Technical Paper

Network Edge Group (NEX)

Empowering Mixed-criticality Industrial Real-time Computing on Performance Hybrid Architecture with Intel's Dynamic Frequency Scaling Evolution

Intel's Dynamic Frequency Scaling evolution, starting from 11th Generation Intel® Core™ processors, is unleashing new opportunities for mixed-criticality industrial real-time computing. It emphasizes its significance for efficient power management in this dynamic landscape.

Authors

Markus Schweikhardt

Intel Corporation

Alexander Slota

Intel Corporation

Table of Contents

Abstract

Building upon our prior research investigating Intel's Dynamic Frequency Scaling (DFS) technology within the realm of mixed-criticality industrial real-time computing[1], this article serves as an essential follow-up. Our comprehensive investigation delves into Intel's optimization in the area of DFS capabilities across the 12th and 13th generation Intel® Core™ processors, providing valuable insights into the Intel performance hybrid core architectures, all while maintaining a special focus on accommodating mixed-criticality real-time scenarios with hard real-time requirements.

In this article, we thoroughly explore the transformative potential of DFS within the context of empowering mixed-criticality workloads. Based on real-world industrial use cases, such as process

[1] "Dynamic Frequency Scaling for Mixed Criticality Real-Time Scenarios on 11th Generation Intel® Core™ Processors" - RDC Document number #723446

automation and motion control, we provide a comprehensive analysis of the implications, benefits, and critical considerations associated with the efficient management of power in this dynamic and demanding computational landscape. By shedding light on the advancements and nuances of Intel's DFS evolution on the Intel performance hybrid architecture-based platforms, this research aims to contribute to the optimization of industrial real-time computing systems and their ability to meet the ever-increasing demands of mixed-criticality system designs.

Introduction

Navigating the Real-time Landscape in Hybrid Microarchitectures

In this section, we embark on a journey to understand the distinctive challenges and opportunities brought about by hybrid microarchitectures, particularly in the realm of real-time computing. As we dive deeper into this subject, it's paramount to remember the core objective of real-time systems: bounding the worst-case execution time (WCET). We will not only examine historical aspects but also spotlight the specific challenges in terms of DFS that arise when working with performance (P) and efficient (E) cores.

Real-Time Challenges

Real-time computing finds its roots in the world of single-core systems, where the main objective was to tame the unpredictable nature of code execution. This unpredictability stemmed from variables such as branching, cache behavior, and external factors like clock frequency and power states. Furthermore, the introduction of integrated GPUs alongside CPU cores added complexity due to shared resources and potential resource contention.

The transition to multicore processors compounded these challenges. Now, multiple cores share vital resources like caches, memory controllers, buses, and I/O systems. The difficulties that had existed in single-core systems remain relevant in the multicore era, regardless of whether the workload was assigned to a P- or E-Core. It's crucial to emphasize that many challenges in hybrid microarchitectures have their origins in these historical issues of real-time computing.

Design Philosophy and Evolution

In the area of Intel® Core™ processors, Intel has consistently pursued a design philosophy that prioritizes high-performance computing. Cores within this architecture are optimized for tasks demanding substantial processing power, making them ideal for resource-intensive computations. On the other end of the spectrum, the Intel® Atom™ architecture reflects a different design philosophy, emphasizing power efficiency and scalability for low-power devices. Intel® Atom™ processors are tailored for mobile devices, Internet of Things (IoT) applications, and other scenarios where energy efficiency is a key requirement. The evolution of Intel Atom® processors has seen a focus on minimizing power consumption while maintaining adequate performance for lightweight computing tasks. The 12th Generation Intel® Core™ processors marked the realization of a long-envisioned concept and brought both design philosophies together. The hybrid architecture integrates P-Cores with E-Cores, combining the strengths of both Intel® Core™ and Intel Atom® designs. This approach aims to find a balance between performance and energy efficiency to meet the diverse workloads in modern computing environments.

Now that we have established the historical context and foundation for hybrid microarchitectures, we will shift our focus to the specific considerations when dealing with DFS and real-time workloads on Intel® performance hybrid architecture. In the upcoming sections, we will begin by examining the high-level architecture within the context of DFS and mixed-criticality real-time workloads. We will then delve into new use cases that arise as a result of the DFS feature enhancements. Finally, we will present some initial measurement results to demonstrate the practicality of these use cases.

