>000000305:</td>
<td>06 00 00 TNT T (1)</td>
<td></td>
</tr>
<tr>
<td>000000308:</td>
<td>4d e0 3c 6d 9c TIP 0x9c6d3ce0</td>
<td></td>
</tr>
<tr>
<td>00000030d:</td>
<td>1c 00 00 TNT TTN (3)</td>
<td></td>
</tr>
<tr>
<td>000000310:</td>
<td>2d f0 3c TIP 0x3cf0</td>
<td></td>
</tr>
<tr>
<td>000000313:</td>
<td>06 TNT T (1)</td>
<td></td>
</tr>
<tr>
<td>000000314:</td>
<td>59 2e MTC 0x2e</td>
<td></td>
</tr>
<tr>
<td>000000316:</td>
<td>94 TNT NNTNTN (6)</td>
<td></td>
</tr>
<tr>
<td>000000317:</td>
<td>a8 TNT NTNTNN (6)</td>
<td></td>
</tr>
<tr>
<td>000000318:</td>
<td>a6 TNT NTNNTT (6)</td>
<td></td>
</tr>
</tbody>
</table>
Program Control Flow (simple-pt)

826.329 [+0.019] [+ 7] intmod_ioctl+88
               [+ 7] intmod_ioctl+148 -> trace_hardirqs_off
               [+ 3] intmod_ioctl+290 -> trace_hardirqs_on
826.490 [+0.161] [+ 5] trace_hardirqs_on+27
               [+ 8] intmod_ioctl+184 -> trace_hardirqs_off
               __ndelay+38 -> delay_tsc
               delay_tsc+24 -> preempt_count_add
               preempt_count_add+27
826.503 [+0.013] [+ 10] .
826.507 [+0.003] [+ 9] .
                    .
826.559 [+0.008] [+ 3] delay_tsc+157
               [+ 1] intmod_ioctl+198 -> trace_hardirqs_on
826.573 [+0.014] [+ 7] intmod_ioctl+118
Intel PT Benchmarks

**Synthetic Tests**

- **Test Setup**
  - Skylake i5 6600K (3.9Ghz), controlled with simple-pt
  - 2MB trace buffer
  - MTC threshold (TSC update) every 512 cycles (accuracy)
  - PSB packet every 2K bytes (better decoder sync and recovery)
- **Wide range of experiments**
  - Image processing to arithmetic intensive tests
  - Mainly memory overhead as PT hardware is 'in parallel'
  - Overhead of V8 test
## Intel PT Benchmarks

### Synthetic Tests

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Overhead (%)</th>
<th>Overhead V8 (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TailFact</td>
<td>22.91</td>
<td>-</td>
</tr>
<tr>
<td>ParseInt</td>
<td>9.65</td>
<td>10.36</td>
</tr>
<tr>
<td>Fib</td>
<td>5.86</td>
<td>5.80</td>
</tr>
<tr>
<td>RandMatStat</td>
<td>2.58</td>
<td>20.00</td>
</tr>
<tr>
<td>Canny_NoOptimize</td>
<td>2.55</td>
<td>-</td>
</tr>
<tr>
<td>PiSum</td>
<td>2.47</td>
<td>6.20</td>
</tr>
<tr>
<td>Canny_Optimize</td>
<td>2.34</td>
<td>-</td>
</tr>
<tr>
<td>Sort</td>
<td>1.05</td>
<td>6.06</td>
</tr>
<tr>
<td>RandMatMul</td>
<td><strong>0.83</strong></td>
<td><strong>11.08</strong></td>
</tr>
</tbody>
</table>

Lots of TIP packets

DGEMM using Intel AVX
Tracing the Tracers (Syscall Latency)

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

Timing Granularity

- Can vary from milliseconds (highly coarse) to around 14ns (CYC mode)
Intel PT

VM Analysis

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

• Collection of trace hardware blocks for ARM
• Advancements from ETMv1, v3 and PFT → ETMv4 [4]
• 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 [9]
• Maturing Linux Kernel and vendor support for multiple chips
**ETMv4**

- Major revision [4], highly configurable
  - Insn only for A family. Data+Insn only for M and R family
    - \textbf{P0} (Insn), P1 and P2 (Data) elements, Other elements
    - \textbf{P0} : Atom elements (E/N), Q elements (cycle count)
    - \textbf{P0} : Branch, Synchronization, Exceptions, TimeStamp,
      Conditional (C), Result (R), Mispredict etc.
  - Decoding same way as libipt (Intel PT decoder lib)
  - \textbf{Trace Control} with CSAL as in HAT [10] or now, Perf!
    - Expose configuration registers by mmaping them
    - Trace start and stop. \textbf{Decoding} with \textbf{OpenCSD}
ARM CoreSight

SoC View

Courtesy, ARM
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, regular upstream in progress
• Decoding
  • **ptm2human** [5] decodes ETMv4 packets but is not as mature as other alternatives. Not recommended
  • **OpenCSD** (https://github.com/Linaro/OpenCSD/tree/arm-dev) provides ETMv4 debugging. Perf integration is underway.
Baremetal Tracing
Chip Boundary

