SFO15-109: SoC / Cluster Idle

Presenters
Ulf Hansson, Kevin Hilman, Lina Iyer
One idle to rule them all?

Idle management of CPUs & IO devices

Kevin Hilman, Linaro
khilman@kernel.org
Two separate worlds

**CPUs**
- CPU idle framework
- `cpu_[cluster_]pm_*()`
- not scaling well for SMP or multi-cluster (c.f. coupled idle states)

**IO devices**
- Runtime PM
- auto-suspend
- PM domains
- generic PM domains (genpd)
One idle to rule them all

What if…

● use runtime PM for CPUs
● and CPU-connected “stuff”
  ○ interrupt controllers (ARM GIC)
  ○ floating-point units
  ○ CPU-local cache (L1$)

● model clusters with genpd
  ○ CPUs are just “devices” in the genpd
  ○ genpd includes shared resources (e.g. L2$)
SoC/Cluster Idle
SoC/Cluster Idle - What we need

- Hierarchy and nesting
- Last-man reference counting
- Platform Callbacks
- CPUIdle and Hotplug
- Relationship in DT
SoC/Cluster Idle - Solution

- Use genpd and Runtime PM
- Describe CPUs and domains in DT
  - CPU: `#power-domains = <&pd>;`
  - `power-controller: #power-domain;`
- Initialize genpd PM domains
- Attach CPU devices to genpd
- Add Runtime PM support for CPUIdle and Hotplug
- Provide platform callbacks
Shortcomings

- Mutexes used in genpd
  - Solution: Optionally, spinlocks for domains
- RCU called from idle CPU
  - RCU_NONIDLE()
Addition to genpd rules

- No change for non-IRQ Safe Domains
- IRQ Safe Domain
  - Contain only IRQ Safe devices & domains
- Can you combine them (subdomains)?
Ran into problems, heh!

- RCU stalls observed
  - Unresolved
- Runtime PM in Hotplug
  - Use Hotplug notifiers for a generic solution
- Performance: Heavy weight for CPUIdle?
- -RT kernel compatibility
Heavy weight for CPU idle?

- Additional latency*
  - Last CPU Idle enter: 50 - 70 us
  - First CPU Idle exit: 73 - 90 us

- Trim fat?

* Initial attempts on a 800 Mhz quad core ARM CPU
-RT compatibility

- Spinlocks may sleep in -RT kernel
  - raw_ spinlocks can spin on -RT

- CPUs can runtime suspend/resume only their devices
  - Do we need to even lock?
- Lockless Runtime PM, perhaps?
Next steps, discussion

- **genpd evolution**
  - locking simplification (Ulf Hansson, merged)
  - removing intermediate states (Ulf Hansson, merged)
  - CPU PM domains, IRQ-safe genpd support (Lina Iyer, posted)
  - etc.

- **CPU PM notifiers:** `cpu_[cluster_]_pm_[enter|exit]()`
  - used for IRQ chips, floating-point units, PMUs, wakeups, etc.
  - can/should we use runtime PM instead? (runtime PM callbacks instead of notifiers)

- **pm_genpdAttachCpuIdle()**
  - no more users? kill it. (Ulf Hansson, posted)
Next steps, discussion (cont)...

- genpd: needs to support multiple levels
  - currently only supports on/off
  - CPU/clusters have more levels (e.g. retention, C-states)
  - IO devices (D-states)
  - RFC by Axel Haslam (BayLibre)

- ACPI 6: low-power idle (LPI)
  - supports hierarchical idle
  - seems to map better to genpd than CPUIdle (c.f. Fig 8-46, 6.0 spec)

- PSCI: OS-Initiated composite StateID