Enabling Collaborative Processor Performance Control (CPPC) on Arm Platforms

Pranav Madhu
Software Engineer, Arm Ltd.
24-March-2021
Contents

● What is CPPC?
● Overview of Activity Monitor Unit (AMU).
● AMU registers as ACPI FFH.
● Overview of SCMI Fastchannel.
Introduction to CPPC

- A mechanism for the OS to manage the performance of the processor core on a contiguous and abstract performance scale.

Diagram:
- OS → Perf change req → Platform Firmware
- Update CPU frequency and voltage if requested perf level is supported

Graph:
- Guaranteed Performance Allowed Range
  - Highest Performance
  - Nominal Performance
  - Lowest Nonlinear Performance
  - Lowest Performance
CPPC for ARM Platform

- Consist of two part:
  - Monitoring Processor Performance
  - Controlling Processor Performance

CPPC OS Framework

- Read AMU register values to obtain CPU perf (operating freq)
- AMU registers exposed to OS as ACPI FixedHW

Performance Monitoring

- Request desired perf.

Performance controlling

- FastChannel

DVFS framework update Freq/Voltage if requested perf level is supported.

SCMI framework

Platform Firmware
Activity Monitor Unit (AMU)

- Optional extension to the Armv8.4 architecture.
- Consists of a group of 64-bit event counters.
- Events counted by the architected event counter are fixed and architecturally defined.
  - AMUv1 supports four architected event counters
- Event counted for each auxiliary event counter may be fixed or programmable (implementation defined).
  - AMUv1 supports sixteen auxiliary event counters
- Architected event counters used in performance monitoring:
  - Processor frequency counter
  - Constant frequency counter
Monitoring Processor Performance

- Processor frequency counter counts the clock supplied to the processor.
  - In other words, processor frequency counter increments based on processor clock.
- The constant frequency counter increments at a constant frequency.
  - Equal to the rate of increment of the System counter.
- These counters does not increment when the Core is in WFI or WFE

Note: Processor frequency counter is also referred as delivered performance counter. And Constant frequency counter is also referred as reference performance counter.
AMU registers as FFixedHW

- The AMU registers are implemented as system registers and are not memory mapped.
- ACPI FFixedHW interface is the recommended method for accessing such platform specific features.

**ACPI _CPC FFixedHW Registers**

```c
... ResourceTemplate () { Register (FFixedHW, 64, 0, 1, 4) }, // Reference Performance Counter Register ResourceTemplate () { Register (FFixedHW, 64, 0, 0, 4) }, // Delivered Performance Counter Register ...

// Acpi Generic Register Resource Descriptor Macro: // Register (AddressSpaceKeyword, RegisterBitWidth, RegisterBitOffset, RegisterAddress, AccessSize, DescriptorName)
```
SCMI FastChannel

- The OS requests the desired performance to the platform over the SCMI fastchannel.
- Fastchannel is a memory mapped region shared between the application processor and the platform firmware.
- There should be one fastchannel per core, for controlling the performance independently.
- A single fastchannel supports four protocols.
  - Performance limits set
  - Performance limits get
  - Performance level set
  - Performance level get
- The *performance level set* protocol (memory mapped address) is used by the OS to request desired performance.
- More information on fastchannel are available in the Arm [SCMI Specification](#).
Controlling the performance

Configure timer to generate event periodically for scmi perf driver on boot

Scmi perf module read fastchannel on timer event

Value Updated?

Is Perf value supported?

Wait for next Timer event

False

True

False

True

DVFS module update CPU frequency

Request performance level
ACPI _CPC Control method

● Platform should specify the following fields of the _CPC package:
  ○ Highest performance
  ○ Nominal performance
  ○ Lowest nonlinear performance
  ○ Lowest performance
  ○ Reference performance
  ○ Lowest frequency
  ○ Nominal Frequency
  ○ Delivered performance counter register
  ○ Reference performance counter register
  ○ Performance level set register

● Platform must use the same performance scale for all processors in the system.
  ○ To make sure any two processors running the same workload at the same performance level will complete in approximately the same time.
ACPI _CPC Control method (Cont...)

