Performance/Power
SoC/ACAP
in Automotive Linux

Jerry Wong
Senior Manager Platform Engineering
July 17, 2019
> Why Xilinx

> Software Implementation

> Futures
> Why Xilinx

> Software Implementation

> Futures
In a survey of over 2000 customers …

> 55% said Flexible I/O
> 43% said Hardware Acceleration of Software
> 31% said Performance/Power
> Many other reasons were given …

… #3 of the many reasons is Performance/Power …
Thomas at Daimler Research reasoned …

> Performance/Power was so important and Xilinx was definitely leading there
> Timeline (complete production/qualification setup)
> Flexible and easy to use software environment
> Great support (local on three continents)

Xilinx Automotive Link ref timestamp 7:37

… Performance/Power can be particularly important to automotive …
What for Automotive:

> Driver Sleep Detection: Eye Closure, Blink Frequency, Yawn, Head Angle, Fixed Gaze, Sleepiness …

> Neural network was custom trained for facial expressions, including sleepiness …

> Platform Zynq UltraScale+ MPSoC

… Machine learning for driver safety …
What for Automotive:

> Convolutional Neural Network allows machine learning by training rather than explicit coding.

> Hardware acceleration of Binary Neural Network goes from 2.17 “tiles” per second with four A53 cores to 15K “tiles” per second with hardware acceleration.

> 60 fps means real time performance …

> Platform Zynq UltraScale+ MPSoC (ZU3)

... Processor acceleration for real time Machine Learning …
What for Automotive:

- IP + Embedded Xilinx gives 2x Lidar range with the same analog hardware. Improves a key performance spec.
- Platform Zynq7000
What for Automotive:

> 8 simultaneous 30fps Autonomous Driving algorithms in 10W.

> Platform Zynq UltraScale+ MPSoC

... Performance/power for autonomous driving ...
What for Automotive:

> ~1s switching from parking algorithm to highway algorithm. Quickly reprogramming the hardware = Dynamic Function Exchange feature (DFX).

> Parking algorithm stitches four cameras with perspective correction. Neural network detection of open spaces.

> Highway algorithm detects lanes and vehicles.

> Platform Zynq7000

... Adaptive DFX and flexible I/O for increased feature count ...
Besides the Inherent Performance/Power …

> With several engines/OSs sharing peripherals, how do you know which peripherals are being used, and which can be turned off …

> The PMU knows the state of all, and automatically turns off the peripherals that are not used.

> The benefit is isolation between functions and code portability. This isolation is a safety-critical feature.

SDG (Software Developer’s Guide)

... Complex systems present unique Power Management problems …
Xilinx Safety Features

- Besides Isolation between APU and RPU cores …
- Redundant Processors: Dual R5, Triple-redundant PMU (laid out to different geometries).
- Redundant Memories: Memory and ECC physically separated (neutrino-proof)
- STL (Software Test Library): PMU runs diagnostics in the background.
- Data exchange between APU and RPU targeting ASIL B and higher, work being done.
- SW Houses targeting ASIL B for ADAS.

Xilinx Functional Safety

… Xilinx has robust Safety features …
Why Xilinx

Software Implementation

Futures
PMU Control

> There’s an EEMI API for:
  >> Wake source
  >> Informing a peripheral is in use
  >> Informing an engine is in use
  >> Clock control
  >> Informing a core is in use

EEMI = Embedded Energy Management Interface)

TRM (Technical Reference Manual)

… For non-Linux OSs, EEMI APIs provide explicit power management …
Linux PMU Overview

> Linux features are made from EEMI, so the features are functionally identical

> The Xilinx implementation makes Power Management “The magic that is largely transparent to the Linux user”

> Let’s see the beauty of the Linux Power Management implementation …

---

CPU PM
CPU Hotplug
Kernel documentation: /Documentation/power/thermal-power-hotplug.txt

The required kernel configuration options are:

- Kernel Features
- Support for hot-pluggable CPU
- Power management options
- CPU suspend support

The use cases are to manage CPU online and offline as needed via the CPU Hotplug control interface.

