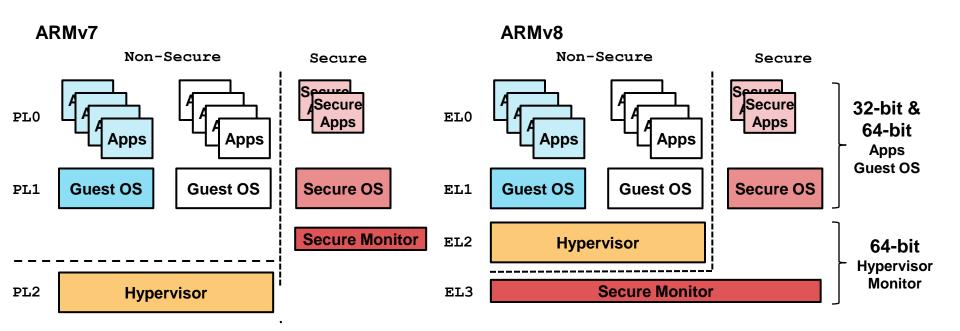
Power State Coordination Interface Update and what next





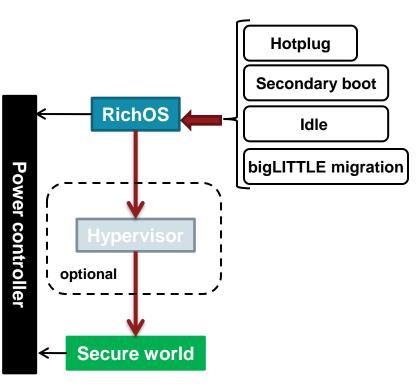
Multiple OS coexisting in one device



- Semico and Trusted OS vendors will supply secure code
- You might have a hypervisor
- Rich OS running at kernel privilege (eg Linux or windows)
- Applications at user level



Power Management Requires Coordination



 Need OSs at different privilege layers to collaborate on power

 OSs need to know when to save/restore their state

 OSs need to manage peripherals they own when entering a power state

Conduit to next layer of privilege

- To talk across OSs you need instructions that can get you up to next level of privilege
- SMC provides such an interface
- SMC can also be trapped at hypervisor level
 - A virtualised guest can use the same interface
- If EL3 is not implemented HVC can be used as conduit to support PSCI compliant guests

Power State Coordination Interface

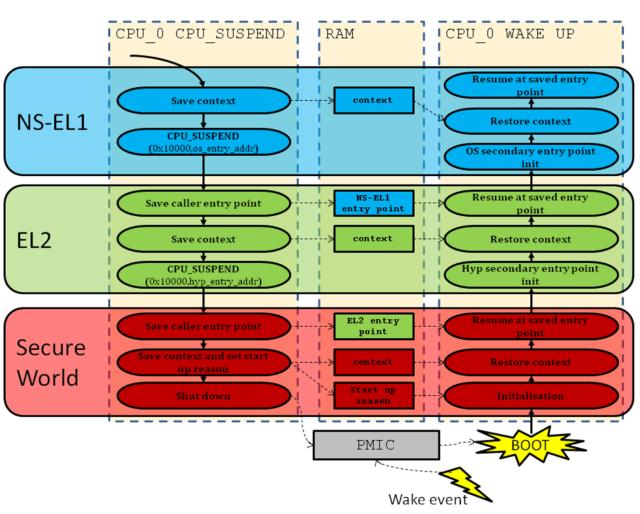
- System APIs are required by
 - Idle Management
 - Hot plug
 - Secondary boot
 - CPU Migration
- Can be captured in a few APIs. Principle APIs are:
 - CPU_SUSPEND
 - CPU_OFF
 - CPU_ON



