ACPI-next C-States

Charles Garcia-Tobin

Oct 2013

ACPI-Next C-states

- Numerical non-equivalency – Types of states
- Topology awareness
- Additional Information: Version, BreakEven, S/R, Cache
- Device, Power Resource, and Interrupt dependencies
- PSCI

Standardise types of State for ARM

- ARM Linux community uses ACPI derived terminology:
  - C-State, P-state
- But it's not done in a very standard way. Some "general rules"
  - C1 tend to be WFI
  - Larger numbers mean deeper states
- Hard to compare systems
- ACPI has strong definitions for C0-C3 states
  - Allows more states, but they are of type C1,C2 or C3
  - But are not defined in ARM terms

Types of State

Run

Hierarchy is running fully operational

Idle

Hierarchy is not executing, but has preserved all context, will wake up through an interrupt

ZZz

Sleep

Hierarchy is not executing, context is lost and must be saved/restored, will wake up through an interrupt. Caches in the hierarchy are off

RIP

Off

Hierarchy is not executing, context is lost and wake up will only occur by an explicit software command or By physical hardware reset

Types of State

Run
Hierarchy is running fully operational

C0

Idle
Hierarchy is not executing, but has preserved all context, will wake up through an interrupt

C1

Sleep
Hierarchy is not executing, context is lost and must be saved/restored, will wake up through an interrupt
Caches in the hierarchy are off

C3

Off
Hierarchy is not executing, context is lost and wake up will only occur by an explicit software command or By physical hardware reset

Zzz

RIP

Types of State

- Propose updating language for C1,C2,C3 so that for ARM based systems:
- C1 is state (WFI) and also a type of state Idle
- C2 not used on ARM systems (as type of state)
- C3 as a type of state has the semantics of Sleep