Dynamic Frequency Scaling on Performance Hybrid Architecture

Architectural Considerations of 12th and 13th Generation Intel® Core™ Processors

For the hybrid architecture design, Intel has been incorporating one Phase-Locked Loop (PLL) for each module. This design philosophy is tailored to accommodate two primary core types: P-Cores and E-Cores, each serving a specific purpose – see .

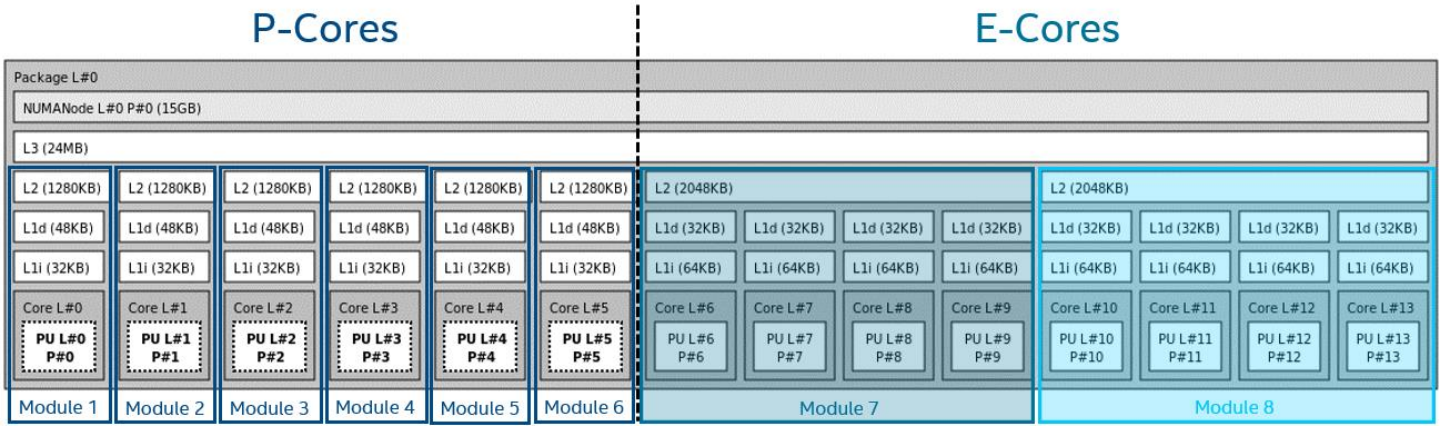
For the P-Cores, Intel's architecture for 12th and 13th generation Intel® Core™ processors assigns a single core per module, ensuring that the p-states of each core can be managed independently. This level of control allows for fine-tuning the performance of individual cores to match the demands of specific workloads, ultimately optimizing the overall system's performance capabilities.

Conversely, when it comes to E-Cores, Intel has chosen to group them in sets of four cores per module for 12th and 13th generation Intel® Core™ processors. These E-Cores are managed collectively within each module. This grouping strategy optimizes power and performance adjustments for the E-Cores, striking a balance between delivering robust performance and conserving energy and space.

Both core types use digital PLLs which allow rapid p-state transition times, typically within the nanosecond range. This means that the hybrid architecture can dynamically and efficiently adjust the frequency of each module, ensuring that it operates at peak performance or efficiency for any given task.

In summary, the hybrid architecture, with its per-core and per-module p-state control, along with the fast digital PLLs, is primed to deliver the best of both worlds: high performance when needed and energy efficiency when it matters most. This level of adaptability and precision enhances the overall performance and responsiveness of the system, making it a versatile solution for various computing needs.

Figure 1. High-level Architecture of 13th Generation Intel® Core™ i7-1370PE Processors



Sidenote - In addition to the DFS features, another crucial distinction between P-Cores and E-Cores of 12th and 13th generation Intel® Core™ processor that should be taken into account when consolidating mixed-critical real-time workloads is the cache architecture – see . For the example depicted in dL3 cache is shared among all cores, P-Cores have an individual L2 cache for each core, whereas E-Cores have a share L2 cache per module – more details like the cache topology of different processor generations can be found in the Intel® Time Coordinated Compute User Guide[2].