CPU_SUSPEND

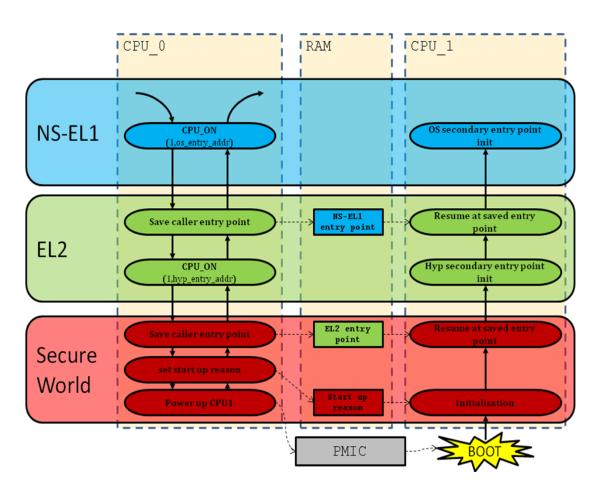
Parameters:

- State type:
 - Tells if you need to save context
- Affinity Level:
 - You can power down up to:
 - just this cpu
 - just this cluster
 - whole system
- Platform specific state ID
- Entry point address:
 - Resume here please



CPU_ON

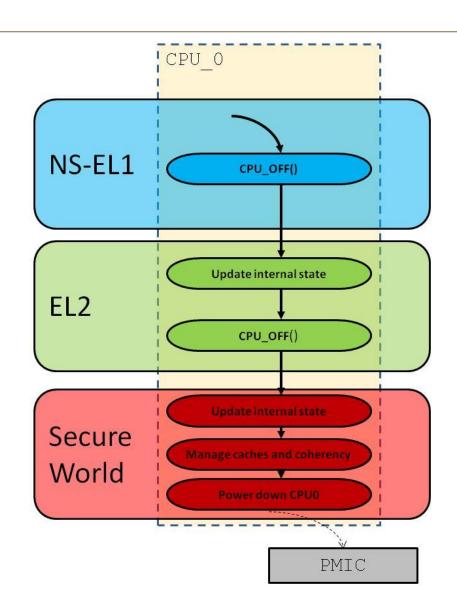
- Parameters:
- Target CPU:
 - MPIDR of target CPU
- Entry point address:
 - Resume here please
- Can only fail if
 - Thermal/reliability issues
 - FW has a the CPU ON pending
 - FW sees CPU as still being ON





CPU_OFF

- Called by CPU turning off. Takes no parameters
- Can fail if secure world cannot turn off CPU because a trusted OS is resident there
- MIGRATE moves Trusted OS
- MIGRATE_STATUS allows determination of Trusted OS presence. Returns:
 - There is no trusted OS, trusted OS does not require migration
 - Trusted OS is UP, does not support migration and its on CPUX (you can't turn X off)
 - Trusted OS is UP, supports migration and its on CPU X. Don't turn off CPU X (unless it's last one)



PSCI Discovery

Presence or absence of PSCI should be discoverable

- FW tables can present:
 - Version
 - Method

SMC, HVC

Call offsets

PSCI-next

- New version of the spec in the works.
- Spec to appear beginning of Q2
- Key deltas:
 - Clarified reset state and starting point for CPU_ON/CPU_SUSPEND
 - Clarifications of responsibilities for callers and implementers
 - Clarifications on CPU_ON/OFF races



PSCI-next

- Key deltas continued:
 - MIGRATE_STATUS: use at boot time to understand migration and hot-plug related requirements of a Trusted OS, where it is, and if it is
 - AFFINITY_STATUS: tells you if a CPU is on or off, or if a cluster has CPUs on or off
 - Considering RESET

PSCI-next

- PSCI on it's own doesn't buy that much
- PSCI is a step towards standardising power management
- We are now looking at :
 - Idle management
 - Many core boot
 - Hot plug



- What could you represent in FDT (per state):
 - Book keeping stuff
 - Latency information
 - Save/Restore requirements
 - Cache management requirements
 - Affinity level coordination (AKA last man standing requirements)
 - Wake up sources
 - State cookie or flags
 - Method of state entry



- What could you represent in FDT (per state):
- Book keeping stuff
 - Name of state
 - Description
 - Approximate power consumption
 - Latency information
 - Save/Restore requirements
 - Cache management requirements
 - Affinity level coordination (AKA last man standing requirements)
 - Wake up sources
 - State cookie or flags
 - Method of state entry