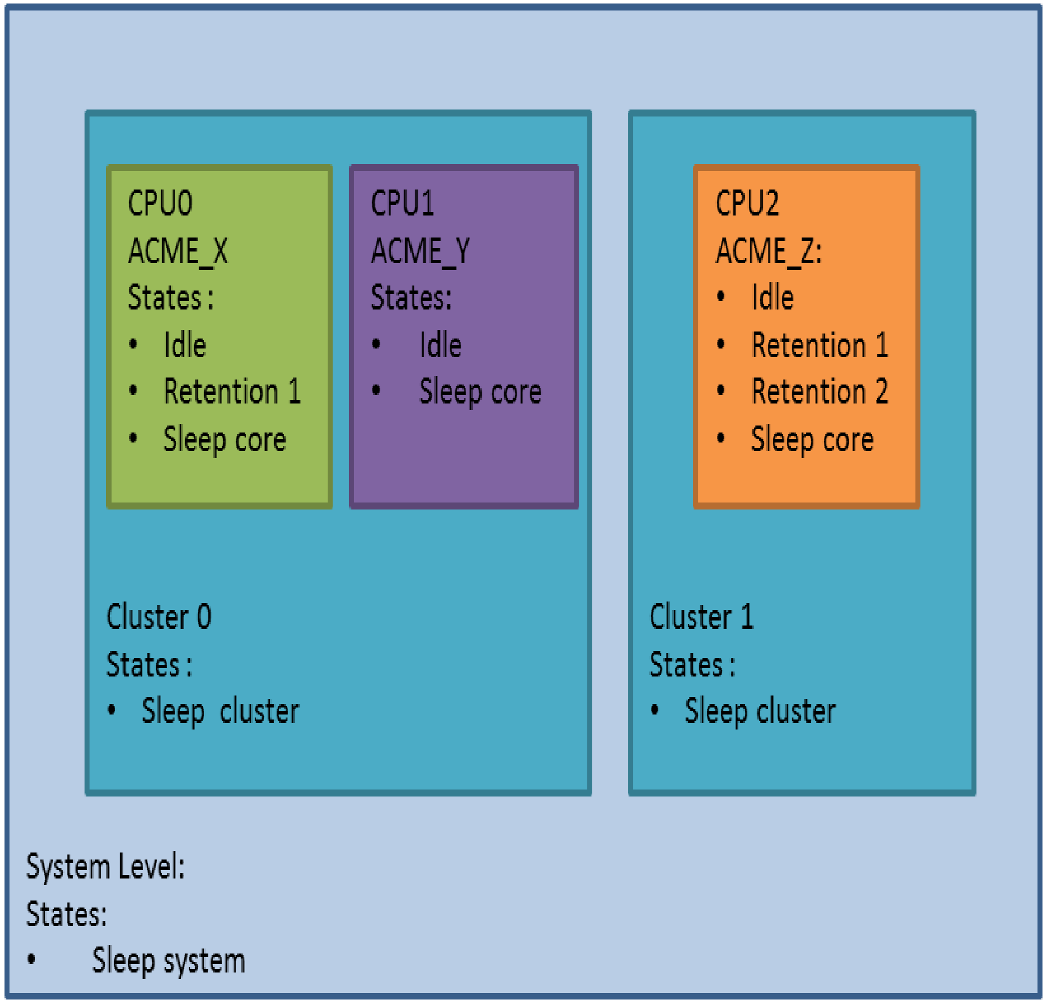
Topology

- ACPI uses _CSD to express coupled C-states
- _CSD relies on the notion of symmetry : Same states for all processors
- This doesn’t bode well for heterogeneous systems that might have different number of states per processor
- Feedback from OSVs, is that _CSD is not a very nice object to work with in kernel

Topology

- ACPI has a tree like name space
- We can use that to represent topology directly
- Modules : Structure that can contain other modules or devices
- We can declare individual CPU states
- We can declare cluster/system states
  - Can represent any arbitrary topology

Topology



```
Module System {
 _CST(Sleep System)
 Module Cluster 0 {
  _CST(Sleep Cluster)
   Processor CPU0 {
     _CST(Idle,
          Retention1,Sleep)
   }
   Processor CPU1 {
     _CST(Idle,Sleep)
   }
 }
Module Cluster 1 {
  _CST(Sleep Cluster)
   Processor CPU2 {
     _CST(Idle,
          Retention1,
          Retention2,
         Sleep)
   }
 }}
```

Additional Information

- _CST doesn't have enough information for an OSPM to manage an ARM system's power states
  - No representation of what context may be lost
  - No representation of what caches may be lost/if they need management or not
  - Last core?
  - Break Even Latency missing
  - No easy way to link C-states with device states or power resource states
  - No easy way to describe wake capability of an interrupt
    - Not every interrupt can wake you from every state
  - No versioning

Additional Information

- Proposal is to replace _CST with new object method _CSX

```
Package {
      Version, // Integer (WORD)
      Count, // Integer (WORD)
      CStatesX[0],  // Package
      …
      CStatesX[N-1]  // Package
      }
```

- Version is self explanatory. It would start at 0

Additional Information

- CStateX

```
Package(9) {
        Register // Buffer (Resource Descriptor)
        Type // Integer (BYTE)
        Latency // Integer (WORD)
        Power // Integer (DWORD)
        BreakEven // Integer (DWORD)
        GenStateFlags // Integer (WORD)
        ArchStateFlags // Integer (DWORD)
        PowerDepth, // Integer (BYTE)
        LastMan, // Reference
}
```

Additional Information

- BreakEven:
  - How long do I have to be in this state compared to WFI to save power
- GenStateFlags:
  - OSPM_MANAGE_CACHE:
    - Set if OSPM has to manage caches,
  - CACHES_CONTENTS_LOST
    - Set if caches up to current topology level are lost with this power state
    - (More on this later)
  - PLATFORM_COORDINATED:
    - If set, platform coordinates and does last man standing.
      - If state is requested it will be entered when last CPU requests that state or deeper
    - If clear if state is requested it will be entered. OSPM has to coordinate
- ArchStateFlags:
  - Architecture specific flags. For ARM: what context needs to be saved/restored
  - Generic Registers, VFP, Debug context, Timer, CPU Interface, GICD

Additional Information

- `PowerDepth`:
  - Identifier describing how deep a power state is
  - Bigger number -> Bigger power savings -> Bigger latencies
  - Defaults to 0