For example, make the CPU online:

```bash
$ echo 1 > /sys/devices/system/cpu/cpu0/online
```

Linux Power Management Wiki

… Solutions: Make it Beautifully Transparent or Integrated with Linux …
Informing a peripheral is in use

> The PMU is automatically informed that the peripheral is in use with the Linux driver “probe”. (All generic power domains execute the zynqmp handler.)

```c
zynqmp_gpd_probe
pd->gpd.attach_dev = zynqmp_gpd_attach_dev;
pd->gpd.detach_dev = zynqmp_gpd_detach_dev;
```

https://github.com/torvalds/linux/blob/master/drivers/soc/xilinx/zynqmp_pm_domains.c

... Registering peripheral use is automatic by Linux driver Probe ...
Informing an Engine is in use

> Request suspend on a CPU reaches a lower power than WFI. PMU can turn off CPU support circuitry when not in use. (Simple APIs to suspend and assign wake.)

echo mem > /sys/power/state
echo +10 > /sys/class/rtc/rtc0/wakealarm

SDG Software Developers Guide
Automatic Power Domain Switching

> If the Power Domain is fully not in use (no peripherals on those power rails needed), the PMU automatically powers the Domain down at the power rails. (All generic power domains execute the zynqmp handler to inform the PMU.)

```c
zynqmp_gpd_probe
  pd->gpd.power_off = zynqmp_gpd_power_off;
  pd->gpd.power_on = zynqmp_gpd_power_on;
```

https://github.com/torvalds/linux/blob/master/drivers/soc/xilinx/zynqmp_pm_domains.c

… Suspend / Resume via API … Domain Off / On automatic …
Informing a Core is in use

> CPU Hotplug implemented to Suspend a core. (PMU API singleton)

> CPU Idle implemented to automatically Suspend a core when not in use. (ISR suspend handler.)

```c
zynqmp_firmware_probe
eemi_ops_tbl = &eemi_ops;
https://github.com/torvalds/linux/blob/master/drivers/firmware/xilinx/zynqmp.c

zynqmp_pm_probe
eemi_ops = zynqmp_pm_get_eemi_ops
devm_request_threaded_irq
zynqmp_pm_isr
https://github.com/torvalds/linux/blob/master/drivers/soc/xilinx/zynqmp_power.c
```

... Core Off / On automatic ...

>> 19
Drivers implementing runtime power management (PM) automatically turn off when not in use. Xilinx PM seamlessly supports Linux PM. (Job reference=0 suspends/clock gates.)

Examples up to here, you may be getting the additional benefits of Linux Power Management without much effort … Beautiful …

dev_pm_ops mali_dev_ext_pm_ops
  .runtime_suspend = mali_driver_runtime_suspend
  mali_disable_clk
mali_scheduler_complete_gp_job
  _mali_osk_pm_dev_ref_put
mali_pm_record_gpu_idle

drivers/gpu/mali400/r4p0/mali/linux/mali_kernel_linux.c and common/mali_scheduler.c

… Drivers can implement real time power management since Xilinx PM seamlessly supports Linux PM …
MALI GPU Runtime PM Code …

The code is in a tar ball, so provided here in detail …

static const struct dev_pm_ops mali_dev_pm_ops = {
    #ifdef CONFIG_PM_RUNTIME
        .runtime_suspend = mali_driver_runtime_suspend,
        .runtime_resume = mali_driver_runtime_resume,
        .runtime_idle = mali_driver_runtime_idle,
    #endif
    .suspend = mali_driver_suspend_scheduler,
    .resume = mali_driver_resume_scheduler,
    .freeze = mali_driver_suspend_scheduler,
    .thaw = mali_driver_resume_scheduler,
    .poweroff = mali_driver_suspend_scheduler,
};

mali_platform_device_register()
{
    ...
    pm_runtime_set_autosuspend_delay(&mali_gpu_device.dev, 1000);
    pm_runtime_use_autosuspend(&mali_gpu_device.dev);
} #endif

pm_runtime_enable(&mali_gpu_device.dev);
...

mali_scheduler_queue_gp_job()
{
    ...
    _mali_osk_pm_dev_ref_get_async()
    ...
}

mali_scheduler_complete_gp_job()
{
    ...
    _mali_osk_pm_dev_ref_pet_async()
    ...
}

mali_pm_common_suspend()
{
    ...
    if (0 < num_groups_down) {
        mali_executor_group_power_down(groups_down, num_groups_down);
    }
    for (i = 0; i < num_l2_down; i++) {
        mali_l2_cache_power_down(l2_down[i]);
    }
    ...
}
Clock Control

> Get additional power savings by **dynamic clock scaling**. (Implements clock divider.)

