Intel Power Management & MSR_SAFE

Energy Efficiency in HPC, IT4Innovation, Ostrava (CZ), 29 January 2020

Dr. Daniele Cesarini, HPC Software Engineer, CINECA
Dr. Robert Schöne, Faculty Computer Science, TU Dresden

And God said, Let there be light: and there was light.
And God saw the light, and it was good; and God divided the light from the darkness.

Maxwell’s Equations

$$\nabla \cdot \mathbf{E} = \frac{\rho_v}{\varepsilon} \qquad \text{(Gauss' Law)}$$

$$\nabla \cdot \mathbf{H} = 0 \qquad \text{(Gauss' Law for Magnetism)}$$

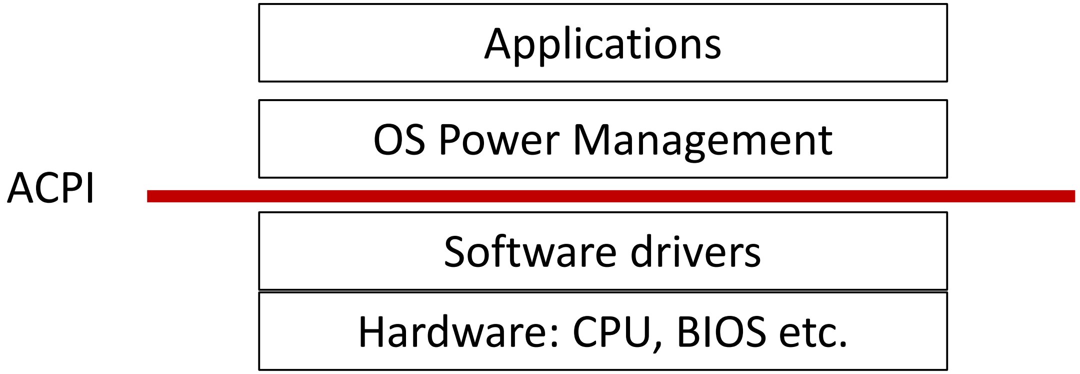
$$\nabla \times \mathbf{E} = -\mu \frac{\partial \mathbf{H}}{\partial t} \qquad \text{(Faraday's Law)}$$

$$\nabla \times \mathbf{H} = \mathbf{J} + \varepsilon \frac{\partial \mathbf{E}}{\partial t} \qquad \text{(Ampere's Law)}$$

Third verse of the Book of Genesis

What is ACPI?

ACPI (Advanced Configuration and Power Interface) is an open industry specification co-developed by Hewlett-Packard, Intel, Microsoft, Phoenix, and Toshiba. ACPI establishes industry-standard interfaces enabling OS-directed configuration, power management, and thermal management of mobile, desktop, and server platforms.



Global system states (g-state)

G0 : Working (e.g. running and idle)

We will focus on G0 states of Intel architecture!

G1 : Sleeping (e.g., suspend, hibernate)

G2 : Soft off (e.g., powered down but can be restarted by interrupts from input devices)

G3 : Mechanical off

Lower number means higher power

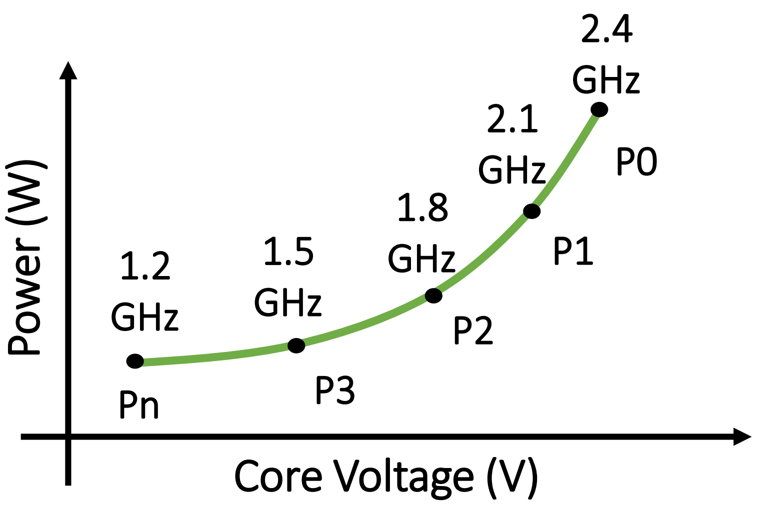
CPUs implement mechanisms to support these states.

BIOS and software drivers hide the difference of CPU implementations to support the ACPI defined data structures and functions.

Enhanced Intel SpeedStep Technology
EIST (aka Software Intel P-states)

ACPI defines performance states (P-States)

- P-States correspond to different performance levels that are applied while the processor is actively executing instructions.
- P-state is frequency and voltage operating point, both are scaled as the P-State increases.
- The driver provides an interface to control the P-State selection for the SandyBridge+ Intel processors.
- P-state takes advantages of turbo boost to increase the performance (core frequency higher than nominal).



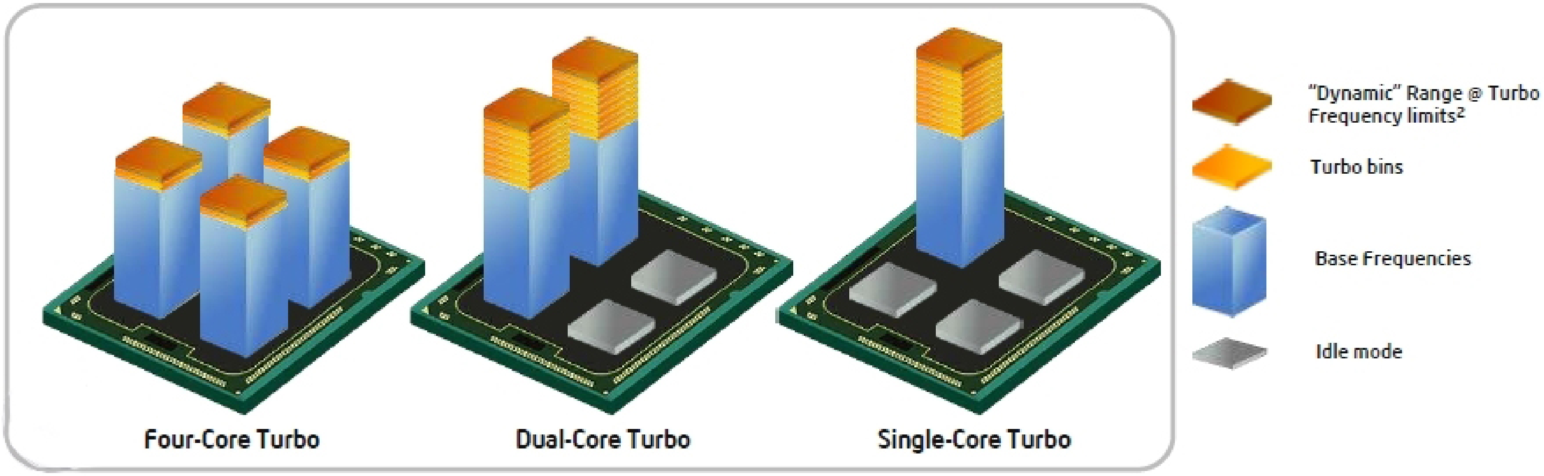
- Cores which share the same voltage domain vote for a p-state
- The highest p-state for each core wins

- Voltage regulator per core
- Don’t need to vote for a common-state
- Each core has its own P-State

Turbo Boost Technology

Turbo Boost is an Intel technology that opportunistically allows the processor to run faster than the nominal frequency if the CPU is operating below power, temperature and current limits. Turbo Boost is not overclocking!

- Operates under OS control – only entered when OS requests higher performance state (P0).
- Max Turbo Boost frequency is dependent on the number of active cores and varies by processor.
- Turbo Boost will depend on workload, operating environment, and platform design.
- Turbo Boost frequency is selected by the firmware of the CPU (no OS control).
- Maximum turbo frequency is related to the number of active core (CPU version and architecture dependent).
- Turbo frequency is related to the mix of instructions executed in the last period of time not from CPU thermal condition (turbo frequency for SIMD instructions ≠ scalar instructions).
- Energy efficient turbo* attempts to reduce the usage of turbo frequencies that do not significantly increase the performance monitoring the stall cycles and selecting a next frequency that is predicted to be optimal. It is driven by the “Energy Performance BIAS Hint” setting.
- Turbo Boost Technology 3 -> use the core with the best performance per power for single threaded applications

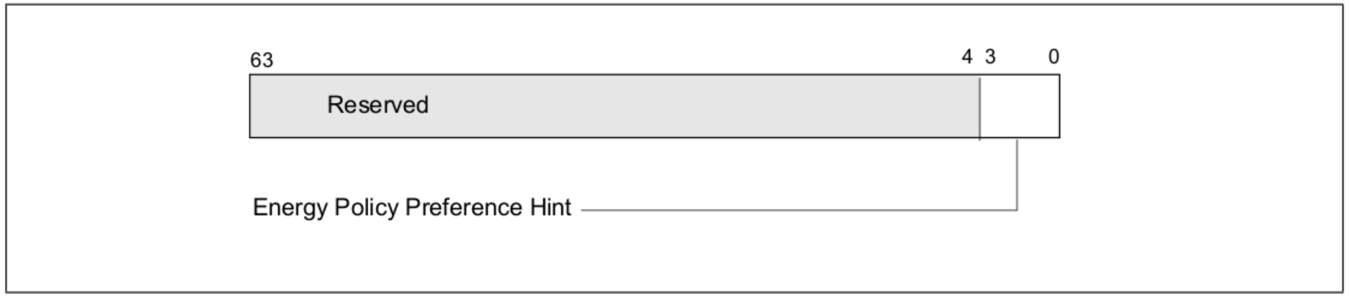


*World Intellectual Property Organization
International Publication Number:
WO 2013/137859 Al
Pub. Date: Sep. 19, 2013

Performance and Energy Bias Hint support

Intel 64 processors may support additional software hint to guide the hardware heuristic of power management features to favor increasing dynamic performance or conserve energy consumption (affect Turbo boost and Uncore Frequency Scaling).

- Software can program the lowest four bits of **IA32_ENERGY_PERF_BIAS** (0x1B0) MSR with a value from 0 - 15.
- **Value of 0**: corresponds to a hint preference for highest performance
- **value of 15**: corresponds to the maximum energy savings.
- **A value of 7** (default) roughly translates into a hint to balance performance with energy consumption.



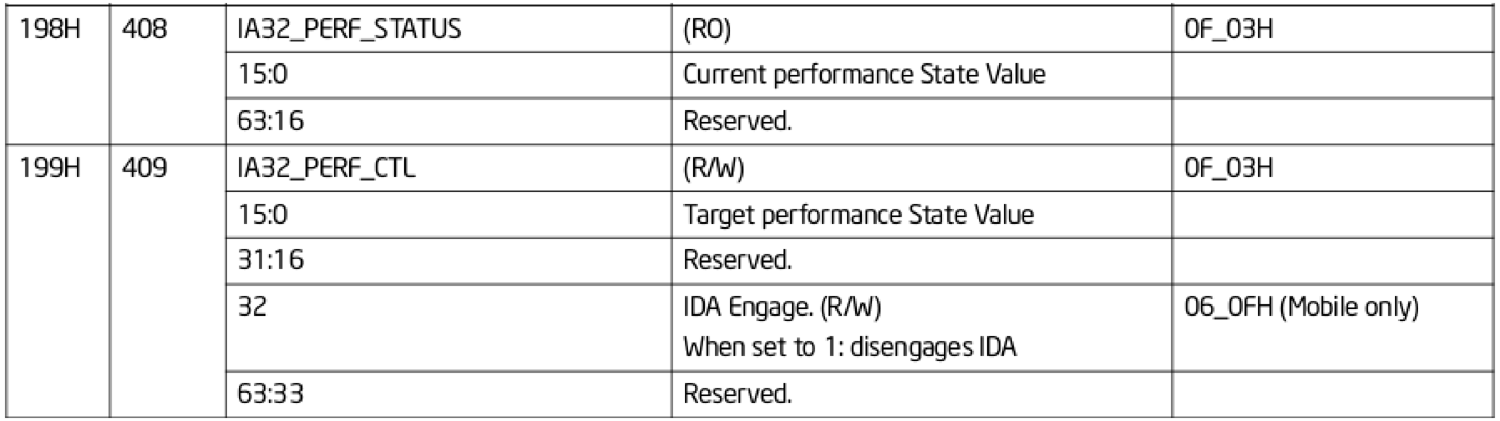
Sysfs interface: /sys/devices/system/cpu/cpu*/power/energy_perf_bias

Intel P-State
Machine Specific Registers (MSR)

Intel P-State driver make state transitions writing a 16-bit value to the **IA32_PERF_CTL** (0x198) register. Reads of **IA32_PERF_CTL** (0x199) determine the targeted operating point. If a transition is already in progress, transition to a new value will subsequently take effect. The current operating point can be read from **IA32_PERF_STATUS**. **IA32_PERF_STATUS** is updated dynamically when target p-state is reached. Turbo boost can be also enabled/disabled by **IA32_MISC_ENABLE** (0x1A0) registers.

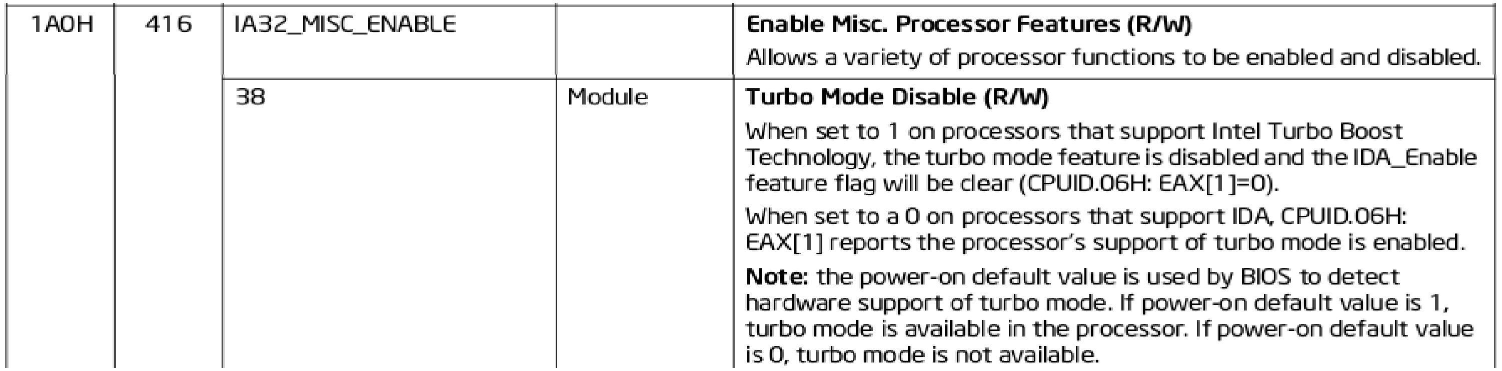
IA32_PERF_STATUS

IA32_PERF_CTL



198H	408	IA32_PERF_STATUS	(RO)	0F_03H
		15:0	Current performance State Value	
		63:16	Reserved.	
199H	409	IA32_PERF_CTL	(R/W)	0F_03H
		15:0	Target performance State Value	
		31:16	Reserved.	
		32	IDA Engage. (R/W) When set to 1: disengages IDA	06_0FH (Mobile only)
		63:33	Reserved.	

IA32_MISC_ENABLE
Turbo boost
enable/disable



1A0H	416	IA32_MISC_ENABLE		**Enable Misc. Processor Features (R/W)** Allows a variety of processor functions to be enabled and disabled.
		38	Module	**Turbo Mode Disable (R/W)** When set to 1 on processors that support Intel Turbo Boost Technology, the turbo mode feature is disabled and the IDA_Enable feature flag will be clear (CPUID.06H: EAX[1]=0). When set to a 0 on processors that support IDA, CPUID.06H: EAX[1] reports the processor's support of turbo mode is enabled. **Note:** the power-on default value is used by BIOS to detect hardware support of turbo mode. If power-on default value is 1, turbo mode is available in the processor. If power-on default value is 0, turbo mode is not available.

Hardware-coordination feedback mechanism APERF/MPERF

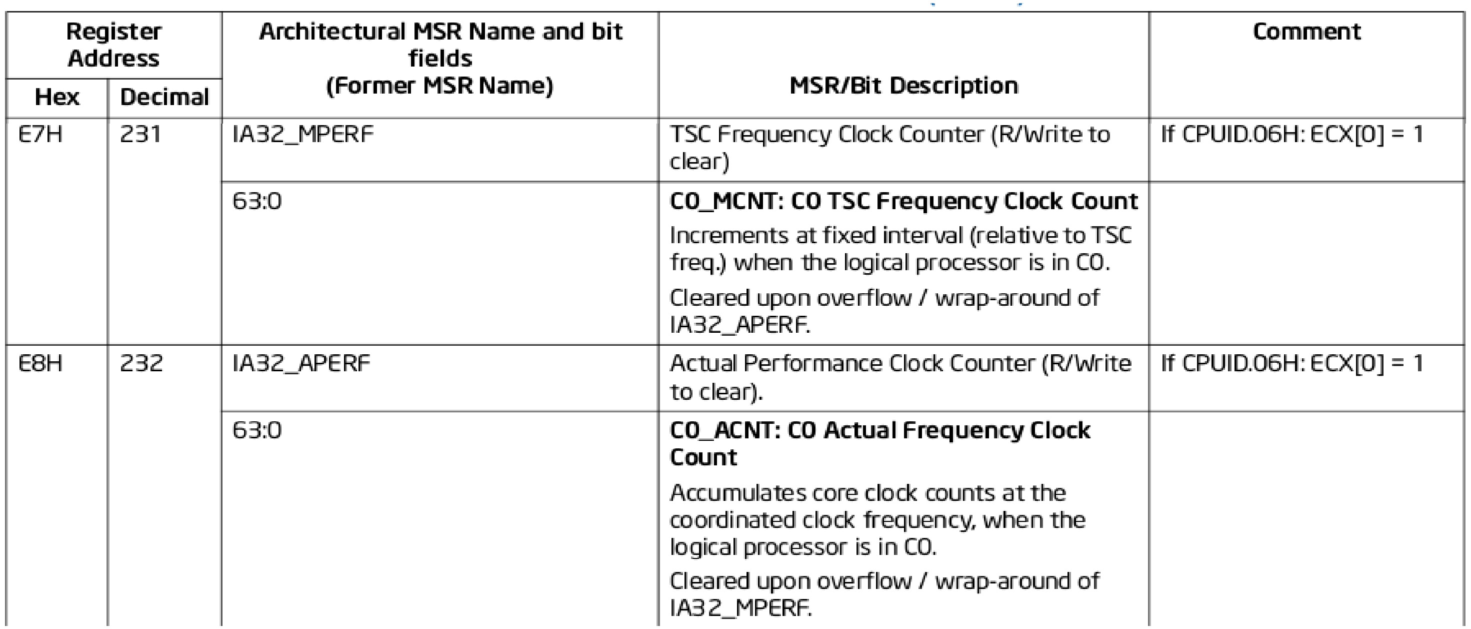
MPERF (0xE7) is a counter that increments itself relative at the nominal frequency at every cycle in C0 status. While, **APERF** (0xEB) increments itself coordinated at clock frequency in C0 status. Current P-state is computed like:

P-state = (delta_APERF / delta_MPERF) * max_pstate

APERF and MPERF are calculated as a different over a measurement interval. The interval is defined to be the time between successive reads (delta_X) on that core.

