Tracing Linux Kernel Events via STM

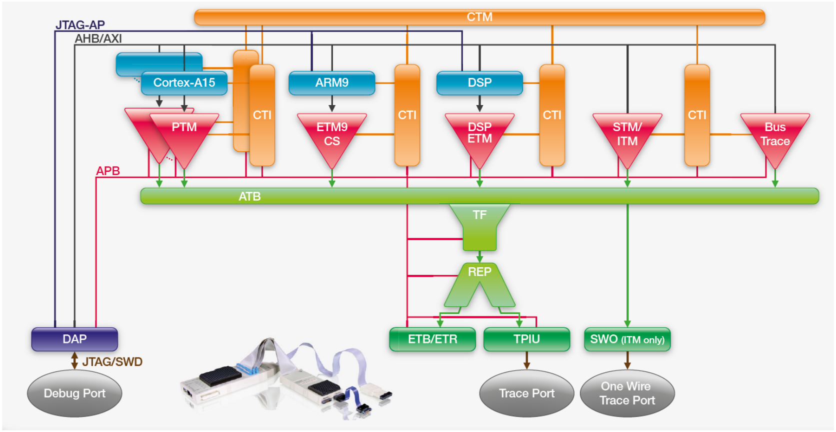
Michael Trimarchi <michael at amarulasolutions.com>
Stefan Kolbinger <Stefan.Kolbinger at lauterbach.com>
This work was done in collaboration with Lauterbach using Lauterbach Power Trace

June 27, 2014

Intro ARM Coresight STM

- Coresight is a standard debug interface for debug components in an embedded System on Chip
- STM is a Coresight component that implements a low latency and high bandwidth printf style debug capability
- Scalable solution enabling multi-processors and processes to access STM without being aware of others; STM supports 65,536 channels enabling significant scalability

ARM Coresight STM



Why do we need to use tracing over STM?

- The Software development costs increase compared to hardware one Modern SoC are very complex system
- Optimization and software quality are critical to ensure product success
- Traditional debug and tracing mechanism implement in modern OS are invasive

Linux Tracer with tracepoints

- We use tracepoints
- A tracepoint placed in code provides a hook to call a function (probe) that you can provide at runtime.
- tracepoint can be "off" it has no effect, except for adding a tiny time penalty
- Tracepoints "on", the function you provide is called each time the tracepoint is executed, in the execution context of the caller. You can put tracepoints at important locations in the code.

Tracing Linux Kernel Events via STM

- Use Linux Kernel tracing architecture
- Add new tracepoints hooks
- Send trace data over STM channel
- Max event size is 8 Words (32bit)
- Use one channel x CPU for increase parallelism
- Make it portables to more SoC and with limited changes to the Linux Kernel
- Define a new standard for hardware debugger that can read data from STM channels and can be ported even to no open-source Operating System

Trace with Linux

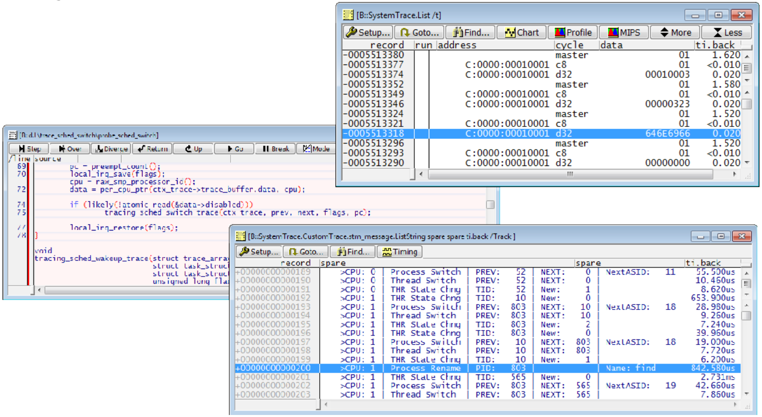
- PowerTrace II system
- Dual-core cortex A9 system
- STM's compatible core
- Widely used platform
- Compatible with OMAP5 platform
- Supported in mainline

What are we tracing?

