ELC 2011

The future of Tracing and Profiling for Power Management and Accelerators

Jean Pihet <j-pihet@ti.com>

Introduction

Background

- Work on ARMv7 support for oprofile/perf/ftrace
- Work on OMAP PM:
  - PM instrumentation,
  - (omap_)devices latency support,
  - Devices wake-up latencies measurements.

Tracing/profiling is used for : Debug, Profiling, Performance Measurements.

Tracing and profiling

Two types of tools:

- Profiling (i.e. Generate stats from events),
- Tracing (i.e. Collect events and generate a timeline).

This presentation is focusing on tracing using ftrace and the parsing tools (py)timechart.

Introduction

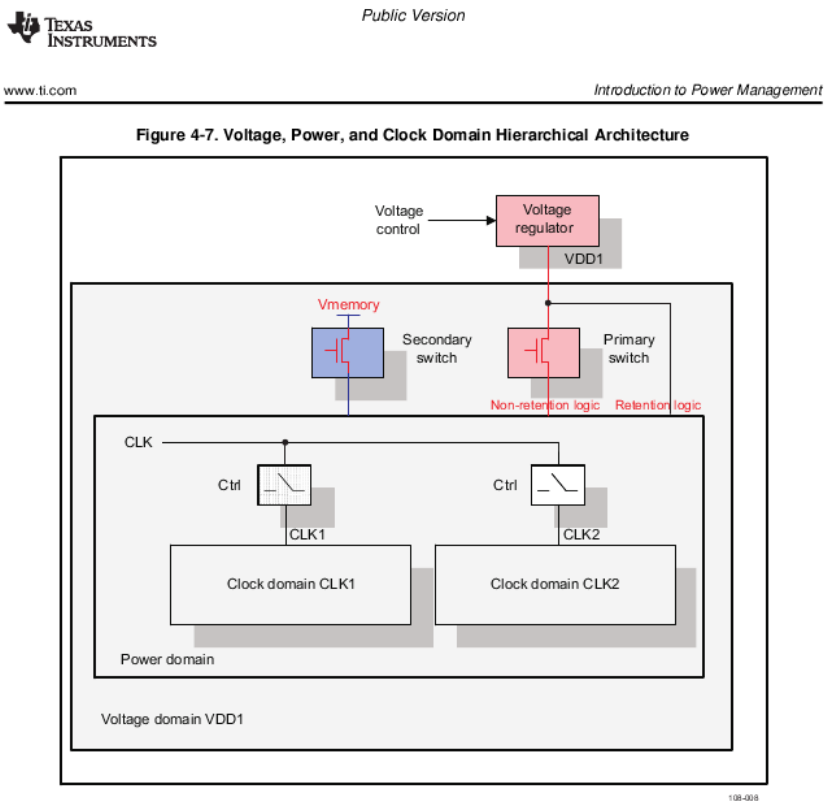
OMAP SoC PM

- Dynamic and hierarchical PM.
  Clock->Pwrdm->Voltdm->
  Voltage Regulators
- On-chip devices count & interfaces
- Multiple frameworks involved : cpuidle, cpufreq, runtime PM

Multiple accelerators

for MM, Crypto ...

Parsing tools & GUI



=> Traditional -static- tools are not suited anymore

=> Challenges for tracing on modern SoCs

Status of PM & Accel trace events

New PM trace API

- Added clock and power_domain events classes (in the old API)

- power:power_start, power:power_end => power:cpu_idle
- power:power_frequency => power:cpu_frequency
- power:machine_suspend is newly introduced

- 'type' field removed
- Old API & tracepoints kept for backward compatibility, to be removed (.41?).
  CONFIG_EVENT_POWER_TRACING_DEPRECATED introduced.

- Unification of cpufreq, cpuidle & suspend tracepoints
  - Tracepoints made generic (in drivers/cpu[freq,idle] and kernel/power code)
  - removal of duplicated events (in arch & framework code)

- OMAP tracepoints patches

Status of PM & Accel trace events

- Parsing tools : (py)timechart patches
- trace example : old vs new PM trace API

```
<idle>-0    [000]  73.946503: power_start: type=1 state=3 cpu_id=0                    OLD
<idle>-0    [000]  73.946503: cpu_idle: state=3 cpu_id=0                              NEW
<idle>-0    [000]  73.946533: power_domain_target: mpu_pwrdm state=1 cpu_id=0
<idle>-0    [000]  73.946533: power_domain_target: core_pwrdm state=3 cpu_id=0
<idle>-0    [000]  73.946594: power_domain_target: neon_pwrdm state=1 cpu_id=0
<idle>-0    [000]  73.946625: clock_disable: uart3_fck state=0 cpu_id=0
<idle>-0    [000]  73.946655: clock_disable: per_48m_fck state=0 cpu_id=0
<idle>-0    [000]  73.953949: clock_enable: per_48m_fck state=1 cpu_id=0
<idle>-0    [000]  73.953979: clock_enable: uart3_fck state=1 cpu_id=0
<idle>-0    [000]  73.954010: power_domain_target: dpll1_pwrdm state=2147484417 cpu_id=0
<idle>-0    [000]  73.954041: power_domain_target: per_pwrdm state=2147484417 cpu_id=0
<idle>-0    [000]  73.954041: power_domain_target: dss_pwrdm state=2147484417 cpu_id=0
<idle>-0    [000]  73.954071: power_domain_target: neon_pwrdm state=2147484417 cpu_id=0
<idle>-0    [000]  73.954071: power_domain_target: mpu_pwrdm state=2147484417 cpu_id=0
<idle>-0    [000]  73.954193: power_end: cpu_id=0                                    OLD
<idle>-0    [000]  73.954193: cpu_idle: state=4294967295 cpu_id=0                    NEW
```