IA32_MPERF

IA32_APERF



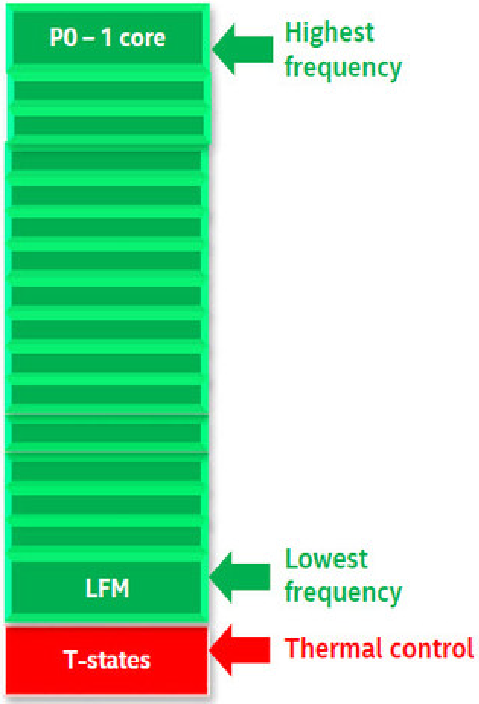
Register Address		Architectural MSR Name and bit fields (Former MSR Name)	MSR/Bit Description	Comment
Hex	Decimal			
E7H	231	IA32_MPERF	TSC Frequency Clock Counter (R/Write to clear)	If CPUID.06H: ECX[0] = 1
		63:0	**C0_MCNT: C0 TSC Frequency Clock Count** Increments at fixed interval (relative to TSC freq.) when the logical processor is in C0. Cleared upon overflow / wrap-around of IA32_APERF.	
E8H	232	IA32_APERF	Actual Performance Clock Counter (R/Write to clear).	If CPUID.06H: ECX[0] = 1
		63:0	**C0_ACNT: C0 Actual Frequency Clock Count** Accumulates core clock counts at the coordinated clock frequency, when the logical processor is in C0. Cleared upon overflow / wrap-around of IA32_MPERF.	

Trick: APERF/MPERF can be also used to check the average frequency in the last period of time:

CPU freq = (delta_APERF / delta_MPERF) * nominal_freq

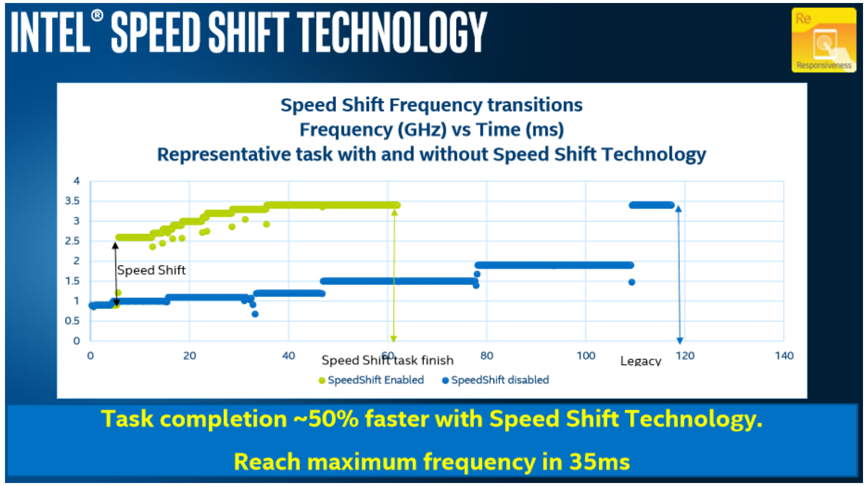
Intel SpeedShift Technology
HWP (aka Hardware P-States)

If the processor is capable of selecting its next P-State internally, then the driver will offload this responsibility to the processor (aka HWP: Hardware P-States). Skylake and newer architectures equip Intel SpeedShift Technology.



- Why change:
  - Highly dynamic power – Multi core, AVX, accelerators
  - Small form factors → large turbo range
  - Smarter power management enables better choices
    - Finer grain and micro architectural observability
- How:
  - Expose entire frequency range
  - A new deal - OS and hardware share power/perf. control
    - OS direct control when and where desired
    - Autonomous control by PCU elsewhere

- Speed Shift vs P-state transitions
  - Speed Shift : 1 ms
  - P-state: 10 ms
- From efficient power state to maximum performance
  - Speed Shift: 35 ms
  - P-state: 100 ms

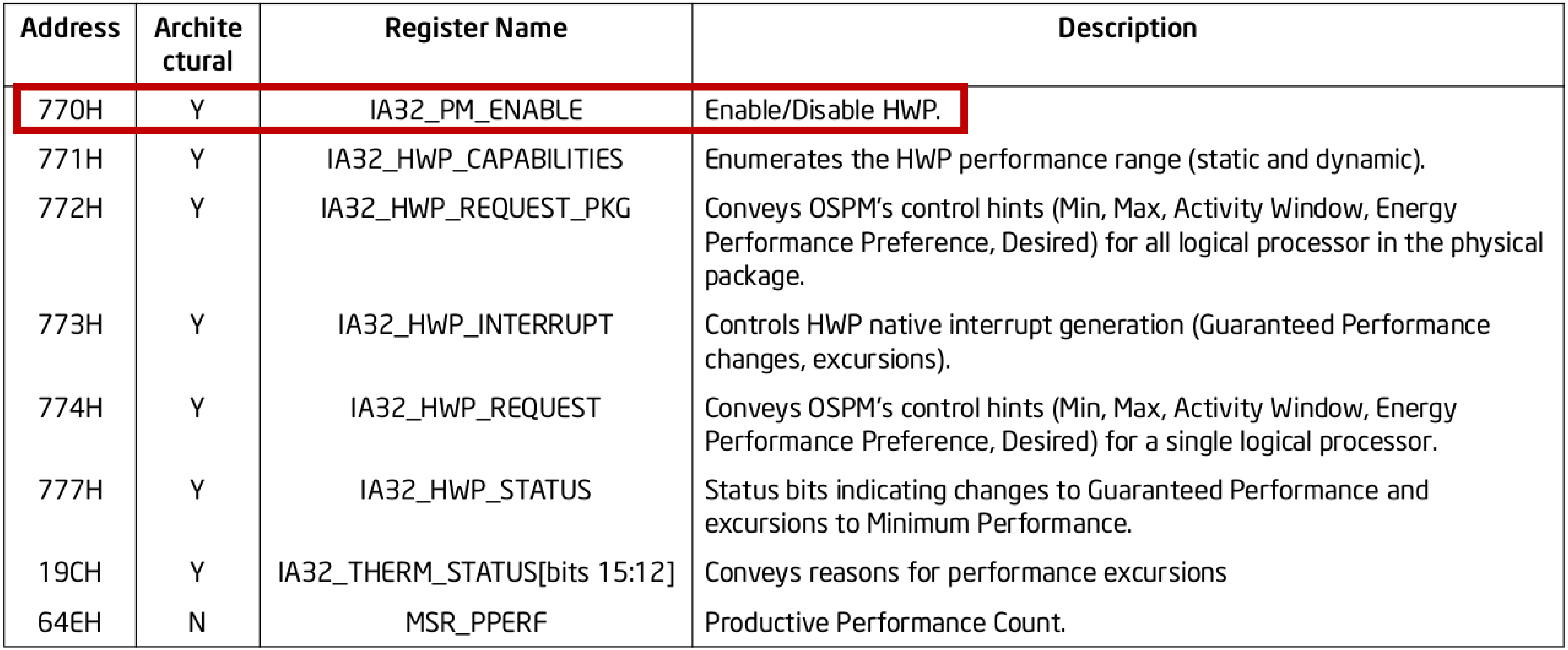

INTEL® SPEED SHIFT TECHNOLOGY

Speed Shift Frequency transitions
Frequency (GHz) vs Time (ms)
Representative task with and without Speed Shift Technology

Task completion ~50% faster with Speed Shift Technology.

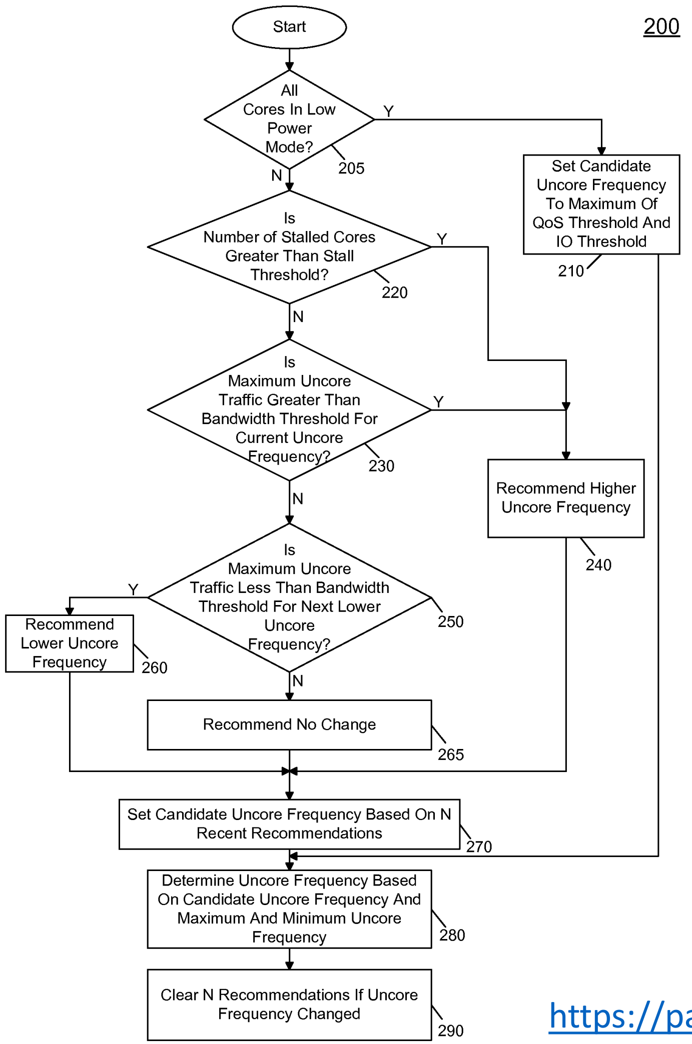
Reach maximum frequency in 35ms

Configuration of Intel SpeedShift Technology (HWP)



Address	Architectural	Register Name	Description
770H	Y	IA32_PM_ENABLE	Enable/Disable HWP.
771H	Y	IA32_HWP_CAPABILITIES	Enumerates the HWP performance range (static and dynamic).
772H	Y	IA32_HWP_REQUEST_PKG	Conveys OSPM's control hints (Min, Max, Activity Window, Energy Performance Preference, Desired) for all logical processor in the physical package.
773H	Y	IA32_HWP_INTERRUPT	Controls HWP native interrupt generation (Guaranteed Performance changes, excursions).
774H	Y	IA32_HWP_REQUEST	Conveys OSPM's control hints (Min, Max, Activity Window, Energy Performance Preference, Desired) for a single logical processor.
777H	Y	IA32_HWP_STATUS	Status bits indicating changes to Guaranteed Performance and excursions to Minimum Performance.
19CH	Y	IA32_THERM_STATUS[bits 15:12]	Conveys reasons for performance excursions
64EH	N	MSR_PPERF	Productive Performance Count.

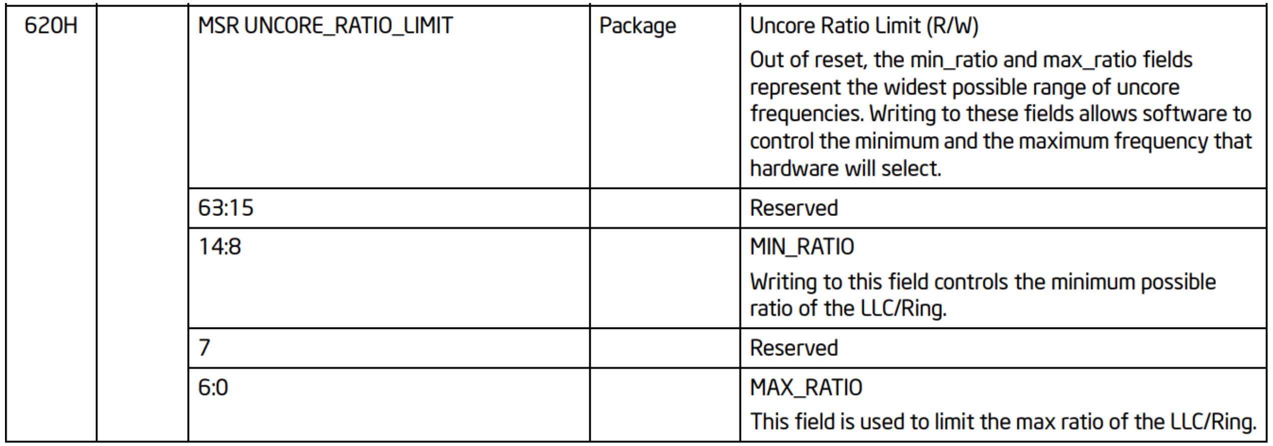
Uncore Frequency Scaling


Start
200
All Cores In Low Power Mode?
Y
N
205
Set Candidate Uncore Frequency To Maximum Of QoS Threshold And IO Threshold
210
Is Number of Stalled Cores Greater Than Stall Threshold?
Y
N
220
Is Maximum Uncore Traffic Greater Than Bandwidth Threshold For Current Uncore Frequency?
Y
N
230
Recommend Higher Uncore Frequency
240
Is Maximum Uncore Traffic Less Than Bandwidth Threshold For Next Lower Uncore Frequency?
Y
N
250
Recommend Lower Uncore Frequency
260
Recommend No Change
265
Set Candidate Uncore Frequency Based On N Recent Recommendations
270
Determine Uncore Frequency Based On Candidate Uncore Frequency And Maximum And Minimum Uncore Frequency
280
Clear N Recommendations If Uncore Frequency Changed
290

Characteristics:

- The uncore frequency is set transparently for operating system and user.
- Uncore frequency is stalled when Package C-state enter in C3 or deeper state.
- The choice of uncore frequency is driven by the number of active cores and the traffic on the on-chip interconnect.
- The uncore frequency scaling partly base their frequency decisions on the “Energy performance BIAS Hints” setting.
- Uncore frequency is bounded to a max and a min frequency range specified in the “MSR_UNCORE_RATIO_LIMIT”.

  Trick: if max and min parameters are set to the same value the uncore frequency doesn't change!



620H		MSR UNCORE_RATIO_LIMIT	Package	Uncore Ratio Limit (R/W) Out of reset, the min_ratio and max_ratio fields represent the widest possible range of uncore frequencies. Writing to these fields allows software to control the minimum and the maximum frequency that hardware will select.
		63:15		Reserved
		14:8		MIN_RATIO Writing to this field controls the minimum possible ratio of the LLC/Ring.
		7		Reserved
		6:0		MAX_RATIO This field is used to limit the max ratio of the LLC/Ring.

https://patentimages.storage.googleapis.com/bb/a2/44/354a91cab2e94e/US20140208141A1.pdf

United States Patent Application Publication Pub. No.: US 2014/0208141 A1 Bhandaru et al. Pub. Date: Jul. 24, 2014

Sysfs for intel_pstate (and cpufreq)

Sysfs Interface: /sys/devices/system/cpu/intel_pstate/*

- **max_perf_pct**: Maximum P-state the driver is allowed to set in percent of the maximum supported performance level (the highest supported turbo P-state).
- **min_perf_pct**: Minimum P-state the driver is allowed to set in percent of the maximum supported performance level (the highest supported turbo P-state).
- **turbo_pct**: Ratio of the turbo range size to the size of the entire range of supported P-states, in percent.
- **status**: Operation mode of the driver: “active”, “passive” or “off”.
- **num_pstates**: Number of P-states supported by the processor (between 0 and 255 inclusive) including both turbo and non-turbo P-states.
- **no_turbo**: If set (equal to 1), the driver is not allowed to set any turbo P-states.
- **hwp_dynamic_boost**: (equal to 1) controls the hardware P-States booting. Allowing intel_pstate to use iowait boosting in the active mode with HWP enabled.

Sysfs Interface: /sys/devices/system/cpu/cpu*/cpufreq/*

- **scaling_governor**: current governor
- **scaling_cur_freq**: current frequency (in 100mhz)
- **cpuinfo_max_freq**: max available frequency (in 100mhz)
- **scaling_setspeed**: set the current frequency (not available with intel_pstate)
- **scaling_min_freq**: minimum frequency that the driver can set (in 100mhz)
- **scaling_max_freq**: maximum frequency that the driver can set (in 100mhz)
- **scaling_driver**: current driver (intel_pstate, intel_cpufreq, acpi_cpufreq)
- **scaling_available_governors**: available governors
- **related_cpus**: current CPU id
- **energy_performance_preference**: set the energy performance preference for HW P-state
- **energy_performance_available_preferences**: available energy performance preferences
- **cpuinfo_min_freq**: minimum frequency that CPU can run (in 100mhz)
- **base_frequency**: nominal frequency (in 100mhz)
- **affected_cpus**: id CPU of the hyperthreading sibling

Turbo and AVX Frequencies

Recap and Numbers (1): Turbo frequencies

- Are used when processor assumes (parts of) the processor do(es) not require as much power as specified (e.g., the TDP)
- Controllable influencing factors on power consumption:
  - Activity on processor
  - ~~Frequency~~
  - ~~Voltage~~
- Sources:
  - [HSW] Hackenberg, D., Schöne, R., Ilsche, T., Molka, D., Schuchart, J., & Geyer, R. (2015, May). *An energy efficiency feature survey of the intel haswell processor*. In 2015 IEEE International Parallel and Distributed Processing Symposium Workshop (pp. 896-904). IEEE.
  - [SKY] Schöne, R., Ilsche, T., Bielert, M., Gocht, A., & Hackenberg, D. (2019). *Energy Efficiency Features of the Intel Skylake-SP Processor and Their Impact on Performance*. arXiv preprint arXiv:1905.12468.