> Control CPU frequency with the `cpufreq/scaling_setspeed` API. (More control.)

**Linux Power Management Wiki**

> Advanced: Turn On/Off PLLs, set PLL frequency, assign PLL topology directly with `sysfs`

```c
zynqmp_clk_divider_set_rate
eemi_ops = zynqmp_pm_get_eemi_ops
eemi_ops->clock_setdivider
```

https://github.com/torvalds/linux/blob/master/drivers/clk/zynqmp/divider.c

... Frequency scaling, and direct clock control …
Non-Linux concepts

> There are some Power Management tasks that have no counterpart in Linux (e.g. Controlling other Engines).

> All the EEMI APIs exposed in Linux through sysfs (debug, registers, GPIOs, etc.) When there is a Linux counterpart, it is recommended to use the Linux version so Linux is aware of State. (EEMI debugfs.)

```bash
$ echo pm_get_node_status <node> > (sysfs node)
zynqmp_pm_debugfs_api_write
    process_api_request
    eemi_ops->query_data
```

https://github.com/torvalds/linux/blob/master/drivers/firmware/xilinx/zynqmp-debug.c

... Direct control via sysfs ...
> Why Xilinx

> Software Implementation

> Futures
What is Xilinx Planning for the Future …

> Some change is necessary. Continue to make it transparent and compatible as possible between hardware generations

… Goal is make features transparent in Linux …
Support of Linux Power Management governors is the remaining feature being worked on. This can make clock control transparent as well. One more automated feature.
Handling Changes for Industry

> Changes for industry: New standards, new tools, emerging customer preferences, and other market driven requirements … Handled as they have been in the past, with consideration of existing users …

… Some evolutionary changes are necessary, support plan considers existing users …
Xen Fast Boot

> Dom0-less System boots multiple VMs simultaneously.

... Xen boot time improvement ...

Xilinx Xen
Continue Easy Support Strategy of Linux Power Management Features

> As new Linux Power Management features are added, Xilinx will continue to implement them as transparent and compatible as possible.
Experimentation Environment

- Jupyter Notebook / Python for code experimentation.

```
In [1]: from poweradvantage import poweradvantage
pa = poweradvantage("VCK198", "SC")
pa.help()
p = pa.printpower()
```

Volts  Amps  Watts
MGTYAVCC  0.880  0.5050  0.4444
MGTYAVTT  1.199  0.0175  0.0210
MGTYAVCCALUX  1.502  0.0000  0.0000
Transceiver Domain  -----  -------  0.4654
VCC1V8  1.799  0.0015  0.0027
VCC3V3  3.311  0.0055  0.0162
VCC1V2  1.200  0.0000  0.0720
VCC1V1  1.100  0.0310  0.0341
PL Domain  -----  -------  0.1270
VCC_RAM  0.080  0.0800  0.0000
VCCAUX  1.500  0.0105  0.0158
System Domain  -----  -------  0.0158
VCC_PSFP  0.780  0.0245  0.0191
FPD  -----  -------  0.0191
VCC_PSLP  0.777  0.0140  0.0189
VCC0_502  1.888  0.0020  0.0036
LPD  -----  -------  0.0145
VCC_PMC  0.780  0.0155  0.0121
VCC0_308  0.480  0.0032  0.0032

... Enables functional prototype ... Increase ease-of-use ...
ACAP


> PMU (Platform Management Unit) will be replaced by the PMC (Platform Management Controller).

> Power Management APIs will be ported to the new system.

Versal ACAP Link

… ACAP is new, it has a new kind of AI Engine and Network On Chip (NOC) …
AI Engine

> The AI Engine (AIE), solves the data starvation of the Vector Cores. There is a local RAM with two ports between each adjacent pair of Vector Cores.

> AI Engine (AIE) gives 6x-10x Performance/Power. Entire AI Domain can be switched off when not in use.

Al Engine Link
Conclusion

> Xilinx can be a Performance/Power solution for Automotive.

> Linux Power Management is transparent and integrated as possible.

> Implementation of new features keeps existing users in mind.

… Ask if Xilinx may be right for your automotive applications …
Adaptable.
Intelligent.