CPU

CPU

CPU

CPU
Chip Boundary

CPU  CPU

CPU  CPU

DSP  FPGA

GPU
Heterogeneous System

**TI Keystone 2**

- 66AK2H12 SoC
- 4 x ARM Cortex A15 core
- 8 x C66 CorePacs DSP
  - Real-time micro-kernel - SYS/BIOS
  - Inter-core communication with an API
- 2GB DDR RAM
- 6MB Shared Memory
- All the latest and cool peripheral controllers
• How can we approach tracing such heterogeneous systems?
• Is it possible to gather traces from multiple cores?
• How would we synchronize such traces and view them?
There is a solution...
There is a solution...
BareCTF

Workflow

 Courtesy, Philippe Proulx (EfficiOS) : http://lttng.org/blog/2015/07/21/barectf-2/
BareCTF

Workflow

Courtesy, Philippe Proulx (EfficiOS) : http://lttg.org/blog/2015/07/21/barectf-2/
/* Instantiates the tracing context and initializes every structure needed to take care of the recorded events and stored packets. */
int8_t barectf_init(void);

/* Returns the tracing context. */
barectf_ctx_t *barectf_get_ctx(void);

/* Specifies the way to access 64bits timestamps on the targeted device. */
uint64_t barectf_get_clock(void *ctx);

/* Initializes (if needed) the counter used to get timestamps. */
void barectf_init_clock(void *ctx);

/* Opens a new packet containing recorded events. */
void barectf_open_packet(void *ctx);

/* Takes care of a full packet. The packet can be sent to the host, put in another memory location, discarded... */
void barectf_close_packet(void *ctx);

/* Finalizes the tracing session. */
int8_t barectf_close(void);
Experimentation

Workflow

Master

Slave

barectf consumer

master code

LTTng

slave code

barectf
Experimentation

**Synchronization**

- *Master* can interrupt a *Slave* processor [MessageQ]
- Shared memory between Master or Slave

![Diagram showing synchronization between Master and Slave processors with LTTng and barectf](image)
Experimentation

Sobel's Filter for Edge Detection
Experimentation

Experimentation

Image Processing Task

SYS/BIOS Modules Instrumented with tracepoints

process_bmp()
process_rgb()
dummy_workload()

MessageQ_put()
MessageQ_get()
Trace View (Non-Synchronized)
Trace View (Synchronized)
Trace View (Synchronized)
Tracing

Trace View (Synchronized)
Tracing

Trace View (Synchronized)
Trace View (Synchronized)
Trace View (Synchronized)
Tracing

Detecting Unwanted Latency

Image processing function
Waiting for replies from DSPs

ARM

Idle
Main task

DSP #0

Process_bmp()
Process_rgb()

Idle
Main task

DSP #1

Process_bmp()
Dummy_workload()
Process_rgb()

Idle

DSP #2

Process_bmp()
Dummy_workload()
Process_rgb()
Tracing

Detecting Unwanted Latency

![Diagram showing the tracing process with timelines for ARM, DSP #0, DSP #1, and DSP #2, highlighting the image processing function, waiting for replies from DSPs, and tasks like Process_bmp() and Process_rgb().]
The Future

Hardware Tracing

- Perf is emerging as a standard way to use hardware tracing
- Accurate instruction profiling
- Applications in security, performance, dynamic code analysis
- Trace Viewing integration with TraceCompass

Baremetal Tracing

- Moving towards standardization
  - HSA standards specify Performance Counter APIs
  - Dedicated APIs for trace gathering as well
- Better Trace Viewing and Analysis
References

[1] Debug and Trace for Multicore SoCs, ARM Whitepaper, 2008
[4] ARM® Embedded Trace Macrocell Architecture Specification (ETMv4.0 to ETMv4.2)
[8] Exploiting Hardware Advances for Software Testing and Debugging, ICSE’11, ACM
References

- HSA Foundation Standards: http://www.hsafoundation.com/standards/
- OpenCSD: https://github.com/Linaro/OpenCSD
- CoreSight, Perf and the OpenCSD Library: http://www.linaro.org/blog/core-dump/coresight-perf-and-the-opencsd-library/
- barectf 2: Continuous Bare-Metal Tracing on the Parallella Board: http://lttng.org/blog/2015/07/21/barectf-2/

All the text and images in this presentation drawn by the authors are released under CC-BY-SA. Images not drawn by authors have been attributed on slides. Glyph and Icons have been used from Ion Icons and Font Awesome which are MIT and GPL licensed.
Acknowledgments

Thanks to EfficiOS, Ericsson Montréal and DORSAL Lab, Polytechnique Montreal for the awesome work on LTTng/UST, TraceCompass and BareCTF. Thanks to DiaMon Workgroup for this opportunity to present.
Questions?

suchakrapani.sharma@polymtl.ca
thomas.bertauld@gmail.com

@tuxology
http://suchakra.in