Recap and Numbers (1): ~~Turbo~~AVX frequencies [SKY]

- When activity includes “heavy” AVX instructions, the nominal frequency is also considered turbo
- Multiple bands of frequencies:

$$\begin{Bmatrix} AVX-512 \\ AVX \\ normal \end{Bmatrix} x\{\#used\ cores\} \rightarrow (f_{min}, f_{max})$$

Recap and Numbers (1): Turbo/AVX frequencies [HSW,SKY]

- https://github.com/tud-zih-energy/FIRESTARTER
- Test 1: Constant usage of AVX/AVX-512 on Haswell/Skylake
- Original set frequency: Turbo

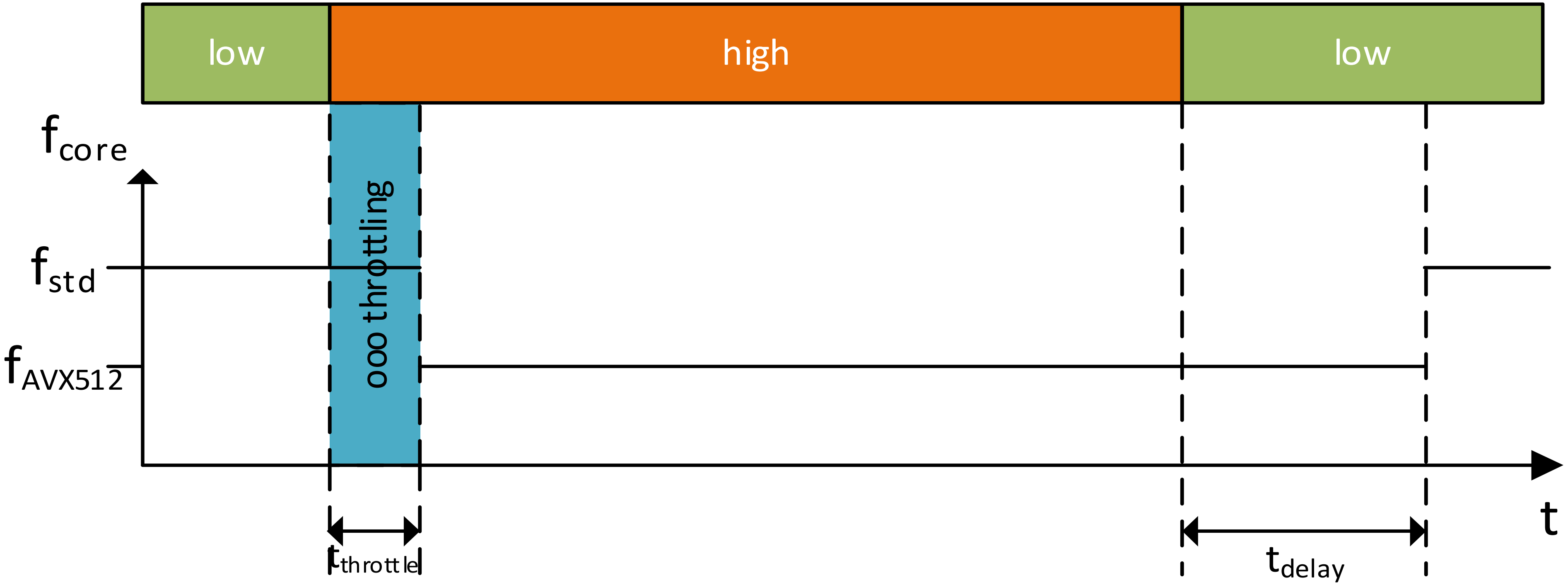
➢Usage of AVX/AVX-512 frequency band

- When disabling turbo/going below nominal and EPB!=performance→ core frequency is increased
- Uncore frequency can be lowered below the given range (even though specification leaves room for lowering f_{core})

Recap and Numbers (2): AVX frequencies [SKY]

- When AVX and AVX-512 is handled differently, then
  - A mechanism has to detect the presence of such instructions
  - The mechanism has to act according to presence
- On AVX-enter (detection of AVX-instructions)
  - Throttle out-of-order execution
  - Request AVX-frequency (applied after $t_{throttle}$)
- At AVX-exit (no more AVX-instructions)
  - Start timer (t_{delay}) to fall back into normal mode

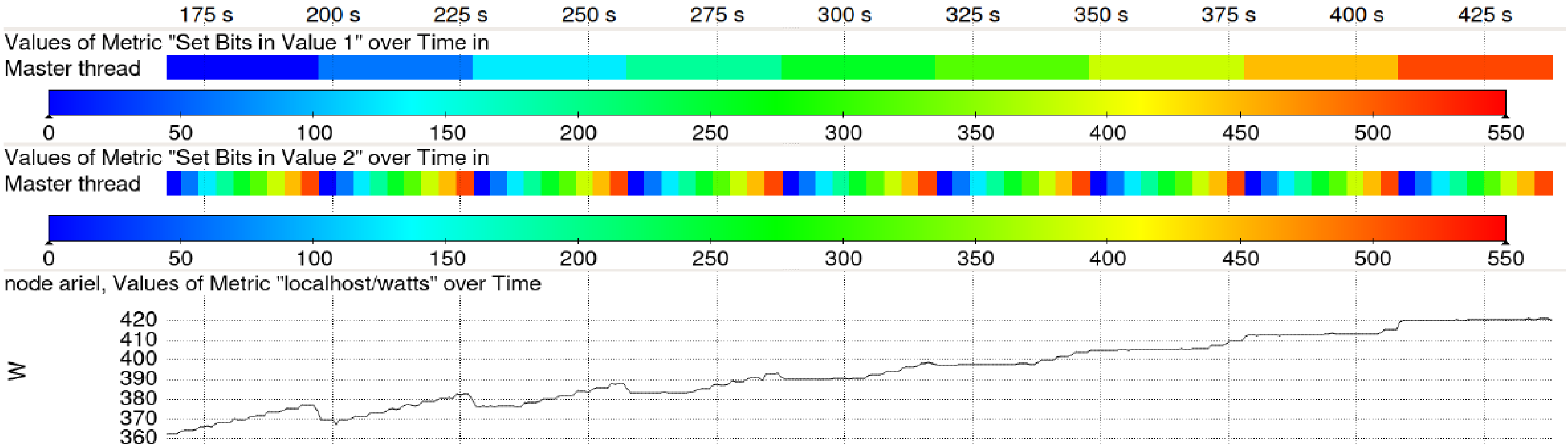
Recap and Numbers (2): AVX frequencies [SKY]



Recap and Numbers (2): AVX frequencies [SKY]

- Measured via Hardware Performance Counters
  - CORE_POWER.THROTTLE
  - CORE_POWER.LVL2_TURBO_LICENSE
- Change FIRESTARTER, instrument with Score-P
  - High phase: intense, AVX-512
  - Low phase: power saving, mfence+cpuid
- 62 µs–75 µs $t_{throttle}$
- ~700 µs t_{delay}
- Worst case scenario: python+MKL
- Ooo throttling also when n. necessary (f=1.2 GHz)

Frequency also dependent on data [SKY]



High core number

Wide SIMD instructions

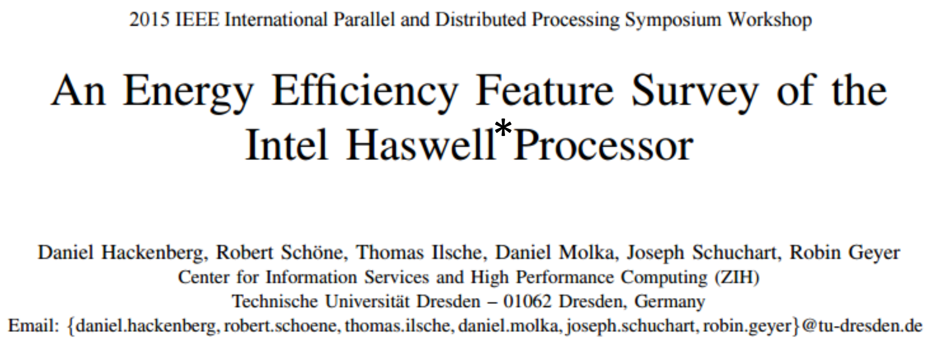
Power varies with bits set in registers

```
vxorps %zmm0,%zmm8,%zmm8
vxorps %zmm1,%zmm9,%zmm9
…
vxorps %zmm7,%zmm15,%zmm15
```

Intel On-chip Power Manager

Intel On-chip Power Manager for Core Frequency [HSW,SKY]

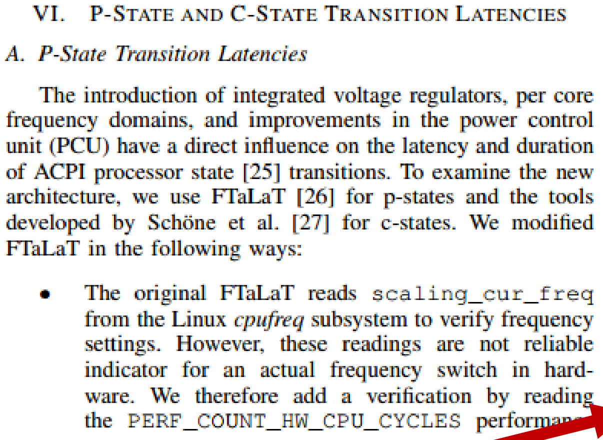
The HW power manager of Intel Architectures is quite slow in frequency variation!
Literatures studied this mechanism and, for reverse engineering, discovered a 500us latency!



2015 IEEE International Parallel and Distributed Processing Symposium Workshop

An Energy Efficiency Feature Survey of the Intel Haswell*Processor

Daniel Hackenberg, Robert Schöne, Thomas Ilsche, Daniel Molka, Joseph Schuchart, Robin Geyer
Center for Information Services and High Performance Computing (ZIH)
Technische Universität Dresden – 01062 Dresden, Germany
Email: {daniel.hackenberg, robert.schoene, thomas.ilsche, daniel.molka, joseph.schuchart, robin.geyer}@tu-dresden.de

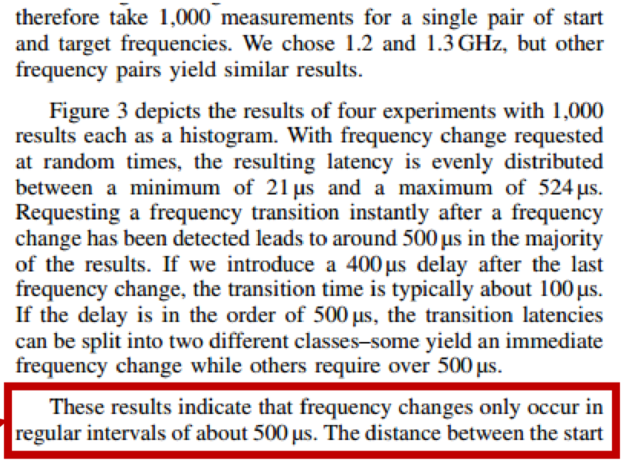


VI. P-State and C-State Transition Latencies

A. P-State Transition Latencies

The introduction of integrated voltage regulators, per core frequency domains, and improvements in the power control unit (PCU) have a direct influence on the latency and duration of ACPI processor state [25] transitions. To examine the new architecture, we use FTaLaT [26] for p-states and the tools developed by Schöne et al. [27] for c-states. We modified FTaLaT in the following ways:

- The original FTaLaT reads `scaling_cur_freq` from the Linux *cpufreq* subsystem to verify frequency settings. However, these readings are not reliable indicator for an actual frequency switch in hardware. We therefore add a verification by reading the `PERF_COUNT_HW_CPU_CYCLES` performan



therefore take 1,000 measurements for a single pair of start and target frequencies. We chose 1.2 and 1.3 GHz, but other frequency pairs yield similar results.

Figure 3 depicts the results of four experiments with 1,000 results each as a histogram. With frequency change requested at random times, the resulting latency is evenly distributed between a minimum of 21 µs and a maximum of 524 µs. Requesting a frequency transition instantly after a frequency change has been detected leads to around 500 µs in the majority of the results. If we introduce a 400 µs delay after the last frequency change, the transition time is typically about 100 µs. If the delay is in the order of 500 µs, the transition latencies can be split into two different classes–some yield an immediate frequency change while others require over 500 µs.

These results indicate that frequency changes only occur in regular intervals of about 500 µs. The distance between the start

500us

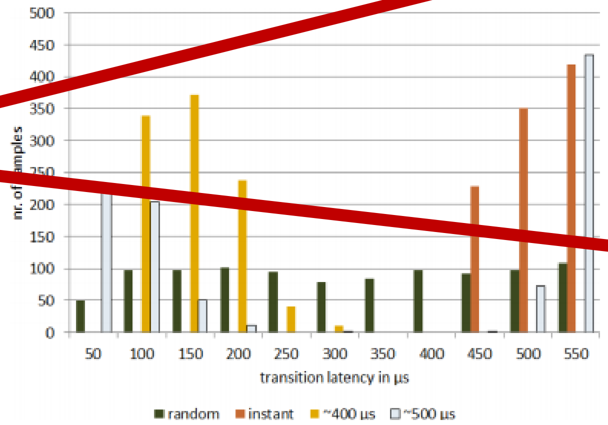
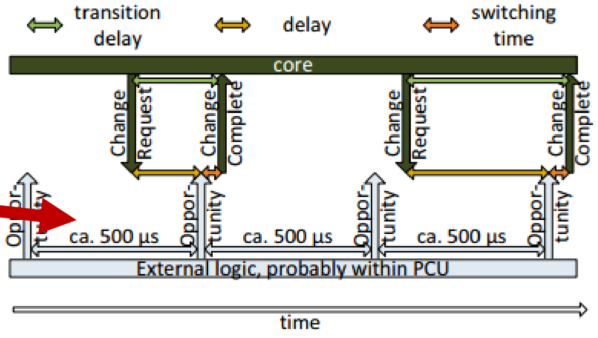
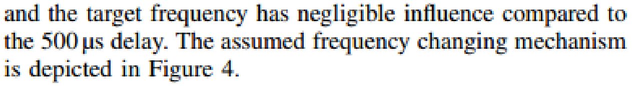


Fig. 3. Histogram of frequency transition latencies for switching between 1.2 and 1.3 GHz, depending on the time since the last frequency change.





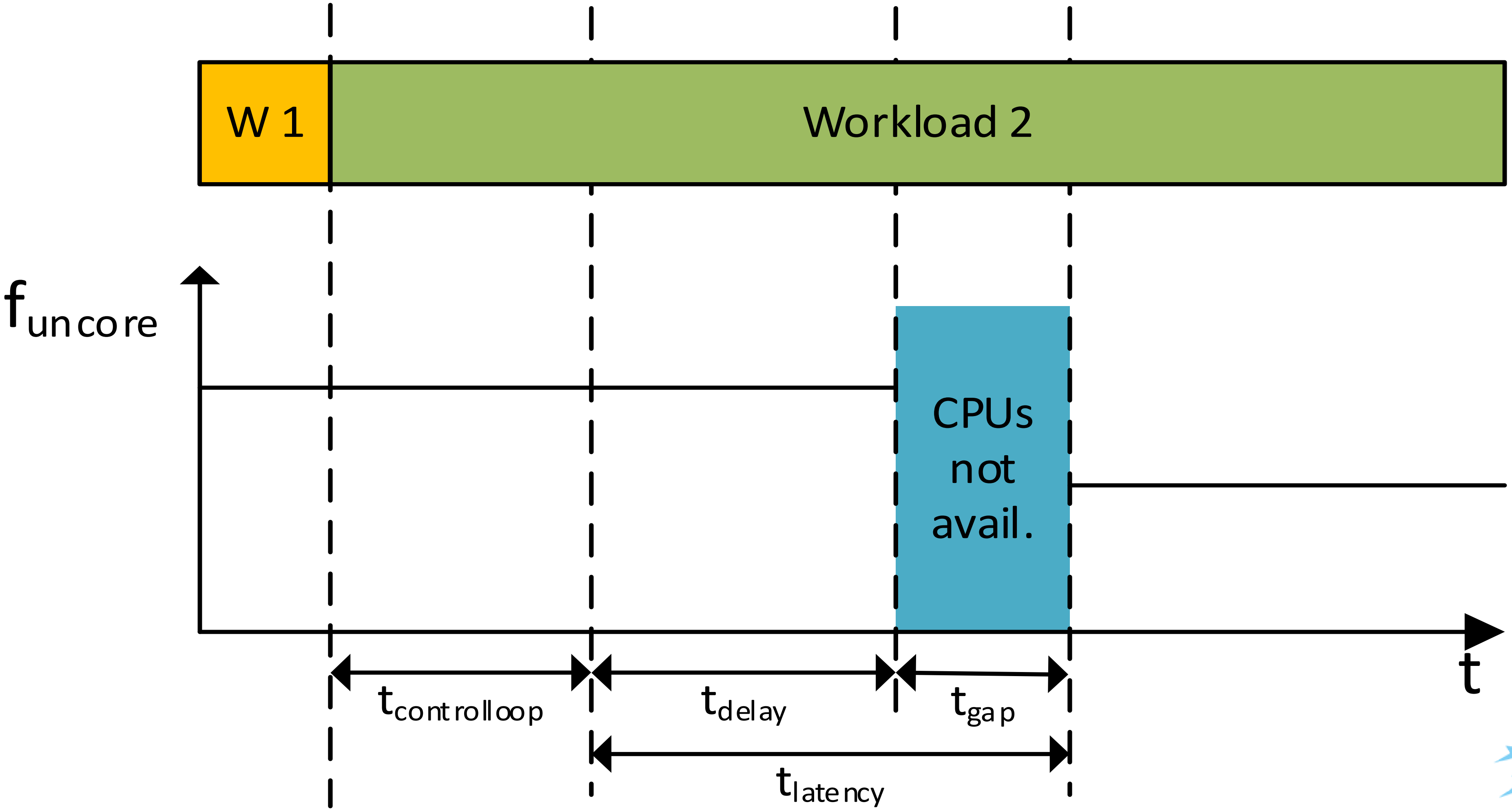
and the target frequency has negligible influence compared to the 500 µs delay. The assumed frequency changing mechanism is depicted in Figure 4.

* Intel Broadwell architectures as well!

Intel On-chip Power Manager for Uncore Frequency [SKY]

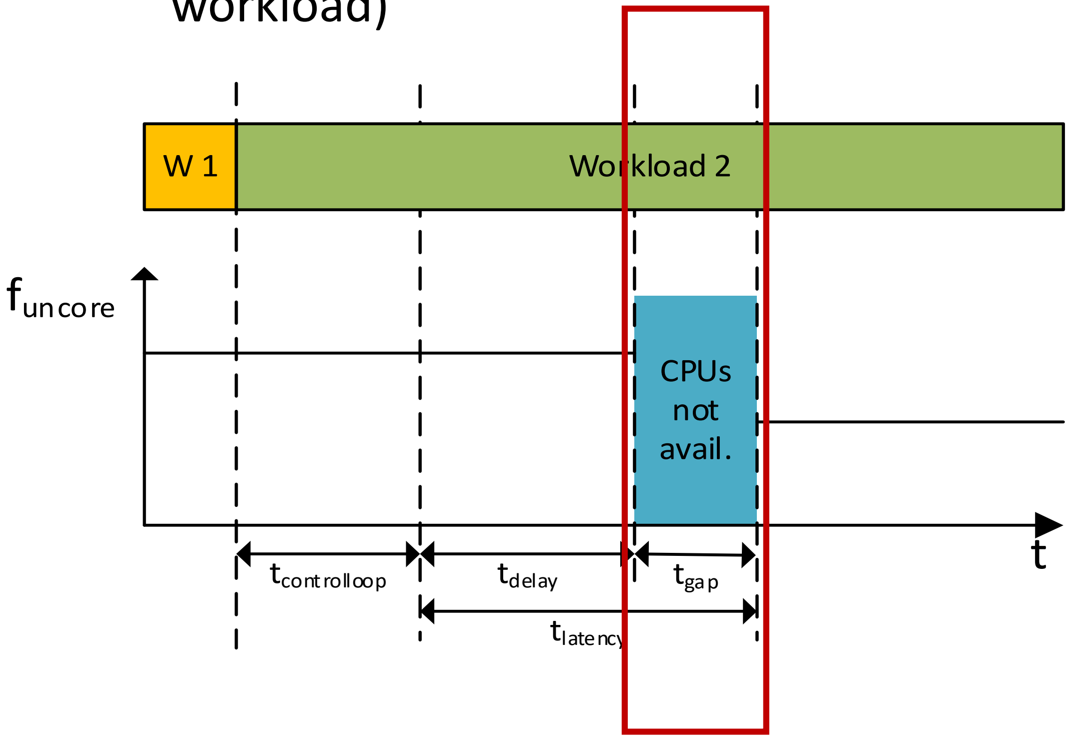
- Usually independently set, alternatively via MSR 0x620
- Manual: How long until takes affect t_{delay}?
- Automatic: How long until detected $t_{controlloop}$?