- + pytimechart screenshots: cf. [1]

Status of PM & Accel trace events

HW accelerators

Problems

Adding new events

- Contributions to mainline kernel, in generic include files
- Reaction time : submit, review, discuss, re-submit, merge in tip kernel...
- API changes that are not generic enough are difficult to merge in

Change of events format, variations : More flexibility is needed

Tracing **non occuring** PM transitions

A power domain could not transition to the desired power state.
An extra tracepoint is needed to track the cause (return/error code, register value...).
How to add this tracepoint ?

- In the API
- As a variation of an existing tracepoint, with extra/different parameters.

Problems

Timestamp generation & alteration

- Timestamp is calculated at generation time
- No direct access to the timestamp field
- How to change the timestamp is case of differed trace generation ?
  E.g. GPU, low level PM transitions with minimum HW support

OMAP clock sources

- OMAP uses 32KHz clock for tracing => 30us resolution
- Need a faster timer as kernel clock source
- Dynamic (auto) switch to the 32KHz when going to low power modes

Problems

Embedded world

- Trace control & dump are performed on the target while the parsing tools are running on the host.
- control tracing (enable/disable, filter),
- dump events trace (data & description)

```
# mount -t debugfs nodev /sys/kernel/debug/
# echo 1 > /sys/kernel/debug/tracing/events/power/enable
# cat /sys/kernel/debug/tracing/trace_pipe > /tmp/trace.txt
# cat /debug/tracing/events/power/power_domain_target/format
name: power_domain_target
ID: 63
format:
        field:unsigned short common_type;       offset:0;       size:2; signed:0;
        field:unsigned char common_flags;       offset:2;       size:1; signed:0;
        field:unsigned char common_preempt_count;       offset:3;       size:1; signed:0;
        field:int common_pid;   offset:4;       size:4; signed:1;
        field:int common_lock_depth;    offset:8;       size:4; signed:1;

        field:__data_loc char[] name;   offset:12;      size:4; signed:0;
        field:u64 state;        offset:16;      size:8; signed:0;
        field:u64 cpu_id;       offset:24;      size:8; signed:0;

print fmt: "%s state=%lu cpu_id=%lu", __get_str(name), (unsigned long)REC->state,
    (unsigned long)REC->cpu_id
```

Problems

Events format

- Format detection : from target debugfs
- Format flexibility : how to add new events or variations of events ?

Parsing tools

- Focus on non-embedded systems
- Importing events trace
- Display options

Solutions

Format

- Flexibility : variable number of args (à la 'int printk(const char *fmt, ...);')
- Mixed format : description and data in the event
  Example : from **'cpu_idle: state=3 cpu_id=0'** **to** **'cpu_idle: state(%lu)=3 cpu_id(%lu)=0'**
  => Allows the parsing of variable args events
- Dynamic filtering

Timestamps : TBDiscussed

OMAP clock sources : on-going, patches from TI

Parsing tools

- Run-time format detection
- Display format options : type of diagram, color, highlighting, field unit/radix ...
- Filtering & stats options
- Using profiles + load/store

Next steps

Discussions -> MLs
- linux-kernel, linux-perf-users
- linux-arm(-kernel), linux-omap

Parsing tools : pytimechart
+ timechart, other (new) tools

Contributors

- Trace code maintainers :
  Steven Rostedt <rostedt@goodmis.org>
  Frederic Weisbecker <fweisbec@gmail.com>
  Ingo Molnar <mingo@redhat.com>

- New trace API & tools : Thomas Renninger <trenn () suse ! de>
- pytimechart : Pierre Tardy <pierre.tardy () intel ! com>

Links

Omapedia wiki

[1] PM debug & profiling
http://www.omappedia.org/wiki/Power_Management_Debug_and_Profiling

PM devices latency measurements
http://www.omappedia.org/wiki/Power_Management_Device_Latencies_Measurement

Mainline patches

PM trace API
http://marc.info/?l=linux-kernel&m=129173937301616&w=2

New API doc & suspend tracepoint
http://marc.info/?l=linux-kernel&m=129425340005149&w=2

Introduced clock and power_domain events classes
http://marc.info/?l=linux-kernel&m=128471217521623&w=2

OMAP tracepoints : clocks, power domains, default idle handler
http://marc.info/?l=linux-omap&m=129805267301984&w=2

pytimechart

http://gitorious.org/pytimechart#more

Backup slides