- What could you represent in FDT (per state):
 - Book keeping stuff
- Latency information
 - Entry/exit latencies (do we need both? Aggregate is probably fine)
 - Break even time for state (AKA target residency)
 - Save/Restore requirements
 - Cache management requirements
 - Affinity level coordination (AKA last man standing requirements)
 - Wake up sources
 - State cookie or flags
 - Method of state entry



- What could you represent in FDT (per state):
 - Book keeping stuff
 - Latency information
- Save/Restore requirements
 - Does state loose context?
 - CPU?
 - GIC?
 - Arch Timer?
 - Cache management requirements
 - Affinity level coordination (AKA last man standing requirements)
 - Wake up sources
 - State cookie or flags
 - Method of state entry



- What could you represent in FDT (per state):
 - Book keeping stuff
 - Latency information
 - Save/Restore requirements
- Cache management requirements
 - Do I need to cache clean? How far (L1 or L1/L2)?
 - Affinity level coordination (AKA last man standing requirements)
 - Wake up sources
 - State cookie or flags
 - Method of state entry



- What could you represent in FDT (per state):
 - Book keeping stuff
 - Latency information
 - Save/Restore requirements
 - Cache management requirements
- Affinity level coordination (AKA last man standing requirements)
 - Does the state require a set of cores to power down together?
 - If so does the OS need to coordinate entry into state?
 - That is implement something like coupled idle? And last core asks for state or
 - Or does the method of entry abstract this? And all cores ask for state

. . . .



- What could you represent in FDT (per state):
 - Book keeping stuff
 - Latency information
 - Save/Restore requirements
 - Cache management requirements
 - Affinity level coordination (AKA last man standing requirements)
- Wake up sources
 - At least one wake up source must be enabled
 - State cookie or flags
 - Method of state entry



- What could you represent in FDT (per state):
 - Book keeping stuff
 - Latency information
 - Save/Restore requirements
 - Cache management requirements
 - Affinity level coordination (AKA last man standing requirements)
 - Wake up sources
- Method of state entry:
 - something that in an abstract way says how you enter power states

```
{...
method = "psci" // implies you support PSCI and that there is a PCSCI
  entry in FDT
};
```

- In principle could have other methods
- State cookie or flags: for PSCI it could be parameter for call



- PSCI does not require:
 - Cache management to be expressed:
 - Implementation does it for you
 - Affinity level coordination is platform coordinated
 - No need for coupled C-states at OS level



ACPI and PSCI for Idle Management

- ACPI due to it's history has a number of areas where it doesn't particularly well with ARM
- Work is still required to make ACPI a good fit for ARM
- Examples are:
 - Expressing save/restore requirements
 - Expressing cache management requirements
 - Can do coupled states, but it's not very easy
 - Not as rich in latency expression: (no target residency)
- But ACPI does has some flexibility in description of C-states that can accommodate PSCI
- Following are ideas on how idle ACPI management can be hooked to PSCI



Ideas for ACPI and PSCI in Idle

- In ACPI Each C-state is associated with a register at an "address" in an "address space"
- One address space is Function Fixed Hardware (FFH)
 - "interpret address in a way defined by the CPU vendor"
 - For Intel FFH means MWAIT. With ARM we can use PSCI

Field	Size (bytes)	Core state
Address Space ID	1	0x7f (FFH)
Access Size	1	4 (64 bit)
Address		Flags: 32-63 PSCI param:0-31

- Address broken into :
 - Flags to indicate context to save restore
 - PSCI parameter for PSCI call



Ideas for ACPI and PSCI in Idle

- Example _CST table
 - Example : simple SMP system with core and cluster gating

Description	Location	Address
C1 WFI	N/A	N/A
C2 Core OFF	OFF	0x10:0x10000 (AFF0 power down)
C3 Cluster Off	OFF	0x10:0x101000 (AFF1, power down)

ACPI idle driver needs to say #ifdef ARM

```
If state == C1 WFI
```

Else if register address is FFH:

save state as appropriate(state_address_upper_32)

PSCI_CALL(state_address_lower_32)



PSCI

- Provides the method of abstracting key power management requests across exception/privilege levels
- Provides a generic way of handling hot plug and secondary core boot
- Provides support for Migration (good for big.LITTLE)
- Can be tied to FDT and ACPI, to help provide generic idle management

Questions