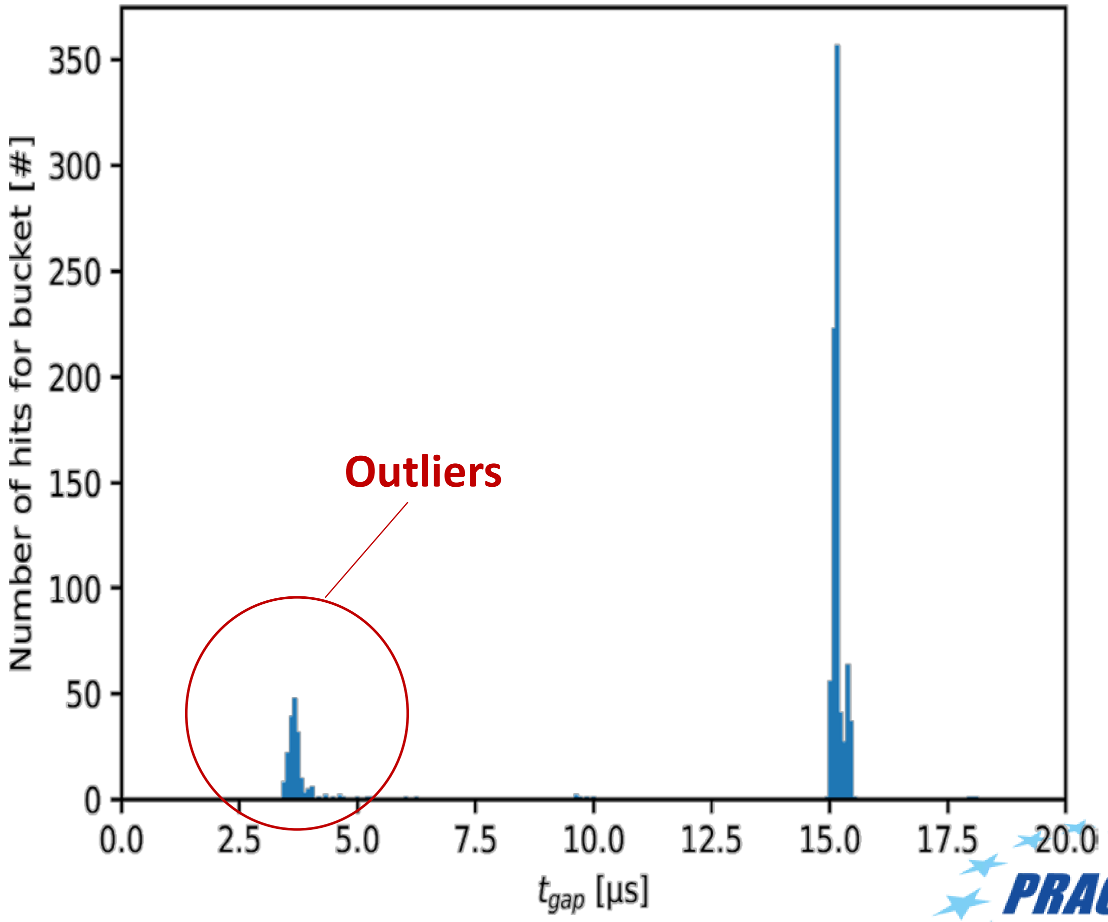
Intel On-chip Power Manager for Uncore Frequency [SKY]



Intel On-chip Power Manager for Uncore Frequency [SKY]

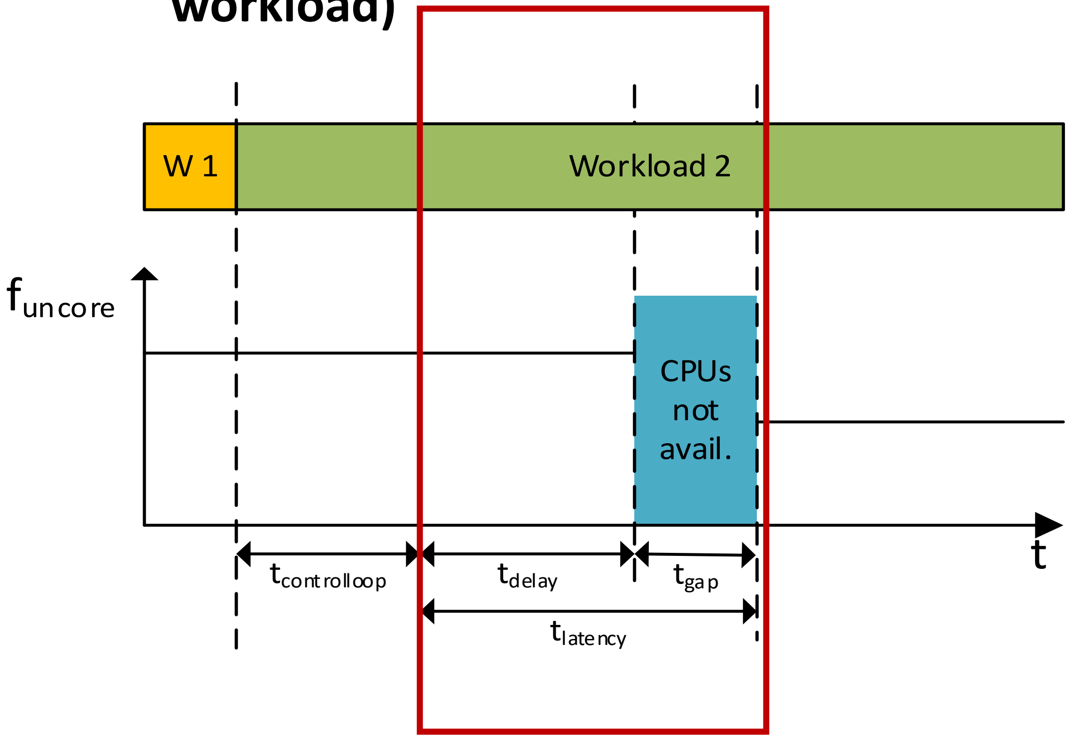
- Change frequency
- While (time[current-1] not **unexpected long**)
  - time[current++]=timer(L3-bound workload)

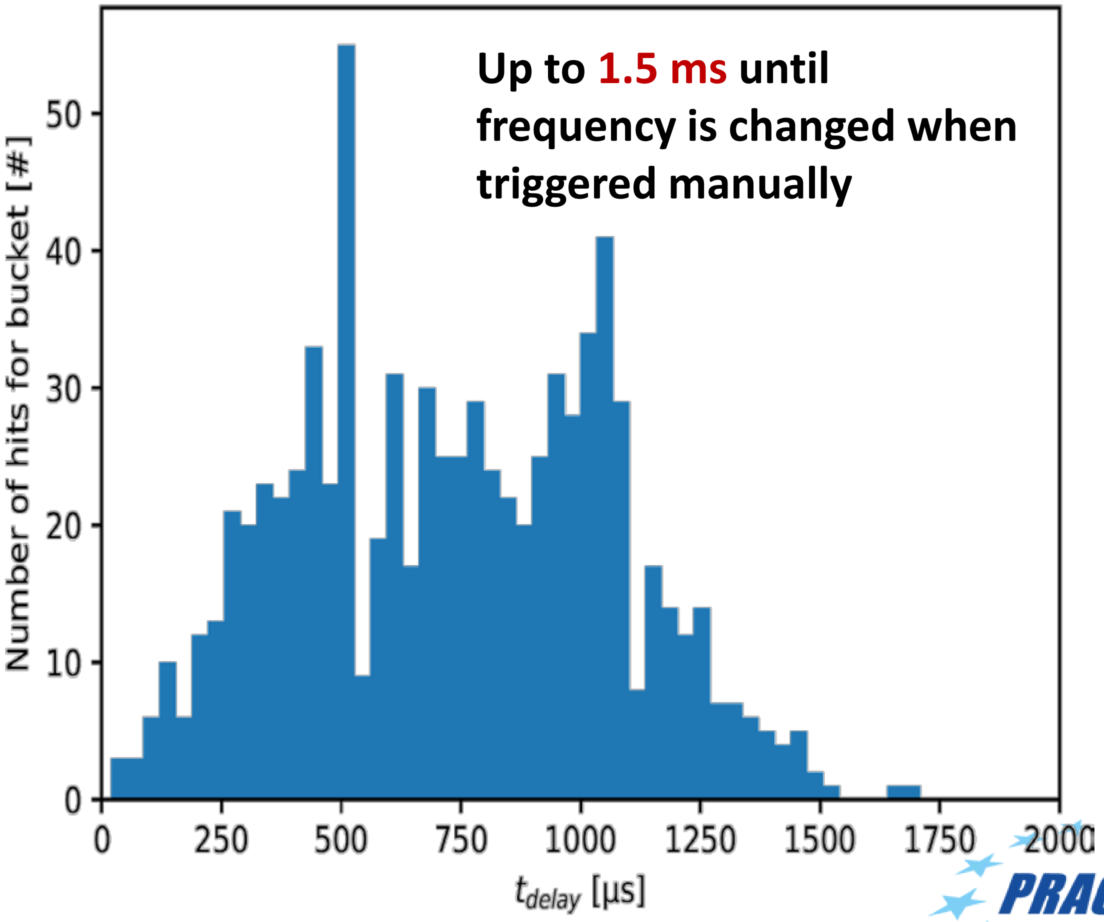




Intel On-chip Power Manager for Uncore Frequency [SKY]

- Change frequency
- **While (time[current-1] not unexpected long)**
  - **time[current++]=timer(L3-bound workload)**





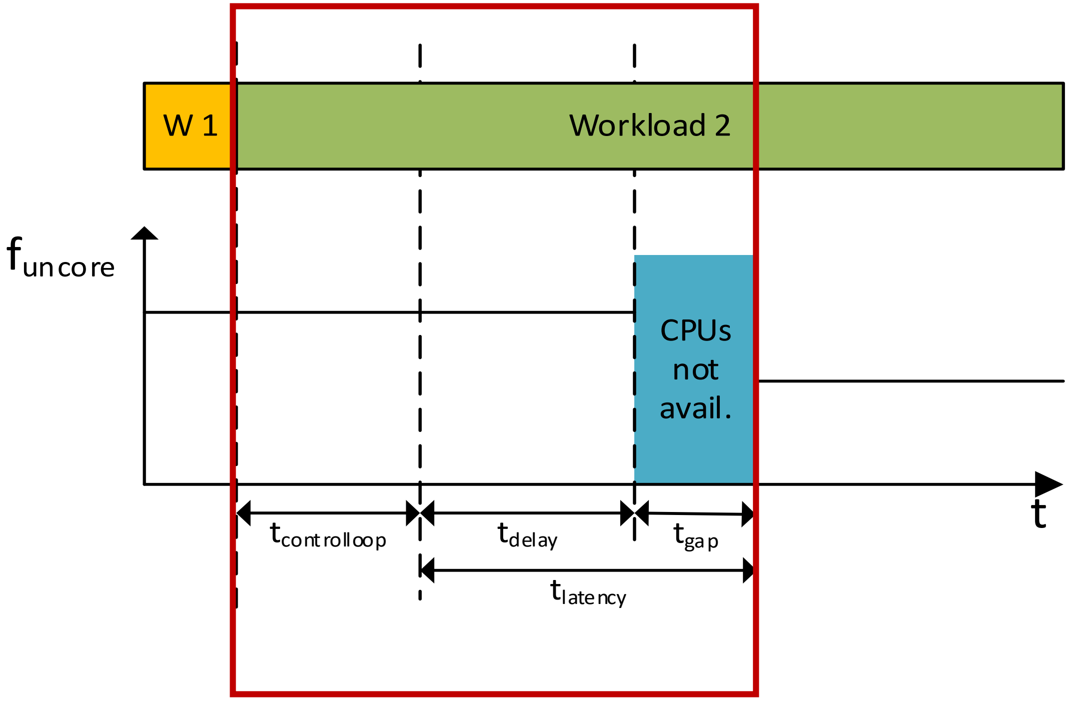
Up to 1.5 ms until frequency is changed when triggered manually

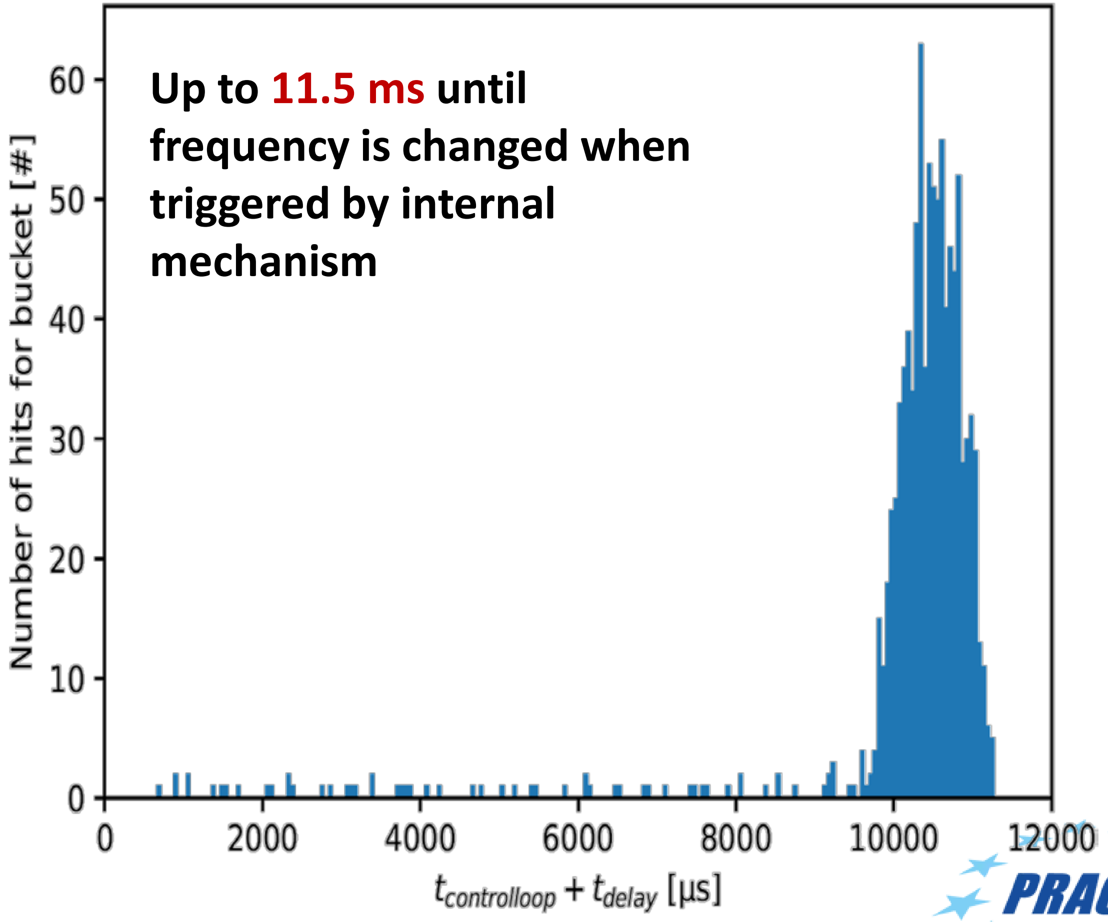
Intel On-chip Power Manager for Uncore Frequency [SKY]

- For (…)

  L1-bound workload

- While (time[current-1] not unexpected long)
  - time[current++]=timer(L3-bound workload)

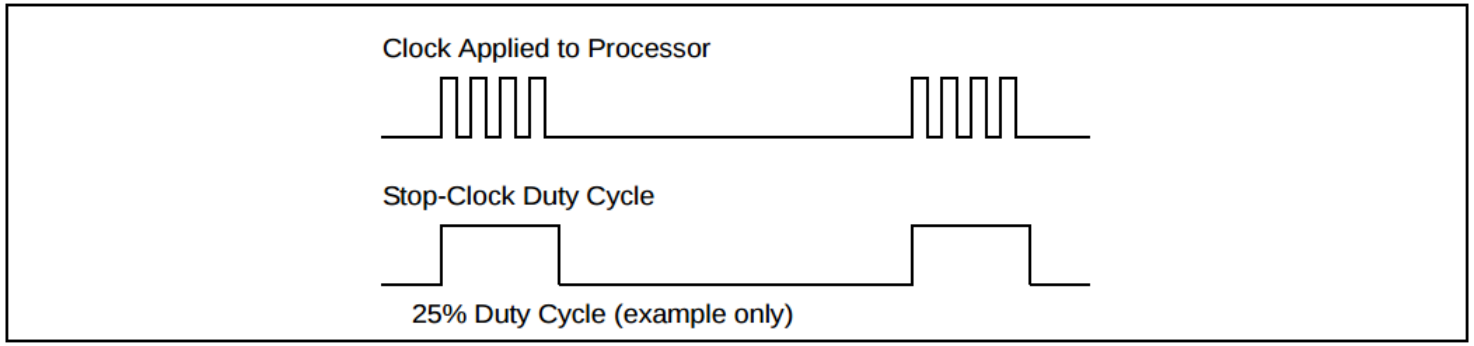


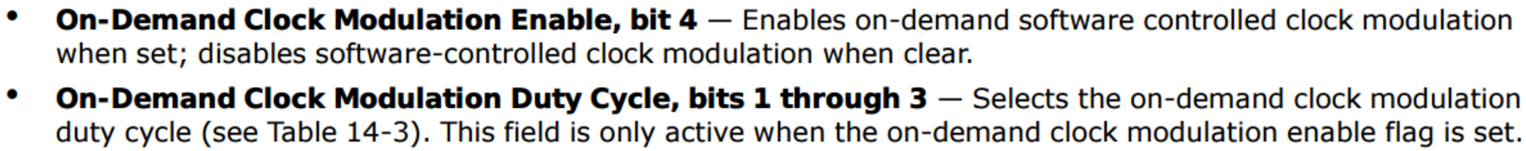


Up to 11.5 ms until frequency is changed when triggered by internal mechanism

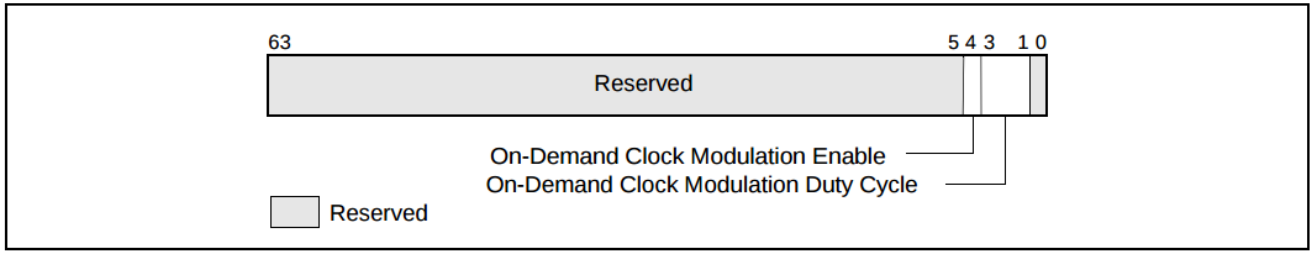
Dynamic Duty Cycle Modulation DDCM (aka T-States)

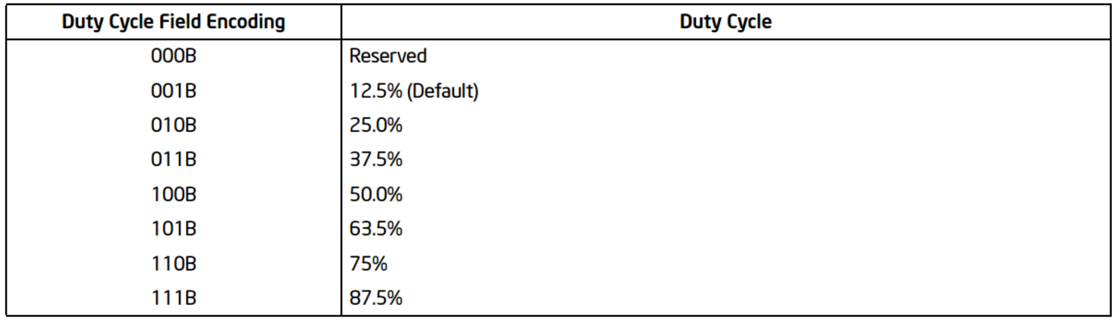
Intel Xeon and Pentium M processors also support software-controlled clock modulation. DDCM was once known as T-state (throttling state). This provides a means for operating systems to implement a power management policy to reduce the power consumption and the temperature of the processor*. Power management software can write to the **IA32_CLOCK_MODULATION** (0x19A) MSR to enable clock modulation and to select a modulation duty cycle.