- Performance scale granularity is chosen such that reference performance is an integer.
- ACPI _PSD (P-State dependency) method should also be declared under the processor object.
  - Example _CPC control method, with performance scale granularity of 20MHz (lowest freq / lowest perf = 20MHz)

```
Name(_CPC, Package) {
  23, /* NumEntries */
  3, /* Revision */
  130, /* Highest Performance */
  130, /* Nominal Performance */
  65, /* Lowest Nonlinear Performance */
  65, /* Lowest Performance */
  ResourceTemplate () { Register (SystemMemory, 0, 0, 0, 0), /* Guaranteed Performance Register */
    ResourceTemplate () { Register (SystemMemory, 0, 0, 0, 0), /* Minimum Performance Register */
    ResourceTemplate () { Register (SystemMemory, 0, 0, 0, 0), /* Maximum Performance Register */
    ResourceTemplate () { Register (SystemMemory, 0, 0, 0, 0), /* Performance Reduction Tolerance Register */
    ResourceTemplate () { Register (SystemMemory, 0, 0, 0, 0), /* Time Window Register */
    ResourceTemplate () { Register (FixedPoint, 64, 0, 1, 4), /* Counter Wraparound Time */
    ResourceTemplate () { Register (FixedPoint, 64, 0, 0, 4), /* Reference Performance Counter Register */
    ResourceTemplate () { Register (SystemMemory, 32, 0, 0x00000000, 3), /* Performance Limited Register */
    ResourceTemplate () { Register (SystemMemory, 32, 0, 0x00000000, 3), /* CPPC Enable Register */
    ResourceTemplate () { Register (SystemMemory, 0, 0, 0, 0), /* Autonomous Selection Enable Register */
    ResourceTemplate () { Register (SystemMemory, 0, 0, 0, 0), /* Autonomous Activity Window Register */
    ResourceTemplate () { Register (SystemMemory, 0, 0, 0, 0), /* Energy Performance Preference Register */
    5, /* Reference Performance */
    1300, /* Lowest Frequency in Mhz */
    2000, /* Nominal Frequency in Mhz */
  })
}
An example of DVFS Configuration in Platform

- The DVFS configuration should have voltage/frequency mapping for performance level.
- The Performance level to frequency mapping in ACPI should match with the SCP firmware DVFS configuration.

```
static struct mod_dvfs_opp opps[] = {
    { .level = 65UL, .frequency = 1313 * FWK_KHZ, .voltage = 800 },
    { .level = 75UL, .frequency = 1531 * FWK_KHZ, .voltage = 850 },
    { .level = 85UL, .frequency = 1750 * FWK_KHZ, .voltage = 900 },
    { .level = 105UL, .frequency = 2100 * FWK_KHZ, .voltage = 950 },
    { .level = 130UL, .frequency = 2600 * FWK_KHZ, .voltage = 1000 },
    { 0 }
};
```
Software Components for Neoverse Reference Design

● **Boot Time**
  ○ Platform Firmware:
    ■ Initialize PSU, DVFS & SCMI drivers.
    ■ Configure timer to generating alarm for polling fastchannel at regular interval.
  ○ Arm TF:
    ■ Enable AMU, and make AMU register accessible to Non-Secure OS.

● **Run Time**
  ○ Platform Firmware:
    ■ Scmi_perf driver polls the fastchannel periodically based on the alarm generated by the timer module and generates event to DVFS.
  ○ UEFI:
    ■ ACPI '_CPC' control method to provide fastchannel address
    ■ ACPI '_CPC' control method to expose AMU registers as FFixedHW to the OS.
  ○ Linux:
    ■ Governor and CPPC driver evaluates _CPC control method and monitor/scale processor performance.
Governors and sysfs entry

- Linux by default boot with schedutil governor
- On boot, kernel request platform to switch to nominal performance, as there is more load.
- The sysfs entry for CPPC is at /sys/devices/system/cpu/cpufreq/policy_x/ (x can be 0, 1, 2, ...)
- From sysfs, it is possible for user to change the governor (supported governors are available from the scaling_available_governors entry)
- Using userspace governor, the user can change the operating frequency by writing the desired frequency into scaling_setspeed.
Reference

Thank you

Accelerating deployment in the Arm Ecosystem