- Used to describe dependencies
  - PowerResources: if ON States of depth X or higher are not available
    - Requires augmenting PowerResource Definition
  - Device state D0, D1, D2, D3hot then:
    - States of depth X=FunctionOfDState or higher are not available
    - Requires new methods
  - IRQ X can wake up power states of PowerDepth Y or lower
    - Required augmenting IRQ descriptions

Additional Information

- `LastMan`:
  - Allows for cases where only one specific CPU can enter a coupled State
- If Plaform Coordinated (PLATFORM_COORDINATED flag set) this should be undefined
- If CPU only power state this should be undefined
- If it is a coupled C state:
  - If state can be initiated on any CPU (doesn't matter who last man is) it must be undefined
  - Otherwise reference to processor node that must initiate the power state transition

Cache

- Propose that each level of topology has a Cache object, _CLV

- Describes Caches that are private to that part of the topology

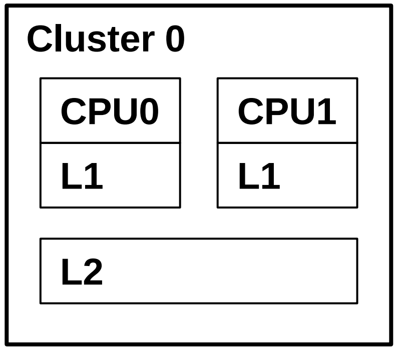
- Simplest instance could be simply a list of caches:

```
_CLV  return package of integers
eg:   Core has L1 cache = {L1}
      Cluster has L2 cache = {L2}
```

- Combines with flags, CACHES_CONTENTS_LOST and OSPM_MANAGE_CACHE

Cache

- When not all CPUs have entered a coupled state:
  - Caches in sub-hierarchies will be lost
  - CPUx: L1

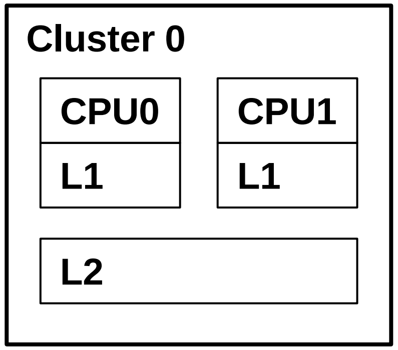


```
Module Cluster 0 {
  _CSX(…
         CStateX : CACHES_CONTENTS_LOST
         …)
  _CLV( L2 )

  Processor CPU0 {
     CSX(…)
     _CLV( L1)
   }
   Processor CPU0 {
     CSX(…)
     _CLV( L1)
   }}
```

Cache

- When not all CPUs have entered a coupled state:
  - Caches in sub-hierarchies will be lost
  - CPUx: L1



```
Module Cluster 0 {
  _CSX(…
        CStateX : CACHES_CONTENTS_LOST
        …)
  _CLV( L2 )

  Processor CPU0 {
     CSX(…)
    _CLV( L1)
   }
   Processor CPU0 {
     CSX(…)
    _CLV( L1)
   }}
```

- When last CPU enters cluster state:
  - All caches up to topology level are lost
    - L2, all L1s

PSCI

- Two flags in FADT Flags field:

PSCI_PRESENT: 1 if PSCI is implemented

USE_HVC: 1 if HVC is should be used in preference to SMC as PSCI conduit

- Note no override or function IDs

PSCI

- CStateX uses Register to denote how to enter a C-state
- Registers represented GAS: Generic Address Structure
- GAS is very flexible and allows encapsulating “Software as Hardware” in very specific cirmcunstance. This is called Function Fixed Hardware
- Power State Parameter of CPU_SUSPEND is the Registers Address
  - Equivalent to method used for Intel today

PSCI

- GAS Register format for a C-State

Field	Description
Address Space ID	0x7f (FFH identifier)
Register Bit Width	32
Register Bit Offset	0
Access Size	3 (DWORD)
Address	Upper DWORD must be zero Lower DWORD must denote the PSCI power_state parameter for the CPU_SUSPEND call Address: 0xF000000_00000000 is reserved for WFI this would not result in a PSCI call

Questions

???