- **On-Demand Clock Modulation Enable, bit 4** — Enables on-demand software controlled clock modulation when set; disables software-controlled clock modulation when clear.
- **On-Demand Clock Modulation Duty Cycle, bits 1 through 3** — Selects the on-demand clock modulation duty cycle (see Table 14-3). This field is only active when the on-demand clock modulation enable flag is set.


63
5 4 3
1 0
Reserved
On-Demand Clock Modulation Enable
On-Demand Clock Modulation Duty Cycle
Reserved



Duty Cycle Field Encoding	Duty Cycle
000B	Reserved
001B	12.5% (Default)
010B	25.0%
011B	37.5%
100B	50.0%
101B	63.5%
110B	75%
111B	87.5%

*Bhalachandra, Sridutt, Allan Porterfield, and Jan F. Prins. "Using dynamic duty cycle modulation to improve energy efficiency in high performance computing." *2015 IEEE International Parallel and Distributed Processing Symposium Workshop*. IEEE, 2015.

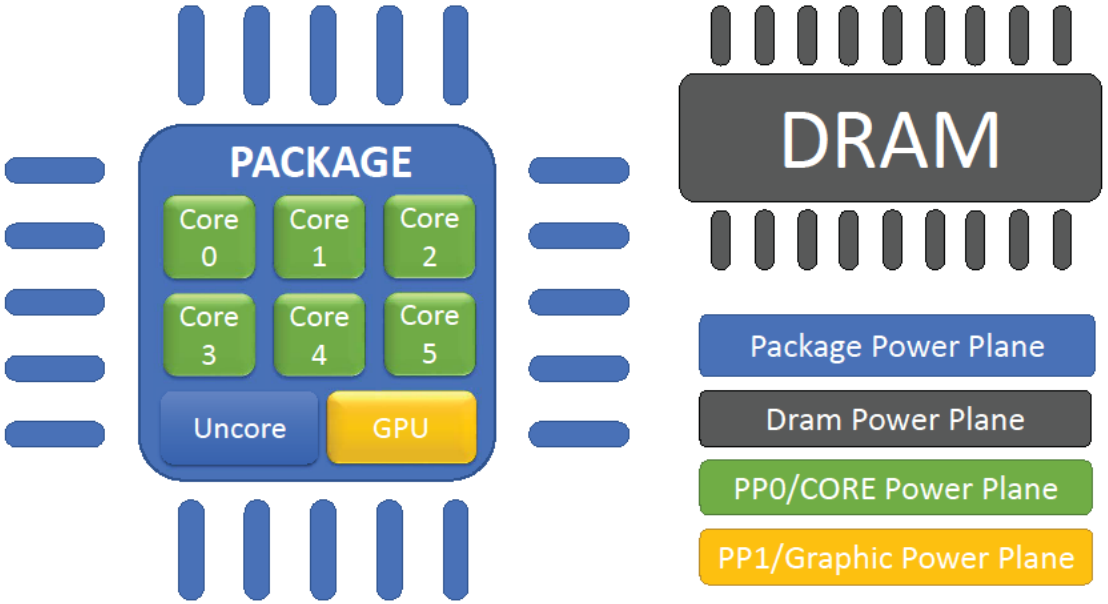
Dynamic Duty Cycle Modulation DDCM (aka T-States)

- You might find papers that say, this works great, probably measured on Sandy Bridge processors
- On some systems (e.g., Sandy Bridge) DVFS is used, when all cores agree to a common T-State – that's why it's "great"
- On Haswell, Skylake:
  - Too many cycles skipped (compared to definition)
  - Highest T-state not available
- In short: Not worth it (only special circumstances, i.e., processor without PCPs, slack-optimization)
- Source: [TState]

RAPL

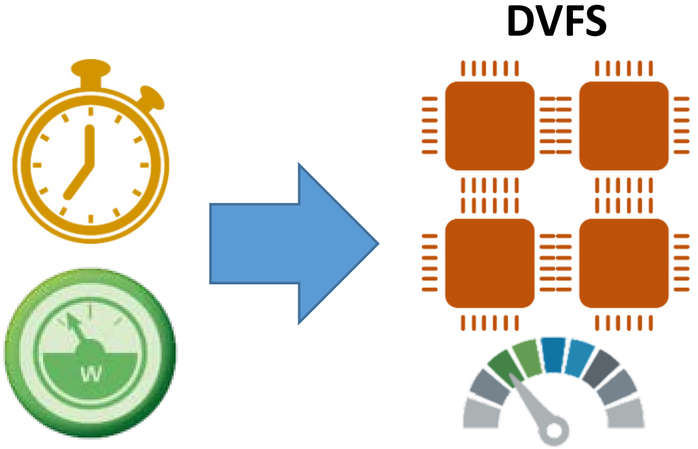
HW Power Control - RAPL

Intel architectures implement a hardware power controller called Running Average Power Limit (RAPL).



Time limiter
(typically from few ms
to seconds)

Power limiter
in Watts
(default TDP)



Power Domains

Package Domain: limits the power consumption for the entire package of the CPU, this includes cores and uncore components.

DRAM Domain: is used to power cap the DRAM memory. It is available only for server architectures. (no client)

PP0/Core Domain: is used to restrict the power limit only to the cores of the CPU (no new server).

PP1/Graphic Domain: is used to power limit only the graphic component of the CPU (no server).

N.B. in the last slides of this presentation there is a complete description of RAPL registers

RAPL – Usage scenarios

- Hardware:
  - Measure power consumption for Turbo frequencies
  - Shift power budgets between cores and GPU
- Operating system/system vendors
  - Set given TDP
  - Monitor power consumption
- Users:
  - Monitor power consumption

RAPL – Mechanism

- Two power ranges:
  - Short term (typically 1 second?), typically exceeds TDP
  - Long term (typically 60 seconds?), typically TDP
- Processor must stay within both limits
- Internally sampled with typically (at least) 1 ms
- Processor can use free budgets for Turbo mechanisms
- Other facts:
  - Initially modelled, now measured
  - Also on AMD since Zen

Sysfs for RAPL

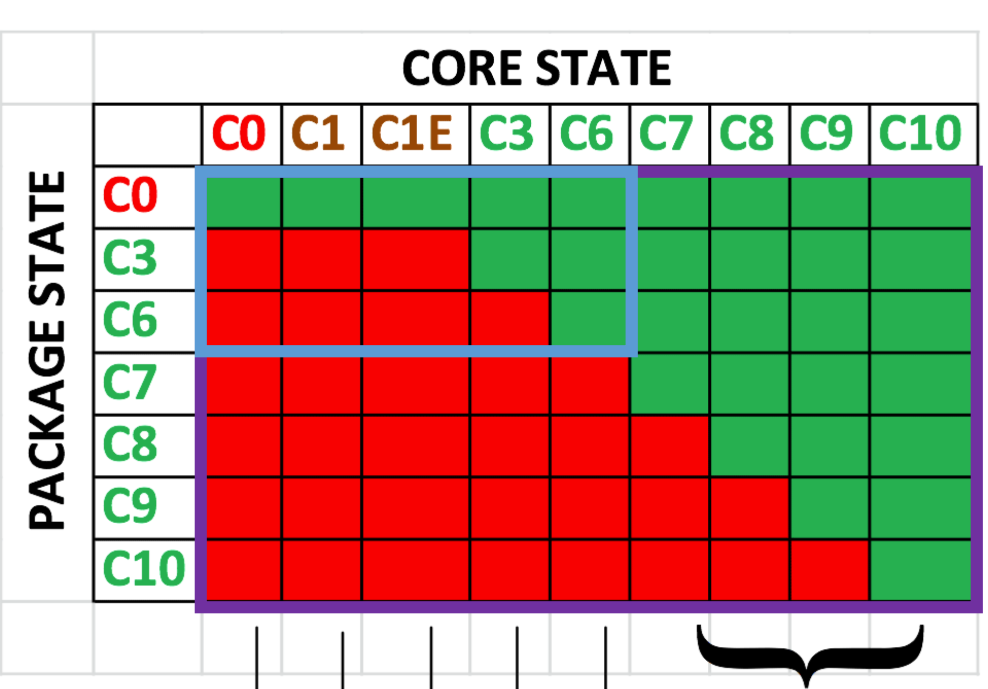
Sysfs Interface: /sys/devices/virtual/powercap/intel-rapl/intel-rapl:X/intel-rapl:0:Y

- X = number of package but this numbering does not reflect the package ID, this level contain the package domain information
- Y = 0 core domain, 1 gpu/uncore domain, 2 dram domain (the numbering can be different check the name)

- **name:** name of the domain and of the package ID
- **max_energy_range_uj:** range of the above energy counter in micro-joules
- **energy_uj:** current energy counter in micro joules
- **enabled:** enable/disable controls at domain level
- **constraint_0_name:** the name of the constraint 0 (usually long term window -> seconds)
- **constraint_0_time_window_us:** time window in micro seconds for the constraint 0
- **constraint_0_power_limit_uw:** power limit in micro watts for the constraint time 0
- **constraint_0_max_power_uw:** maximum allowed power in micro watts for the constraint 0
- **constraint_1_name:** the name of the constraint 1 (usually short term window -> milli seconds)
- **constraint_1_time_window_us:** time window in micro seconds for the constraint 0
- **constraint_1_power_limit_uw:** power limit in micro watts for the constraint time 1
- **constraint_1_max_power_uw:** maximum allowed power in micro watts for the constraint 0

Idle State (aka C-states)

Processor Package and Core C-States



Haswell EP

Haswell ULP
Broadwell
Skylake

CORE STATE

PACKAGE STATE	C0	C1	C1E	C3	C6	C7	C8	C9	C10
C0	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
C3	🟥	🟥	🟥	🟩	🟩	🟩	🟩	🟩	🟩
C6	🟥	🟥	🟥	🟥	🟩	🟩	🟩	🟩	🟩
C7	🟥	🟥	🟥	🟥	🟥	🟩	🟩	🟩	🟩
C8	🟥	🟥	🟥	🟥	🟥	🟥	🟩	🟩	🟩
C9	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟩	🟩
C10	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟩

→ One or more cores or GT executing instructions

→ All cores and GT in C3 or deeper, L3 may be flushed and turned off, memory in self refresh, some Uncore clocks stopped, some Uncore voltages reduced

→ All cores and GT in C6 or deeper, L3 may be flushed and turned off, memory in self refresh, all Uncore clocks stopped, some Uncore voltages reduced

→ Package C6 + L3 flushed and turned off, additional Uncore voltages reduced

→ Package C7 + most Uncore voltages reduced to 0V

→ Package C8 + VR12.6 in low power state

→ Package C9 + VR12.6 turned off

→ Core behaves the same as Core C6 state

→ All core clocks are stopped, core state saved and voltage reduce to 0V

→ Cores flush L1/L2 into L3, all core clocks are stopped

→ Core halted, most core clocks stopped and voltage reduced to Pn

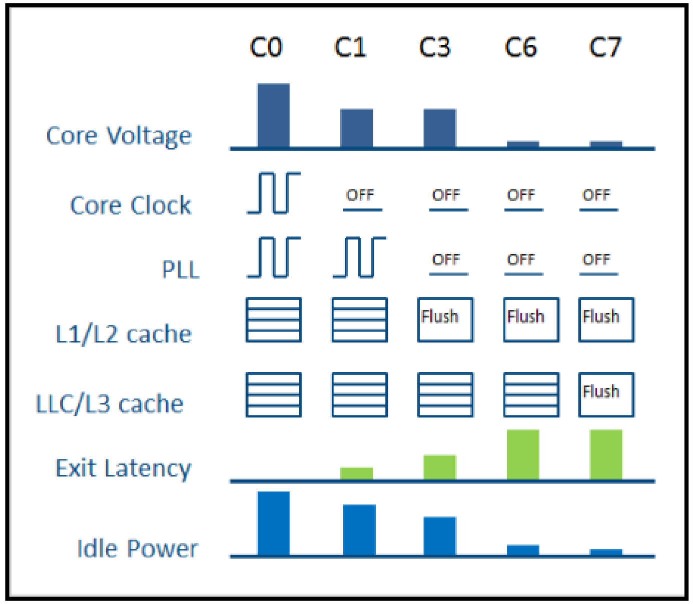
→ Core halted, most core clocks stopped

→ Core is executing code

🟩 Possible combination of core/package states

🟥 Impossible combination of core/package states

Note: The "core state" relates to the core which is in the HIGHEST power state in the package (most active)



Do not trust the last slide

- Check
  /sys/devices/system/cpu/cpuX/cpuidle/stateY
  to find out which (core) C-States your system supports
- This depends on:
  - Server or desktop or mobile
  - BIOS
  - Kernel
- Package C-state X are only used when all cores in the system (not only on one processor) are in core C-state X
- Otherwise, Package C2 can be used on the idling package (reduce latencies)

HLT & MWAIT Instructions

HLT (opcod F4):

- HLT is an assembly language instruction which halts the logical processor (clock gating) until the next external interrupt is fired.
- The HLT instruction is a privileged instruction, it requires ring 0 access, it can only be run by privileged system software such as the kernel (not in Xeon Phi!).
- The HLT instruction is executed by the operating system when there is no immediate work to be done, and the system enters its idle state (C1/C1E).

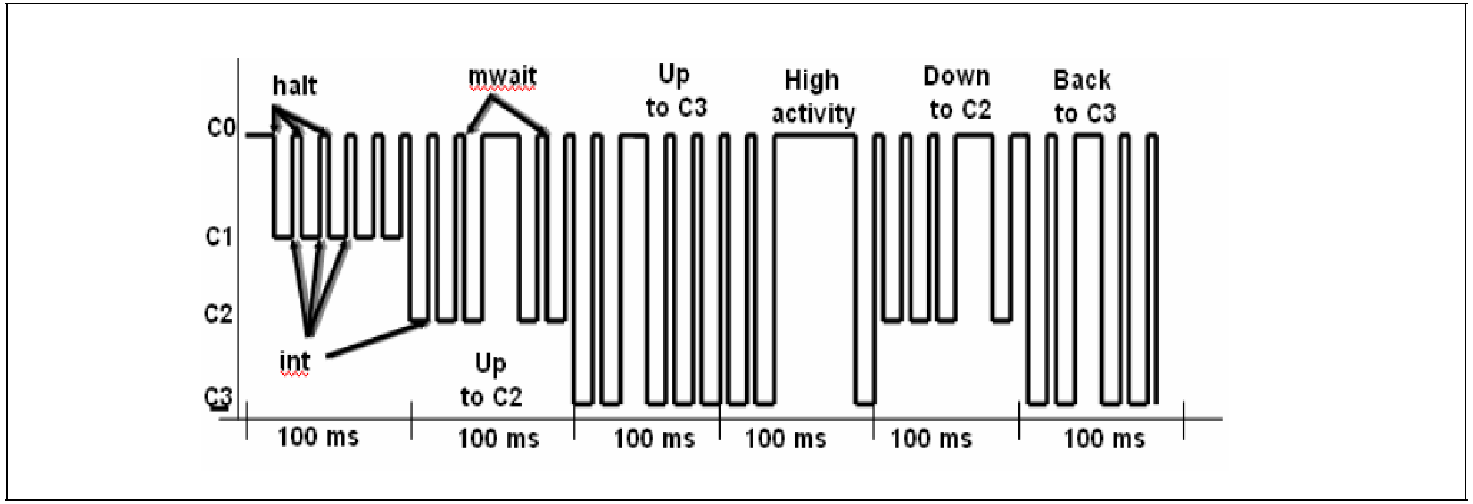
MWAIT (opcod 0F 01 C9):

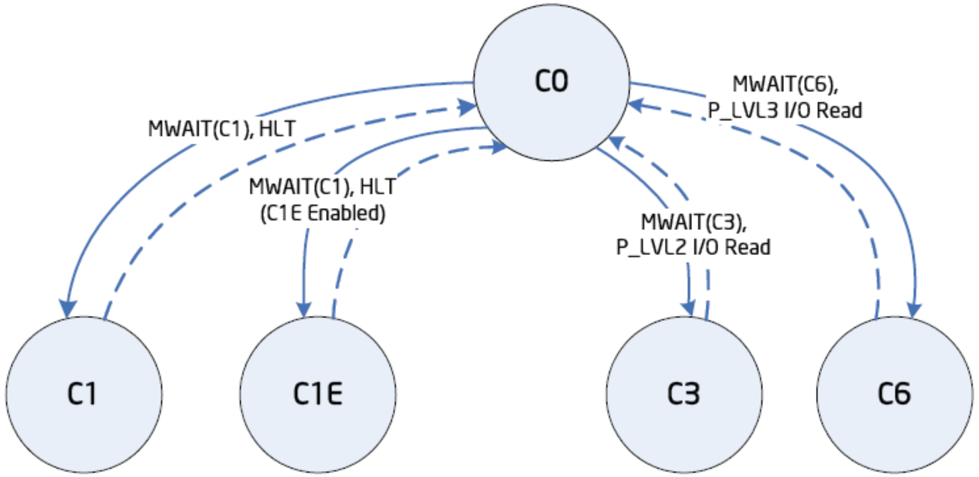
- MWAIT is an assembly language instruction that provides hints to allow the processor to enter an implementation-dependent optimized state.
- MWAIT accepts a hint and optional extension to the processor that it can enter a specified target C state while waiting for an event.
- The MWAIT instruction can be executed only at privilege level 0.

C-State Transitions

Whenever OS sees idling period (more on that later), it will

1. Predict the length of the idle period
2. Choose a C-state based on proposed residency





NOTE:

- HLT or MWAIT instruction triggers the transition to lower power states.
- Legacy method of I/O reads from the ACPI-defined processor clock control registers, referred to as P_LVLx. For legacy operating systems, P_LVLx I/O reads are converted within the processor to the equivalent MWAIT C-state request.
- Interrupts (among others) triggers the transition to C0.

C-state Transition Times

Haswell (EP)

C-state	Exit Latency	Residency
C1	2 us	2 us
C1E	10 us	20 us
C3	33 us	100 us
C6	133 us	400 us
C7(s)	166 us	500 us
C8	300 us	900 us
C9	600 us	1800 us
C10	2600 us	7700 us

Broadwell

C-state	Exit Latency	Residency
C1	2 us	2 us
C1E	10 us	20 us
C3	40 us	100 us
C6	133 us	400 us
C7(s)	166 us	500 us
C8	300 us	900 us
C9	600 us	1800 us
C10	2600 us	7700 us

Skylake

C-state	Exit Latency	Residency
C1	2 us	2 us
C1E	10 us	20 us
C3	70 us	100 us
C6	85 us	200 us
C7(s)	124 us	800 us
C8	200 us	800 us
C9	480 us	5000 us
C10	890 us	5000 us

Note: information given by intel_idle kernel module (3.10 to 4.6 version) -> fake! This information are conservative, they are faster [CSTATE,HSW,SKY]!