- Process/Thread Creation
- Process/Thread Switch
- Process/Thread Deletion
- Process Rename
- Thread State Change
- Interrupt Exception Entry/Exit
- ISR Entry/Exit
- ASID Assign

Tracing Linux kernel events via STM: example

This is an example of the execution of a tracepoint. In this case we have a Process Context Switch, that generate the event to the STM channel



```
      record  run address             cycle   data         ti.back
-0005513380                           master          01     1.620
-0005513377         C:0000:00010001   c8              01    <0.010
-0005513374         C:0000:00010001   d32       00010003     0.020
-0005513352                           master          01     1.580
-0005513349         C:0000:00010001   c8              01    <0.010
-0005513346         C:0000:00010001   d32       00000323     0.020
-0005513324                           master          01     1.520
-0005513321         C:0000:00010001   c8              01    <0.010
-0005513318         C:0000:00010001   d32       646E6966     0.020
-0005513296                           master          01     1.520
-0005513293         C:0000:00010001   c8              01    <0.010
-0005513290         C:0000:00010001   d32       00000000     0.020
```

[B::d.l \trace_sched_switch\probe_sched_switch]
Step Over Diverge Return Up Go Break Mode

```
line source
        pc = preempt_count();
        local_irq_save(flags);
        cpu = raw_smp_processor_id();
        data = per_cpu_ptr(ctx_trace->trace_buffer.data, cpu);

        if (likely(!atomic_read(&data->disabled)))
                tracing_sched_switch_trace(ctx_trace, prev, next, flags, pc);

        local_irq_restore(flags);
}

void
tracing_sched_wakeup_trace(struct trace_array
                           struct task_struct
                           struct task_struct
                           unsigned long fla
```

[B::SystemTrace.CustomTrace.stm_message.ListString spare spare ti.back /Track]
Setup... Goto... Find... Timing

```
           record  spare                                                  spare                ti.back
+0000000000000189    >CPU: 0 | Process Switch | PREV:   52 | NEXT:   0 | NextASID:  11     55.500us
+0000000000000190    >CPU: 0 | Thread Switch  | PREV:   52 | NEXT:   0 |                   10.460us
+0000000000000191    >CPU: 0 | THR State Chng | TID:    52 | New:    1 |                    8.620us
+0000000000000192    >CPU: 1 | THR State Chng | TID:    10 | New:    0 |                  653.900us
+0000000000000193    >CPU: 1 | Process Switch | PREV:  803 | NEXT:  10 | NextASID:  18     28.980us
+0000000000000194    >CPU: 1 | Thread Switch  | PREV:  803 | NEXT:  10 |                    9.260us
+0000000000000195    >CPU: 1 | THR State Chng | TID:   803 | New:    2 |                    7.240us
+0000000000000196    >CPU: 1 | THR State Chng | TID:   803 | New:    0 |                   39.960us
+0000000000000197    >CPU: 1 | Process Switch | PREV:   10 | NEXT: 803 | NextASID:  18     19.000us
+0000000000000198    >CPU: 1 | Thread Switch  | PREV:   10 | NEXT: 803 |                    7.720us
+0000000000000199    >CPU: 1 | THR State Chng | TID:    10 | New:    1 |                    6.200us
+0000000000000200    >CPU: 1 | Process Rename | PID:   803 |            | Name: find       842.580us
+0000000000000201    >CPU: 1 | THR State Chng | TID:   565 | New:    0 |                    2.731ms
+0000000000000202    >CPU: 1 | Process Switch | PREV:  803 | NEXT: 565 | NextASID:  19     42.660us
+0000000000000203    >CPU: 1 | Thread Switch  | PREV:  803 | NEXT: 565 |                    7.860us
```

PowerView can show the event on a graphics interface

Pending problems

- tracing is slow (up to 24uS event)
- tracing work on STM 1.0 from TI
- Stm Clock is not optimized (NDA required with TI)
- code is written on top of an old TI driver (cleanup is not sufficient)

Next steps...

- Find someone that has time.. (1)
- Rebase to a newer linux kernel and Coresight implementation (3.16 and Patrick Pratel patchset https://lkml.org/lkml/2012/12/19/331)
- Export more tracepoint events or create pluggable backend to the tracer
- Create a graphical visualization of the events on Lauterbach TRACE32(R) Combine hardware tracing information with Linux Kernel Tracing over STM
- Mainline kernel submission??? refer to point (1)