Transition times from CX to C0 depend on

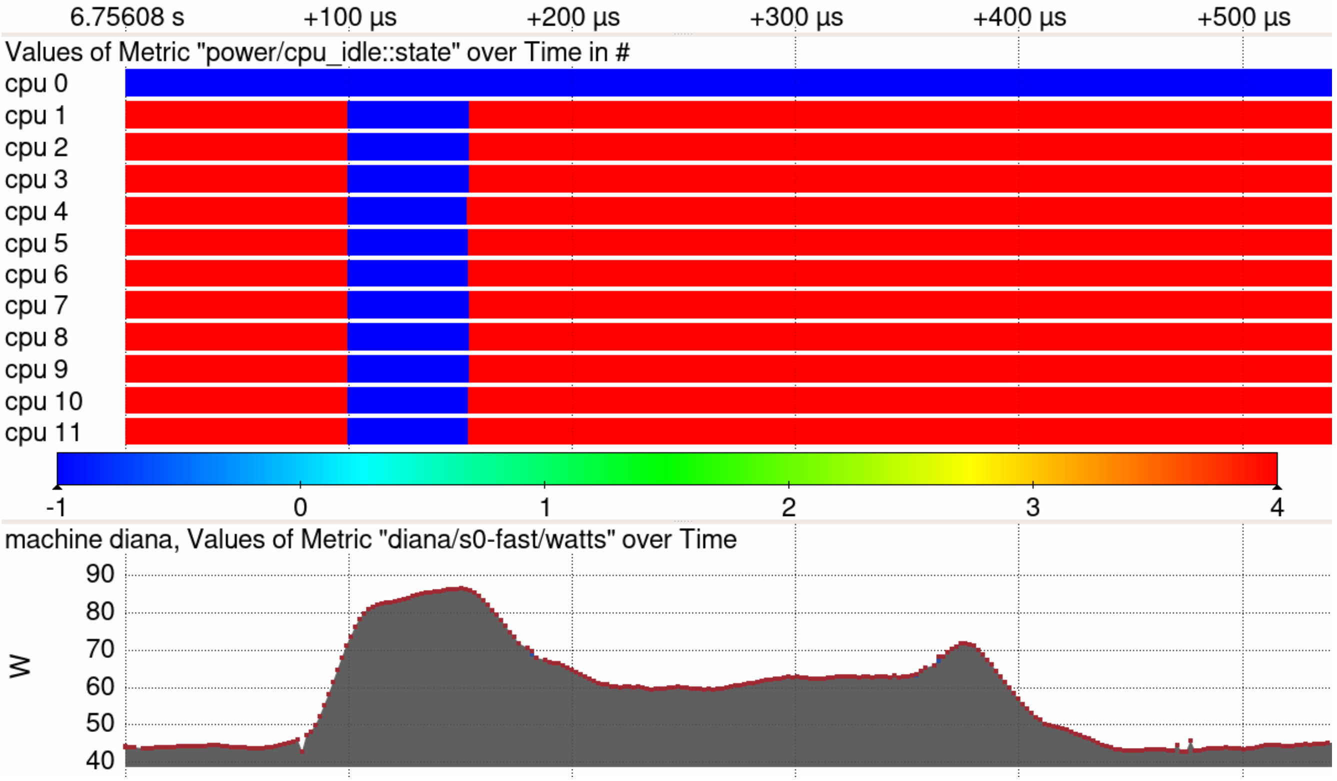
- Initial C-state
  - The lower the C-state, the higher the latency
- Frequency
  - Whether this has influence depends on the (Package) C-State
- How is the core waken up
  - From another core? Where is the other core?
  - By timer?
- Source:

  [CSTATE] Schöne, R., Molka, D., & Werner, M. (2015). Wake-up latencies for processor idle states on current x86 processors. Computer Science-Research and Development, 30(2), 219-227.

  [HSW,SKY]

Transition times from C0 to CX

- Entering a C-state can be delayed by hardware
- Waking up instantaneously after MWAIT can be fast



Source:
[LO2S-C] Ilsche, T., Schöne, R., Joram, P., Bielert, M., & Gocht, A. (2018, May). *System Monitoring with lo2s: Power and Runtime Impact of C-State Transitions.* In 2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW) (pp. 712-715). IEEE.

OS Power Governor

Governors

acpi_cpufreq

- **Performance**: sets the CPU statically to the highest frequency
- **Powersave**: sets the CPU statically to the lowest frequency
- **Userspace**: allows the user, or any userspace program running with UID "root", to set the CPU to a specific frequency by making a sysfs file "scaling_setspeed" available in the CPU-device directory
- **Ondemand**: sets the CPU depending on the current usage. To do this the CPU must have the capability to switch the frequency very quickly
- **Conservative**: much like the "ondemand“ governor, sets the CPU depending on the current usage. It differs in behavior in that it gracefully increases and decreases the CPU speed rather than jumping to max speed the moment there is any load on the CPU

intel_pstate

- **Performance**: always picks the highest frequency, race-to-halt machine or just don't care about energy, very similar to “performance” of cpufreq (only 1% faster)
- **Powersave**: attempts to balance performance with energy savings is completely different than the cpufreq "powersave" governor. The strategy is similar to cpufreq "ondemand", where the requested P-State is related to the system load.

N.B. In Intel CPU the default Linux driver is intel_pstate and the default governor is powersave!

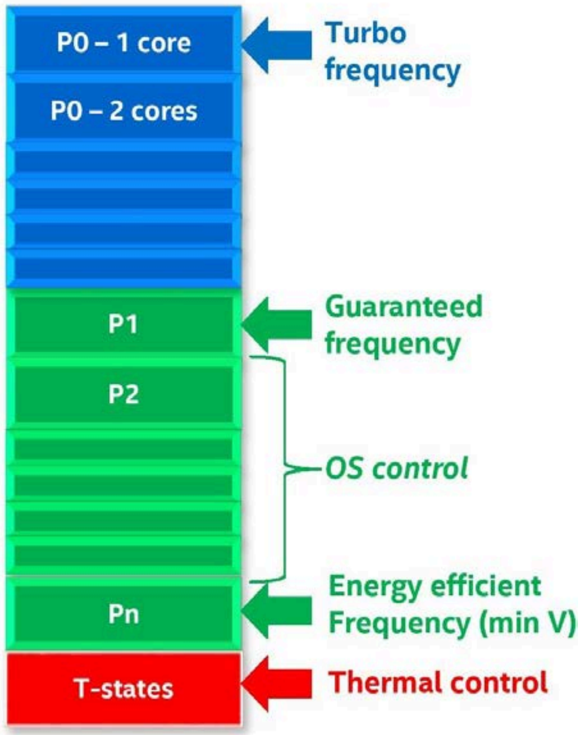
OS Power Governor intel_pstate

- **Active Mode With HWP:** Start with HWP enable and cannot be disabled.
  - **performance:** intel_pstate will write 0 to the processor’s Energy-Performance Preference (EPP) knob (if supported) or its Energy-Performance Bias (EPB) knob (otherwise). This will override the EPP/EPB setting coming from the sysfs interface
  - **powersave**: intel_pstate will set the processor’s Energy-Performance Preference (EPP) knob (if supported) or its Energy-Performance Bias (EPB) knob (otherwise) to whatever value it was previously set to via sysfs (or whatever default value it was set to by the platform firmware). This usually causes the processor’s internal P-state selection logic to be less performance-focused.
- **Active Mode Without HWP:** This is the default operation mode for processors that do not support the HWP feature. It also is used by default with the intel_pstate=no_hwp argument in the kernel command line. However, in this mode intel_pstate may refuse to work with the given processor if it does not recognize it.
  - **performance:** It selects the maximum P-state it is allowed to use, subject to limits set via sysfs, every time the P-state selection computations are carried out by the driver’s utilization update callback for the given CPU (that does not happen more often than every 10 ms), but the hardware configuration will not be changed if the new P-state is the same as the current one.
  - **powersave**: It generally selects P-states proportional to the current CPU utilization, so it is referred to as the “proportional” algorithm.
- **Passive Mode (also called intel_cpufreq):** This mode is used if the intel_pstate=passive argument is passed to the kernel in the command line (it implies the intel_pstate=no_hwp setting too). Like in the active mode without HWP support, in this mode intel_pstate may refuse to work with the given processor if it does not recognize it. It implements the CPUFREQ interface.

Intel P-state driver: intel_pstate

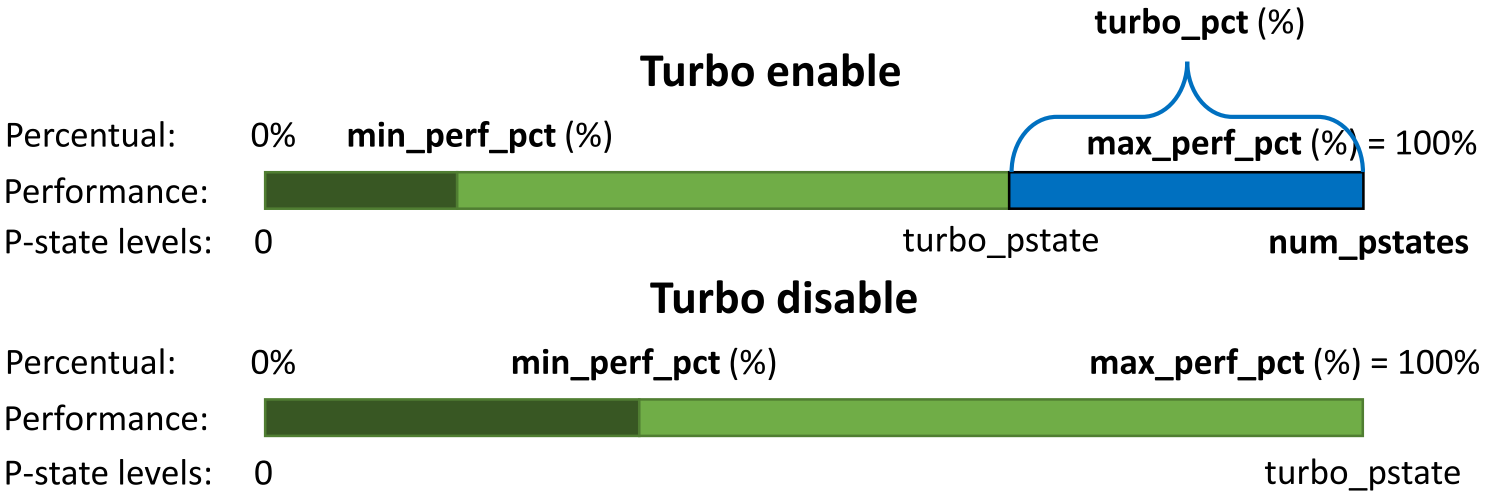
P-state

(All CPUs, plus Skylake)



Fixed, Limited Discrete P-states

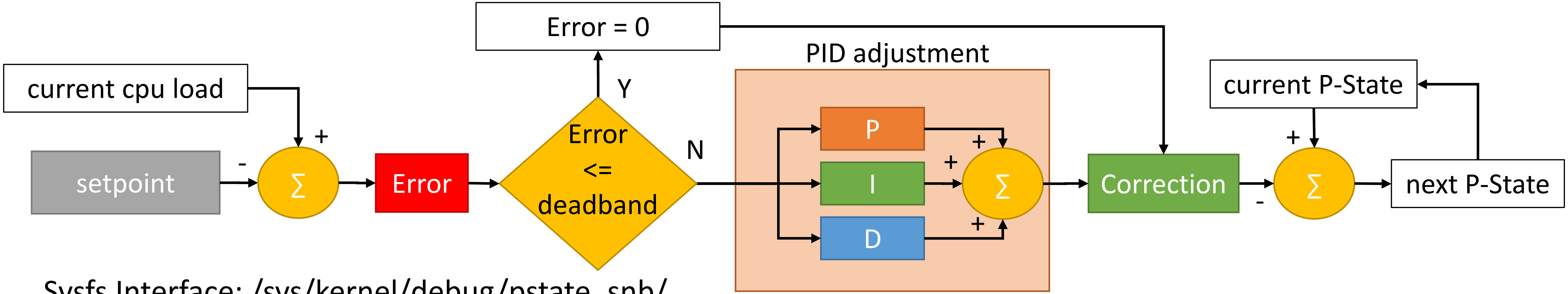
Sysfs Interface: /sys/devices/system/cpu/intel_pstate/*



NOTE:

- Any changes made to these files are applicable to all CPUs (even in a multi-package system).
- In Intel p-state P0 the turbo boost take the responsibility to change the frequency with its own policy.
- T-states existed to save processors from burning themselves up when things went very badly, such as when the cooling fan failed while the processor was running as fast as it could (use private temperature sensors).

Intel P-State Algorithm – PID control



Sysfs Interface: /sys/kernel/debug/pstate_snb/

- **sample_rate_ms**: sample rate in ms of algorithm execution, it can be changed to make the system more responsive to load changes (<u>default 10 ms</u>)
- **setpoint**: is used to make the system use higher performance, even if the load is lower, setpoint can be adjusted to a lower number. This will also lead to faster ramp up time to reach the maximum P-State (<u>default 97</u>)
- **p_gain_pct**: proportional gain coefficient in percentage (<u>default 20</u>)
- **i_gain_pct**: integral gain coefficient in percentage (<u>default 0</u>)
- **d_gain_pct**: derivate gain coefficient in percentage (<u>default 0</u>)
- **deadband**: limit the granularity of error coefficient for PID controller in percentage (<u>default 0</u>)

If there are <u>no derivative and integral coefficients (equal to zero)</u>, the next P-State will be equal to:

next P-State = current P-State - ((setpoint - current cpu load) * p_gain_pct)

Intel P-State – Userspace governor

Unfortunately, Intel P-state driver does not allow userspace governor! But intel_pstate driver is the default one on the most HPC system based on Intel architecture.

So, we developed a hack which is able to inhibit the intel_pstate driver. You can find it in the school folder: **/gpfs/work/cin_powerdam_4/eeschool**

What does the hack?

- Disable the turbo logic of intel_pstate driver
- Set the minimum frequency to the cpufreq interface of intel_pstate
- Set the maximum p-state using MSR
- Set the "energy performance bias hint" with the maximum performance value

How to run:

```
# Switch-off the intel_pstate
$> mpirun -n $SLURM_JOB_NUM_NODES -ppn 1 $WORK/eeschool/off_ipstate.sh

# Switch-on the intel_pstate
$> mpirun -n $SLURM_JOB_NUM_NODES -ppn 1 $WORK/eeschool/on_ipstate.sh
```

Idle governors

The **Intel core C-states** are requested by a power-aware kernel driver:

intel_idle: is a CPU idle driver that supports modern Intel processors. The intel_idle driver presents the kernel with the duration of the target residency and exit latency for each supported Intel processor. The CPU idle menu governor uses this data to predict how long the CPU will be idle.

acpi_idle: implements the BIOS ACPI idle states. For this reason, it defines only four idle states from C0 to C3 and it has no concept of core vs package C-states of Intel processors. Moreover, it is inaccurate in C-state latencies and it has no concept of C-state energy break-even.

Configuration of Idle States: intel_idle vs acpi_idle

Intel_idle driver uses knowledge of the various CPUs to control C-states without input from system firmware (BIOS).
If you want control over C-states, you should use kernel parameters:

intel_idle:

- **intel_idle.max_cstate=0**: this parameter disables the intel_idle, the Linux kernel will use the acpi_idle driver to use C-states. System firmware (BIOS) provides a list of available C-states to the operating system using an ACPI table.
- **intel_idle.max_cstate=n:** Limit the maximum depth of C-state from n = 1 to 9 to specify maximum depth of C-state.
- **Disable C1e**: C1e can be disabled in BIOS setup for low latency. When C1E is enabled, the processor will try to lower processor clock speed and voltage when it enters the C1 C-state, which might result in higher latency. From Linux kernel 3.9, the intel_idle driver treats C1 and C1E as separate states. So, the user can control whether C1E is used without disabling it in BIOS setup.

acpi_idle (intel_idle disable):

- **idle=poll**: If a user wants the <u>absolute minimum latency</u>, kernel parameter "idle_poll" can be used to keep the processors in C0 even when they are idle (the processors will run in a loop when idle, constantly checking to see if they are needed). <u>Hyperthreading should probably be disabled</u>, as keeping processors in C0 can interfere with proper operation of logical cores. It can improve the performance but will <u>use a lot of power and make the system run hot, not recommended</u>.
- **idle=halt**: C-states can also be limited to C1 with the kernel parameter "idle=halt".
- **Idle=nomwait**: Disable mwait instruction for CPU C-states

Dynamic control of C-States

To dynamically control C-states, open the file /dev/cpu_dma_latency and write the maximum allowable latency to it.

- This will prevent C-states with transition latencies higher than the specified value from being used, as long as the file **/dev/cpu_dma_latency** is kept open.

- **cpu_dma_latency=0:** Writing a maximum allowable latency of 0 will keep the processors in C0 (like using kernel parameter "idle=poll")

- **cpu_dma_latency=n**: The value used should correspond to the latency values in /sys/devices/system/cpu/cpuX/cpuidle/stateY/latency (where X is the CPU number and Y is the idle state). This values corresponds to the transition time table in the previous slide.

- CPU idle states that have a greater latency than written to /dev/cpu_dma_latency should not be used.

Trick: One simple way to do this is by compiling a simple program that will write to this file, and stay open until it is killed.

N.B. avoiding CPU to enter in deep C-states limit the turbo frequency!!!

Sysfs for intel_idle (and acpi_idle)

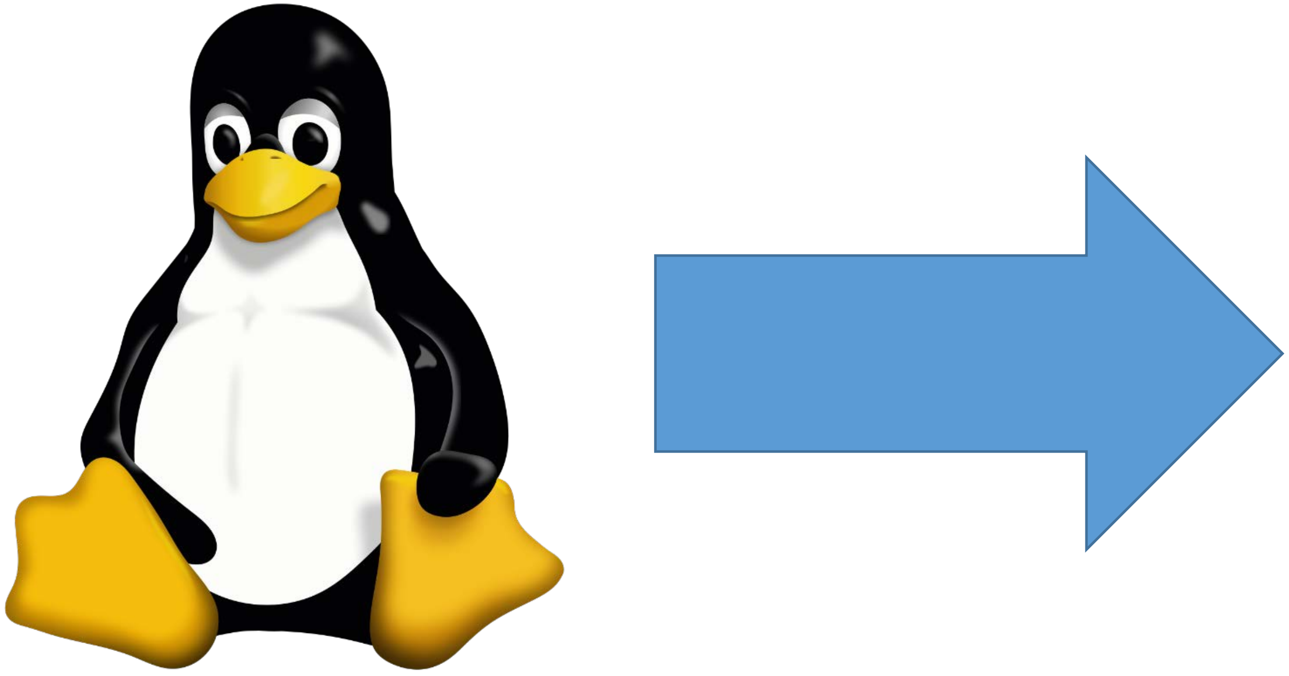
Sysfs Interface: /sys/devices/system/cpu/cpuX/cpuidle/stateY -> Y = idle states are numbered (not level!)

- **name:** Name of the idle state (e.g. C1).
- **desc:** Description of the idle state (or the mwait code).
- **latency:** Exit latency of the idle state in microseconds.
- **power:** Power drawn by hardware in this idle state in milliwatts (if specified, 0 otherwise).
- **residency:** Target residency of the idle state in microseconds.
- **time:** Total time spent in this idle state by the given CPU (as measured by the kernel) in microseconds.
- **usage:** Total number of times the hardware has been asked by the given CPU to enter this idle state.
- **above**: Total number of times this idle state had been asked for, but the observed idle duration was certainly too short to match its target residency.
- **below**: Total number of times this idle state had been asked for, but certainly a deeper idle state would have been a better match for the observed idle duration.
- **disable:** Whether or not this idle state is disabled.

MSR & MSR_SAFE

What are MSRs?

A model-specific register (MSR) is any of various control registers in the x86 instruction set used for debugging, program execution tracing, computer performance monitoring, and toggling certain CPU features.



Standard MSR linux driver

Only root can access

Access to MSRs is Critical

- **Processors provide low-level access to critical information and settings via MSRs**
  - **Power**: package (socket) and dram power
  - **Thermal**: core, package in deg C
  - **Performance Counters**:
    - Effective frequency
    - Instructions retired
- **Enables studies on**
  - Advance performance measurements
  - Power measurements
  - Control for over-provisioned systems

Accessing MSR Data

- **Special assembly instructions in kernel space (ring 0)**
  - rdmsr, wrmsr

- **User level access through MSR kernel module**
  - Provides filesystem interface to all of the MSRs through /dev/cpu/X/msr
  - No finer-grained permissions

- **Access to MSR: rdmsr & wrmsr tool**
  - $> sudo modprobe msr
  - $> sudo rdmsr -p <cpu_id> -d -f <high bit>:<lower bit> <hex_reg>
  - $> sudo wrmsr -p <cpu_id> <hex_reg>

Problem to solve

- **No access/control for regular users in existing interfaces due to:**
  - **Security Concerns**
    - Full access to MSRs could allow you to “root” the machine
    - Pointer to the vector of hardware interrupt handlers is held in an MSR
  - **Permissions**
    - All or nothing access
  - **Complexity in Registers**
    - Error prone

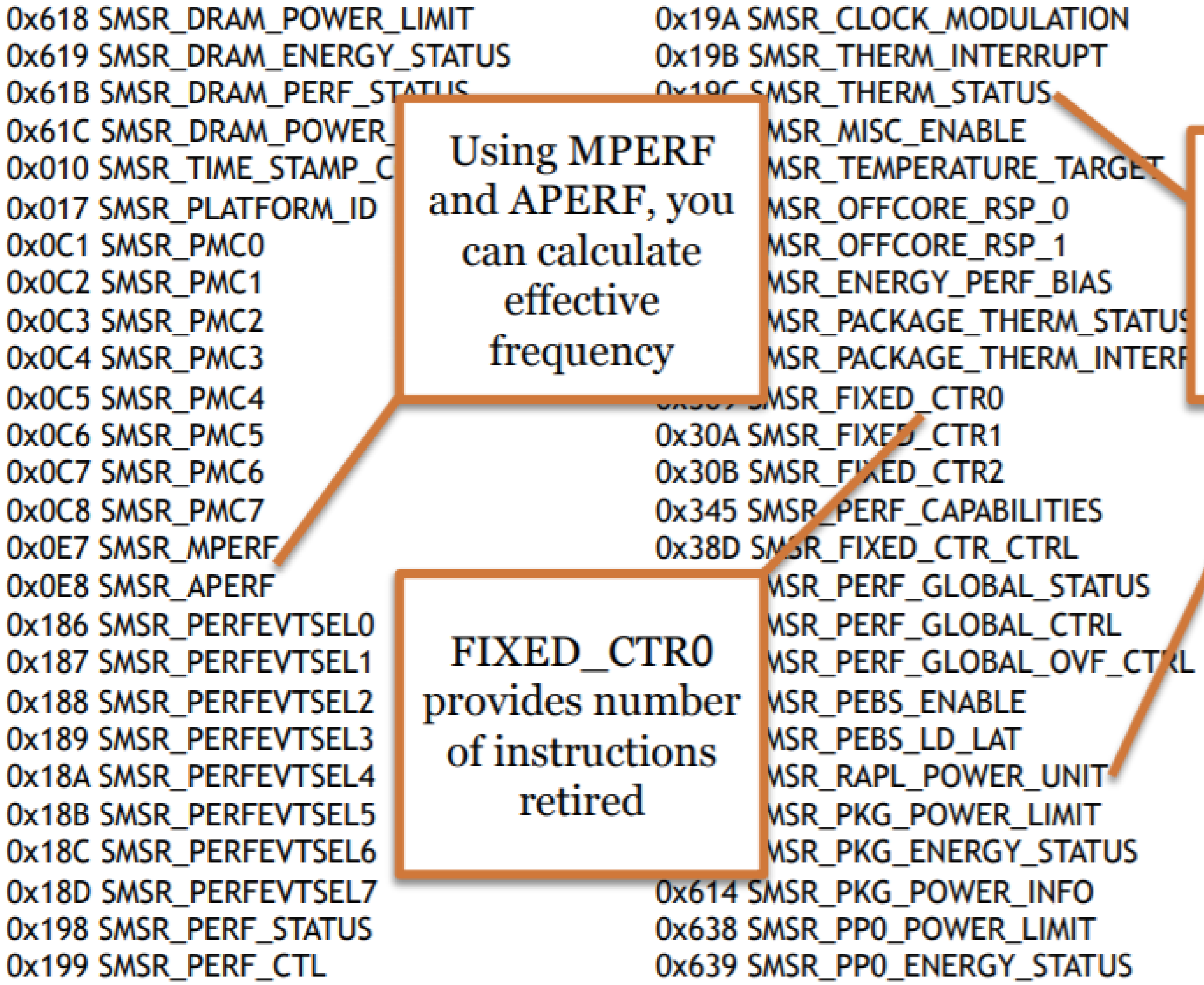
Solution

MSR_SAFE: kernel module developed by LLNL

- **MSR kernel module + file permissions**
  - Only allow “trusted” users to have access with standard linux permissions
  - Access through /dev/cpu/#/msr_safe
  - **msrmod**: equivalent tool of rdmsr and wrmsr but it is based on MSR_SAFE driver
- **Whitelist**
  - Bit level granularity
  - Access to power, thermal, and performance counters/controls
  - Formatted with tables to match Intel manuals (relatively easy to add new registers)
  - Configuration through /dev/cpu/msr_whitelist

https://github.com/LLNL/msr-safe

Example whitelist



```
0x618 SMSR_DRAM_POWER_LIMIT
0x619 SMSR_DRAM_ENERGY_STATUS
0x61B SMSR_DRAM_PERF_STATUS
0x61C SMSR_DRAM_POWER_
0x010 SMSR_TIME_STAMP_C
0x017 SMSR_PLATFORM_ID
0x0C1 SMSR_PMC0
0x0C2 SMSR_PMC1
0x0C3 SMSR_PMC2
0x0C4 SMSR_PMC3
0x0C5 SMSR_PMC4
0x0C6 SMSR_PMC5
0x0C7 SMSR_PMC6
0x0C8 SMSR_PMC7
0x0E7 SMSR_MPERF
0x0E8 SMSR_APERF
0x186 SMSR_PERFEVTSEL0
0x187 SMSR_PERFEVTSEL1
0x188 SMSR_PERFEVTSEL2
0x189 SMSR_PERFEVTSEL3
0x18A SMSR_PERFEVTSEL4
0x18B SMSR_PERFEVTSEL5
0x18C SMSR_PERFEVTSEL6
0x18D SMSR_PERFEVTSEL7
0x198 SMSR_PERF_STATUS
0x199 SMSR_PERF_CTL
```

```
0x19A SMSR_CLOCK_MODULATION
0x19B SMSR_THERM_INTERRUPT
0x19C SMSR_THERM_STATUS
MSR_MISC_ENABLE
MSR_TEMPERATURE_TARGET
MSR_OFFCORE_RSP_0
MSR_OFFCORE_RSP_1
MSR_ENERGY_PERF_BIAS
MSR_PACKAGE_THERM_STATUS
MSR_PACKAGE_THERM_INTERR
0x309 SMSR_FIXED_CTR0
0x30A SMSR_FIXED_CTR1
0x30B SMSR_FIXED_CTR2
0x345 SMSR_PERF_CAPABILITIES
0x38D SMSR_FIXED_CTR_CTRL
MSR_PERF_GLOBAL_STATUS
MSR_PERF_GLOBAL_CTRL
MSR_PERF_GLOBAL_OVF_CTRL
MSR_PEBS_ENABLE
MSR_PEBS_LD_LAT
MSR_RAPL_POWER_UNIT
MSR_PKG_POWER_LIMIT
MSR_PKG_ENERGY_STATUS
0x614 SMSR_PKG_POWER_INFO
0x638 SMSR_PP0_POWER_LIMIT
0x639 SMSR_PP0_ENERGY_STATUS
```

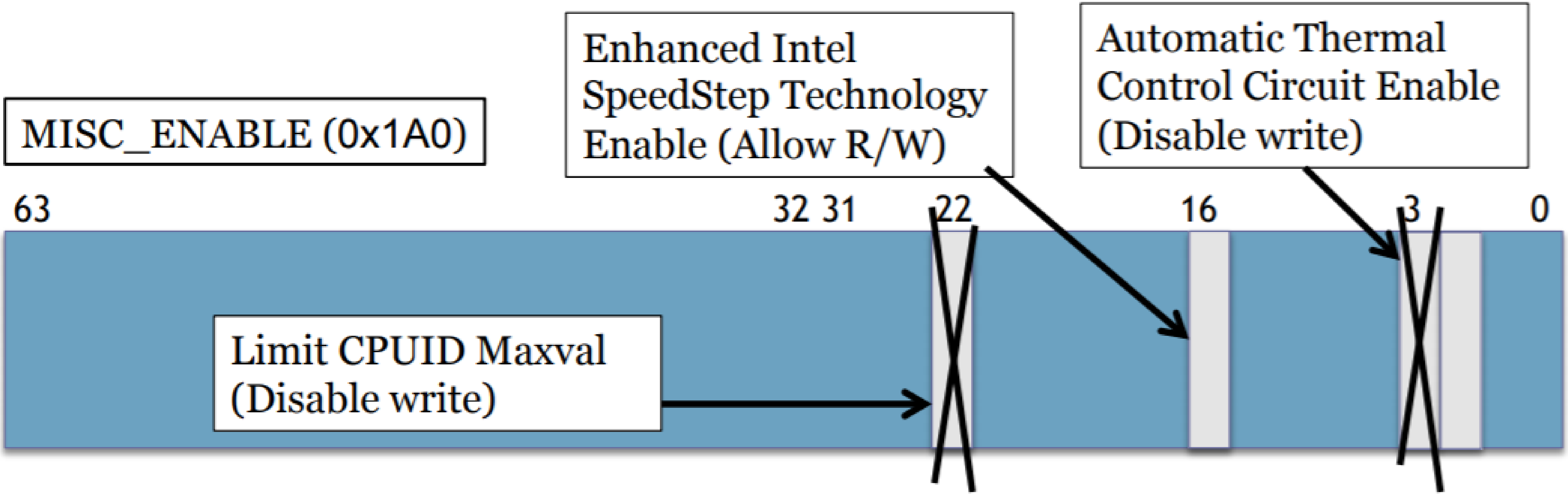
Using MPERF and APERF, you can calculate effective frequency

THERM_STATUS can give thermal information per core

FIXED_CTR0 provides number of instructions retired

Using POWER_UNIT an POWER_LIMIT, you can set power limits on a per package (socket) level

Example whitelist permissions


MISC_ENABLE (0x1A0)
Enhanced Intel SpeedStep Technology Enable (Allow R/W)
Automatic Thermal Control Circuit Enable (Disable write)
63
32 31
22
16
3
0
Limit CPUID Maxval (Disable write)

Convenient access through Libmsr

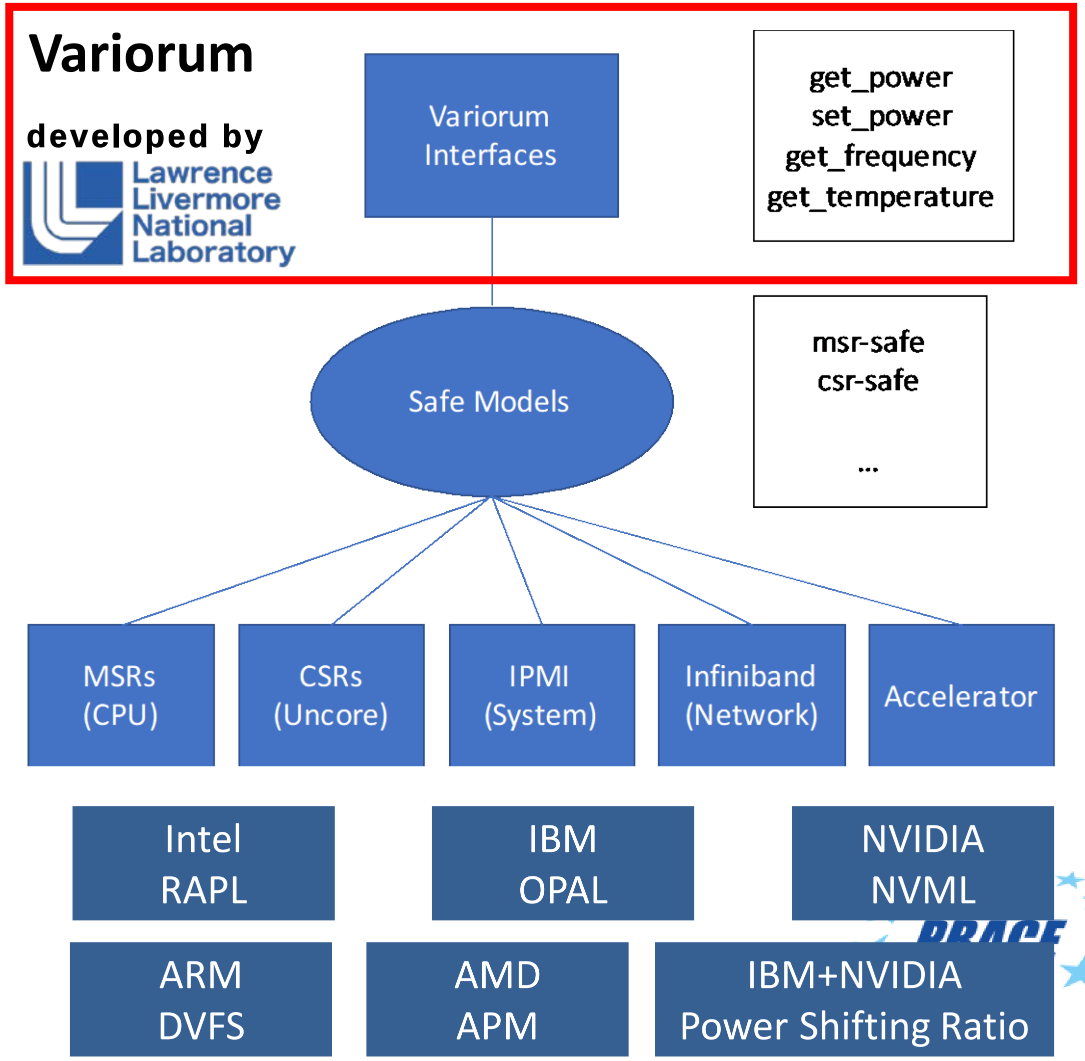
- **Companion library developed at LLNL**
  - Call high level library functions such as:
    - dump_thermal_terse()
    - dump_rapl_limit(…)
  - Build your own with easy to use:
    - Structs
    - Lower level functions
  - The library will do:
    - Error Checking
    - Low Level Work

https://github.com/LLNL/libmsr

Variorum: Vendor-neutral user space library for hardware control knobs

- Platform-agnostic simple front-facing APIs
- Security layer provided to ensure safe, reliable operation
- Batching interfaces minimizing overheads of reading/writing many MSRs
- Production version of `libmsr`, which targeted Intel architectures
- C-based library
- Function pointers to specific implementation for target architecture

https://github.com/LLNL/variorum



Variorum

developed by Lawrence Livermore National Laboratory

Variorum Interfaces

get_power
set_power
get_frequency
get_temperature

Safe Models

msr-safe
csr-safe
...

MSRs (CPU)

CSRs (Uncore)

IPMI (System)

Infiniband (Network)

Accelerator

Intel RAPL

IBM OPAL

NVIDIA NVML

ARM DVFS

AMD APM

IBM+NVIDIA Power Shifting Ratio

References

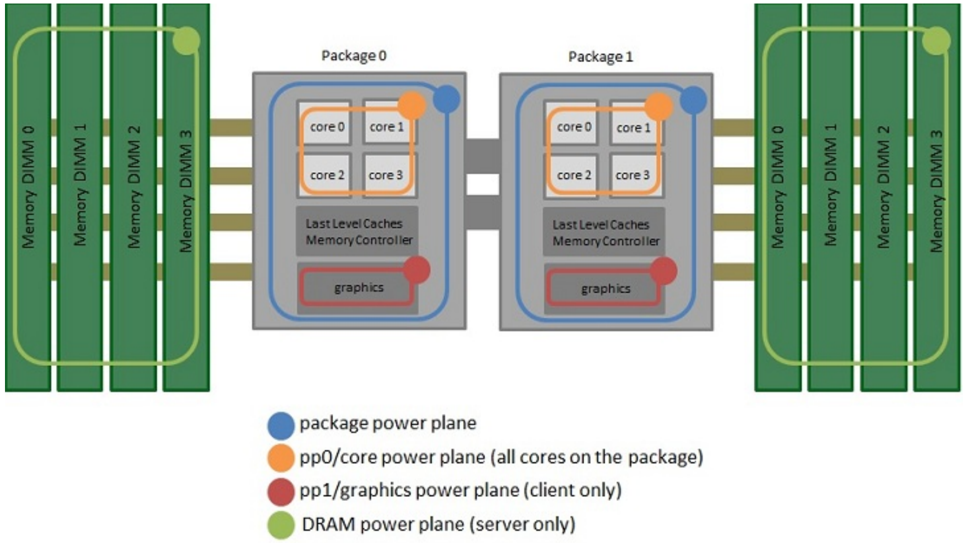
- http://events.linuxfoundation.org/sites/events/files/slides/LinuxConEurope_2015.pdf
- https://www.kernel.org/doc/Documentation/cpu-freq/intel-pstate.txt
- https://people.cs.pitt.edu/~kirk/cs3150spring2010/ShiminChen.pptx
- Intel® 64 and IA-32 Architectures Software Developer's Manual Combined Volumes: 1, 2A, 2B, 2C, 3A, 3B, 3C and 3D
- Intel® Xeon Processor E5 and E7 v3 Family Uncore Performance Monitoring Reference Manual
- Source code of Linux kernel from 3.10 to 4.6 (www.kernel.org)

Follow-up

Running Average Power Limit RAPL

RAPL interfaces provide mechanisms to enforce power consumption limit. RAPL interfaces consist of non-architectural MSRs. RAPL expose multiple domains (power planes) of power rationing within each processor socket. Each RAPL domain supports the following set of capabilities:

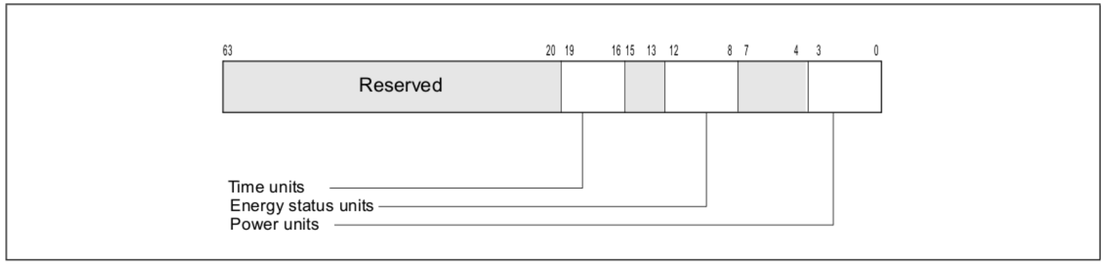
- **Power limit** - MSR interfaces to specify power limit, time window; lock bit, clamp bit etc.
- **Energy Status** - Power metering interface providing energy consumption information.
- **Policy** (Optional) - 4-bit priority information that is a hint to hardware for dividing budget between sub-domains in a parent domain.
- **Perf Status** (Optional) - Interface providing information on the performance effects (regression) due to power limits. It is defined as a duration metric that measures the power limit effect in the respective domain. The meaning of duration is domain specific.
- **Power Info** (Optional) - Interface providing information on the range of parameters for a given domain, minimum power, maximum power etc.



Each of the above capabilities requires specific units in order to describe them. Power is expressed in Watts, Time is expressed in Seconds, and Energy is expressed in Joules. Scaling factors are supplied to each unit to make the information presented meaningful in a finite number of bits.

RAPL – Time Units and Domains

Units for power, energy, and time are exposed in the read-only **MSR_RAPL_POWER_UNIT** (0x606) MSR.



- **Power Units** (bits 3:0): Power related information (in Watts) is based on the multiplier, **1/2^PU**; where PU is an unsigned integer represented by bits 3:0. Default value is 0011b, indicating power unit is in 1/8 Watts increment.
- **Energy Status Units** (bits 12:8): Energy related information (in Joules) is based on the multiplier, **1/2^ESU**; where ESU is an unsigned integer represented by bits 12:8. Default value is 10000b, indicating energy status unit is in 15.3 micro-Joules increment.
- **Time Units** (bits 19:16): Time related information (in Seconds) is based on the multiplier, **1/2^TU**; where TU is an unsigned integer represented by bits 19:16. Default value is 1010b, indicating time unit is in 976 microseconds increment.

RAPL support the following RAPL domain hierarchy: entire package (PKG), DRAM, power plane for cores (PP0) and power plane for uncore graphic device (PP1). Each level of the RAPL hierarchy provides a respective set of RAPL interface MSRs.

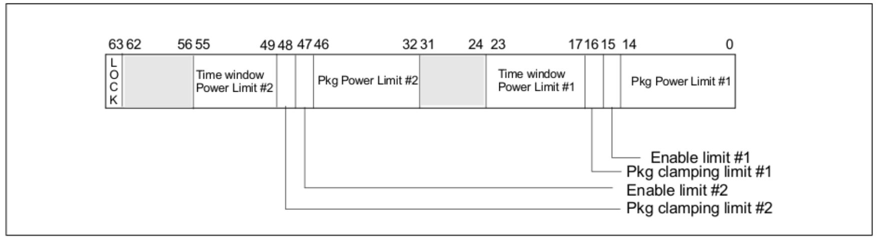
Domain	Power Limit (Offset 0)	Energy Status (Offset 1)	Policy (Offset 2)	Perf Status (Offset 3)	Power Info (Offset 4)
PKG	MSR_PKG_POWER_LIMIT	MSR_PKG_ENERGY_STATUS	RESERVED	MSR_PKG_PERF_STATUS	MSR_PKG_POWER_INFO
DRAM	MSR_DRAM_POWER_LIMIT	MSR_DRAM_ENERGY_STATUS	RESERVED	MSR_DRAM_PERF_STATUS	MSR_DRAM_POWER_INFO
PP0	MSR_PP0_POWER_LIMIT	MSR_PP0_ENERGY_STATUS	MSR_PP0_POLICY	MSR_PP0_PERF_STATUS	RESERVED
PP1	MSR_PP1_POWER_LIMIT	MSR_PP1_ENERGY_STATUS	MSR_PP1_POLICY	RESERVED	RESERVED

NOTE: PP1 not present in server architectures
DRAM present only in server architectures

RAPL – Package Domain

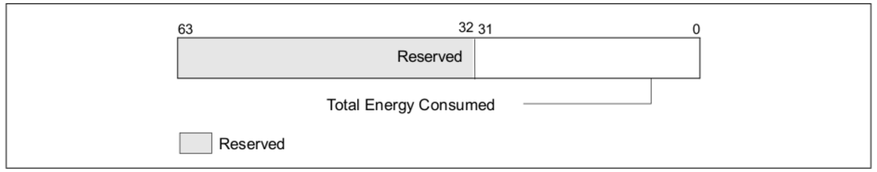
MSR_PKG_POWER_LIMIT allows a software agent to define power limitation for the package domain. Power limitation is defined in terms of average power usage (Watts) over a time window. Two power limits can be specified, corresponding to time windows of different sizes. Each power limit provides independent clamping control that would permit the processor cores to go below OS-requested state to meet the power limits. A lock mechanism allow the software agent to enforce power limit settings. Once the lock bit is set, the power limit settings are static and un-modifiable until next RESET.

- **Package Power Limit**: Sets the average power usage limit of the package domain corresponding to related time window. The unit of this field is specified by the “Power Units” field of MSR_RAPL_POWER_UNIT
- **Enable Power Limit**: 0 = disabled; 1 = enabled
- **Package Clamping Limitation**: Allow going below OS-requested P/T state setting during time window.
- **Time Window for Power Limit**: Indicates the time window for power limit.
  Time limit = 2^Y * (1.0 + Z/4.0) * Time Unit.
  Here “Y” is the unsigned integer value represented by bits 21:17 – 53:49, “Z” is an unsigned integer represented by bits 23:22 – 55:54. “Time Unit” is specified by the “Time Units” field of MSR_RAPL_POWER_UNIT.
- **Lock**: If set, all write attempts to this MSR are ignored until next RESET.



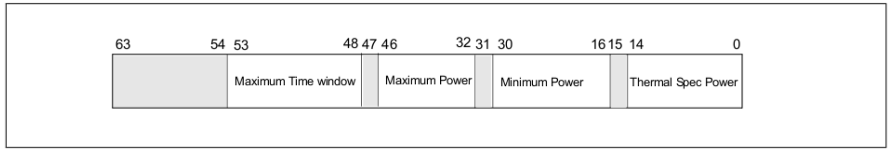
MSR_PKG_ENERGY_STATUS is a read-only MSR. It reports the actual energy use for the package domain. This MSR is updated every ~1msec. It has a wraparound time of around 60 secs when power consumption is high, and may be longer otherwise.

- **Total Energy Consumed**: The unsigned integer value represents the total amount of energy consumed since that last time this register is cleared. The unit of this field is specified by the “Energy Status Units” field of MSR_RAPL_POWER_UNIT.



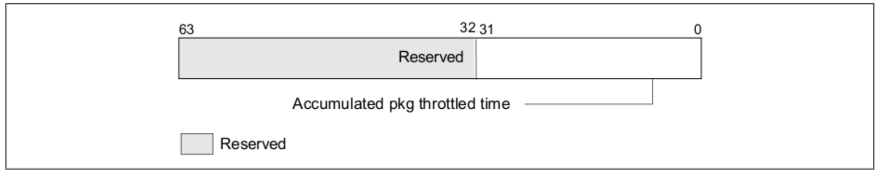
MSR_PKG_POWER_INFO is a read-only MSR. It reports the package power range information for RAPL usage. This MSR provides maximum/minimum values (derived from electrical specification), thermal specification power of the package domain. It also provides the largest possible time window for software to program the RAPL interface.

- **Thermal Spec Power**: The unsigned integer value is the equivalent of thermal specification power of the package domain. The unit of this field is specified by the “Power Units” field of MSR_RAPL_POWER_UNIT.
- **Minimum Power**: The unsigned integer value is the equivalent of minimum power derived from electrical spec of the package domain. The unit of this field is specified by the “Power Units” field of MSR_RAPL_POWER_UNIT.
- **Maximum Power**: The unsigned integer value is the equivalent of maximum power derived from the electrical spec of the package domain. The unit of this field is specified by the “Power Units” field of MSR_RAPL_POWER_UNIT.
- **Maximum Time Window**: The unsigned integer value is the equivalent of largest acceptable value to program the time window of MSR_PKG_POWER_LIMIT. The unit of this field is specified by the “Time Units” field of MSR_RAPL_POWER_UNIT.



MSR_PKG_PERF_STATUS is a read-only MSR. It reports the total time for which the package was throttled due to the RAPL power limits. Throttling in this context is defined as going below the OS-requested P-state or T-state. It has a wrap-around time of many hours.

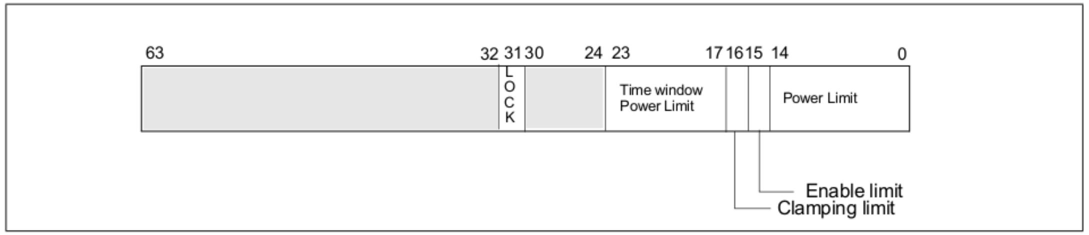
- **Accumulated Package Throttled Time**: The unsigned integer value represents the cumulative time (since the last time this register is cleared) that the package has throttled. The unit of this field is specified by the “Time Units” field of MSR_RAPL_POWER_UNIT.



RAPL – PP0/PP1 Domain

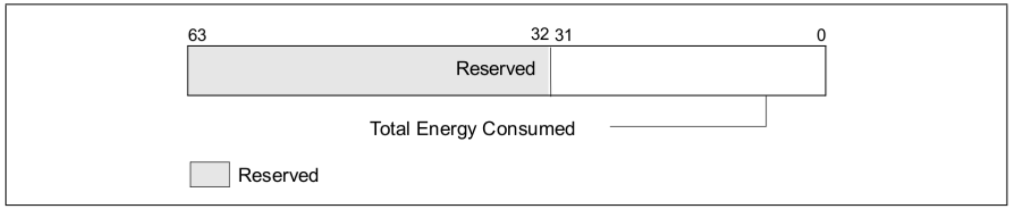
MSR_PP0_POWER_LIMIT/MSR_PP1_POWER_LIMIT allow a software agent to define power limitation for the respective power plane domain. A lock mechanism in each power plane domain allows the software agent to enforce power limit settings independently. Once a lock bit is set, the power limit settings in that power plane are static and un-modifiable until next RESET.

- **Power Limit**: Sets the average power usage limit of the respective power plane domain. The unit of this field is specified by the “Power Units” field of MSR_RAPL_POWER_UNIT.
- **Enable Power Limit**: 0 = disabled; 1 = enabled.
- **Clamping Limitation**: Allow going below OS-requested P/T state setting during time window.
- **Time Window for Power Limit**: Indicates the length of time window over which the power limit will be used by the processor. The numeric value encoded by bits 23:17 is represented by the product of 2^Y *F; where F is a single-digit decimal floating-point value between 1.0 and 1.3 with the fraction digit represented by bits 23:22, Y is an unsigned integer represented by bits 21:17. The unit of this field is specified by the “Time Units” field of MSR_RAPL_POWER_UNIT.
- **Lock**: If set, all write attempts to the MSR and corresponding policy MSR_PP0_POLICY/MSR_PP1_POLICY are ignored until next RESET.



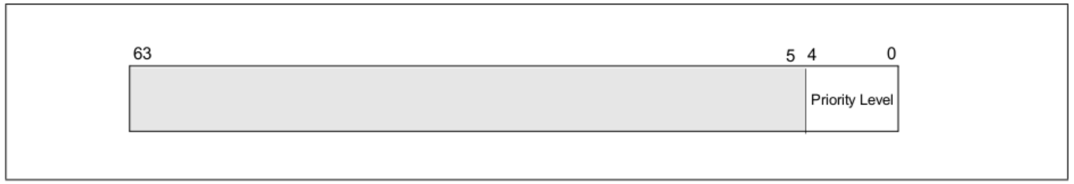
MSR_PP0_ENERGY_STATUS/MSR_PP1_ENERGY_STATUS are read-only MSRs. They report the actual energy use for the respective power plane domains. These MSRs are updated every ~1msec.

- **Total Energy Consumed**: The unsigned integer value represents the total amount of energy consumed since the last time this register was cleared. The unit of this field is specified by the “Energy Status Units” field of MSR_RAPL_POWER_UNIT.



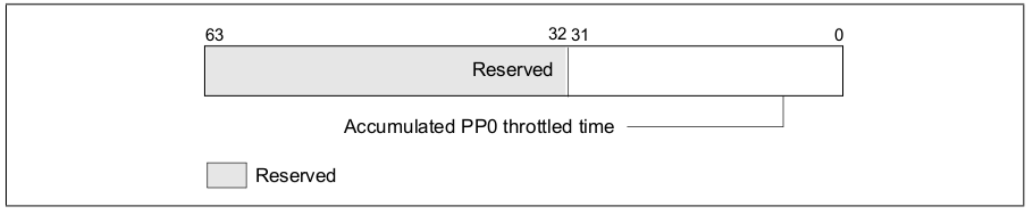
MSR_PP0_POLICY/MSR_PP1_POLICY provide balance power policy control for each power plane by providing inputs to the power budgeting management algorithm. On platforms that support PP0 (IA cores) and PP1 (uncore graphic device), the default values give priority to the non-IA power plane. These MSRs enable the PCU to balance power consumption between the IA cores and uncore graphic device.

- **Priority Level**: Priority level input to the PCU for respective power plane. PP0 covers the IA processor cores, PP1 covers the uncore graphic device. The value 31 is considered highest priority.



MSR_PP0_PERF_STATUS is a read-only MSR. It reports the total time for which the PP0 domain was throttled due to the power limits. This MSR is supported only in server platform. Throttling in this context is defined as going below the OS-requested P-state or T-state.

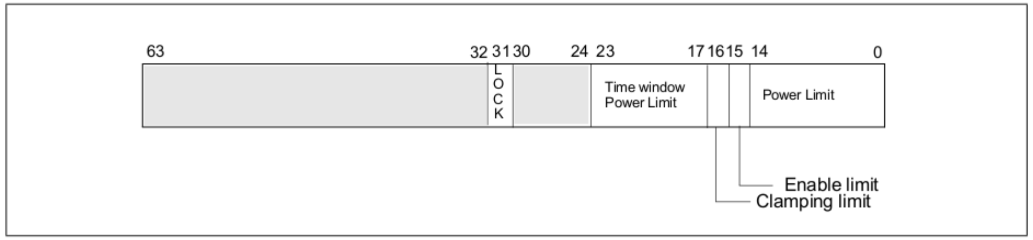
- **Accumulated PP0 Throttled Time**: The unsigned integer value represents the cumulative time (since the last time this register is cleared) that the PP0 domain has throttled. The unit of this field is specified by the “Time Units” field of MSR_RAPL_POWER_UNIT.



RAPL – DRAM Domain

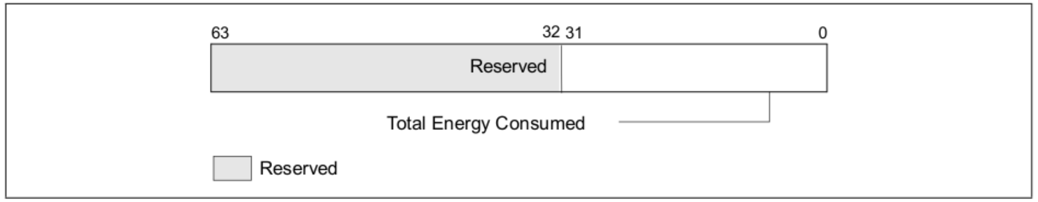
MSR_DRAM_POWER_LIMIT allows a software agent to define power limitation for the DRAM domain. Power limitation is defined in terms of average power usage (Watts). A power limit can be specified along with a time window. A lock mechanism allow the software agent to enforce power limit settings. Once the lock bit is set, the power limit settings are static and un modifiable until next RESET.

- **DRAM Power Limit**: Sets the average power usage limit of the DRAM domain corresponding to time window. The unit of this field is specified by the “Power Units” field of MSR_RAPL_POWER_UNIT.
- **Enable Power Limit**: 0 = disabled; 1 = enabled.
- **Time Window for Power Limit**: Indicates the length of time window over which the power limit will be used by the processor. The numeric value encoded by bits 23:17 is represented by the product of 2^Y *F; where F is a single-digit decimal floating-point value between 1.0 and 1.3 with the fraction digit represented by bits 23:22, Y is an unsigned integer represented by bits 21:17. The unit of this field is specified by the “Time Units” field of MSR_RAPL_POWER_UNIT.
- **Lock**: If set, all write attempts to this MSR are ignored until next RESET.



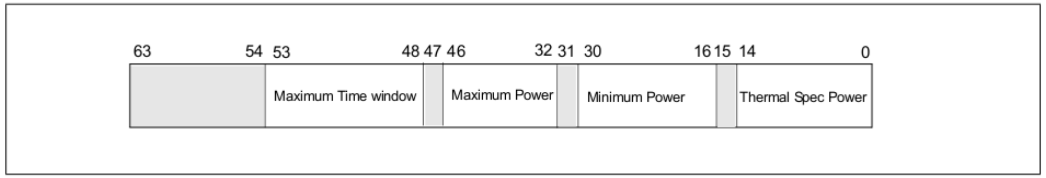
MSR_DRAM_ENERGY_STATUS is a read-only MSR. It reports the actual energy use for the DRAM domain. This MSR is updated every ~1msec.

- **Total Energy Consumed**: The unsigned integer value represents the total amount of energy consumed since that last time this register is cleared. The unit of this field is specified by the “Energy Status Units” field of MSR_RAPL_POWER_UNIT.



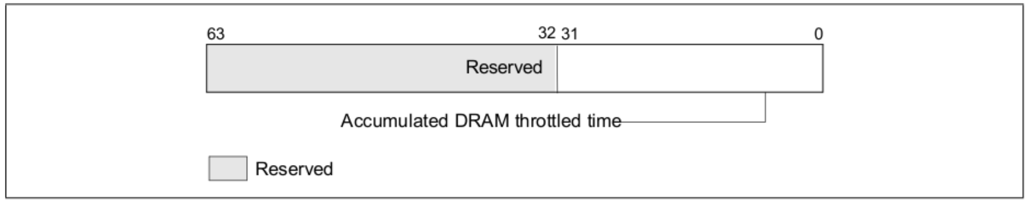
MSR_DRAM_POWER_INFO is a read-only MSR. It reports the DRAM power range information for RAPL usage. This MSR provides maximum/minimum values (derived from electrical specification), thermal specification power of the DRAM domain. It also provides the largest possible time window for software to program the RAPL interface.

- **Thermal Spec Power**: The unsigned integer value is the equivalent of thermal specification power of the DRAM domain. The unit of this field is specified by the “Power Units” field of MSR_RAPL_POWER_UNIT.
- **Minimum Power**: The unsigned integer value is the equivalent of minimum power derived from electrical spec of the DRAM domain. The unit of this field is specified by the “Power Units” field of MSR_RAPL_POWER_UNIT.
- **Maximum Power**: The unsigned integer value is the equivalent of maximum power derived from the electrical spec of the DRAM domain. The unit of this field is specified by the “Power Units” field of MSR_RAPL_POWER_UNIT.
- **Maximum Time Window**: The unsigned integer value is the equivalent of largest acceptable value to program the time window of MSR_DRAM_POWER_LIMIT. The unit of this field is specified by the “Time Units” field of MSR_RAPL_POWER_UNIT.



MSR_DRAM_PERF_STATUS is a read-only MSR. It reports the total time for which the package was throttled due to the RAPL power limits. Throttling in this context is defined as going below the OS-requested P-state or T-state. It has a wrap-around time of many hours.

- **Accumulated Package Throttled Time**: The unsigned integer value represents the cumulative time (since the last time this register is cleared) that the DRAM domain has throttled. The unit of this field is specified by the “Time Units” field of MSR_RAPL_POWER_UNIT.


63
32 31
0
Reserved
Accumulated DRAM throttled time
Reserved