[2] "Intel Time Coordinated Compute User Guide" – RDC ID #786715

Hardware Controlled Performance States (HWP)

The 12th and 13th generation Intel processors come equipped with a feature known as HWP, or Intel® Speed Shift technology. In contrast to the Enhanced Intel SpeedStep® technology, where the operating system plays a central role in controlling and monitoring discrete frequency-based operating points with HWP activated, the processor takes on the responsibility of independently choosing the most suitable performance states based on the workload demand. It does so while considering certain guiding hints programmed by the operating system. These hints encompass factors like setting minimum and maximum performance thresholds, indicating a preference for energy efficiency or performance, and defining a specific time frame for observing the history of workload patterns.

The operating system is also equipped with the capability to override HWP's autonomous selection of performance states and assign a desired performance target. However, the actual frequency delivered is still subject to the processor's energy efficiency and performance optimizations. In essence, HWP strikes a dynamic balance between autonomous performance state selection and OS-driven performance directives.

To make use of HWP/Intel® Speed Shift Technology, it is essential to enable it in the BIOS. Software can detect HWP support using the CPUID instruction. The most important model-specific registers (MSR) of the HWP interface are shown below.

Address	Register Name	Description
0x770	`IA32_PM_ENABLE`	Enable / Disable HWP
0x771	`IA32_HWP_CAPABILITIES`	Lists HWP performance range
0x774	`IA32_HWP_REQUEST`	Used by OS to provide hints (min, max, desired, energy performance preferences to HWP)

For a more comprehensive and detailed explanation of the HWP programming interface, you can refer to the Intel® Software Developer's Manual[3].

HWP - Energy Performance Preferences (EPP)

The EPP value is a bit field within the IA32_HWP_REQUEST MSR, and the OS can use it to bias a performance request for a specific core towards either energy efficiency or performance. In practical terms, for cores optimized for performance with an EPP close to zero, HWP consistently works to minimize core utilization, resulting in a high core frequency even with low core utilization. Conversely, for cores optimized for energy efficiency with an EPP closer to 255, HWP aims to maximize core utilization, effectively running the core at the lowest performance required. The exact mapping of an EPP value to the core utilization could change from generation to generation.

In scenarios involving mixed-criticality real-time workloads, it's advisable to fine-tune the EPP value of real-time (RT) cores towards performance and best-effort (BE) cores more towards energy efficiency. It's important to note that with "HWP Autonomous EPP Grouping" enabled in the BIOS, all cores with the same EPP value follow the maximum performance request of the group. If disabled, each core is treated independently. Additionally, it is important to keep in mind that EPP is a crucial parameter for power balancing between cores, cores biased towards performance have a higher priority compared to cores biased towards energy efficiency.

[3] Intel® 64 and IA-32 Architectures Software Developer's Manual – Volume 3, Section 15.4 Rev. March 2023

CPU Performance Scaling in Linux* OS

Instead of directly using the HWP MSRs, the Linux* kernel features a subsystem designed for managing CPU performance scaling known as CPUFreq, comprising three key components: the core, scaling governor, and scaling driver.

The CPUFreq[4] core serves as the foundational infrastructure and user interface.

Scaling governors are responsible for executing algorithms that assess the necessary CPU capacity.

Scaling drivers, on the other hand, facilitate communication with the hardware, provide information about hardware capabilities to governors and interact with platform-specific hardware interfaces to make p-state adjustments as chosen by the scaling governors.

The intel_pstate scaling driver has two modes of operation: active and passive. In the active mode, the driver bypasses the scaling governor and uses its own internal performance scaling governor algorithm, or it allows the hardware to handle performance scaling by itself if HWP is supported. If HWP is enabled, the following scaling governors are available:

HWP + Performance	Intel P-state logical will focus on performance - EPP set to 0.
HWP + Powersave	Intel P-State logical will focus more on energy efficency – EPP set to default value.

The following policy attributes can be used to provide performance hints to the intel_pstate driver:

- `/sys/devices/system/cpu/cpu<n>/cpufreq/scaling_max_freq`
- `/sys/devices/system/cpu/cpu<n>/cpufreq/scaling_min_freq`
- `/sys/devices/system/cpu/cpu<n>/cpufreq/scaling_governor`

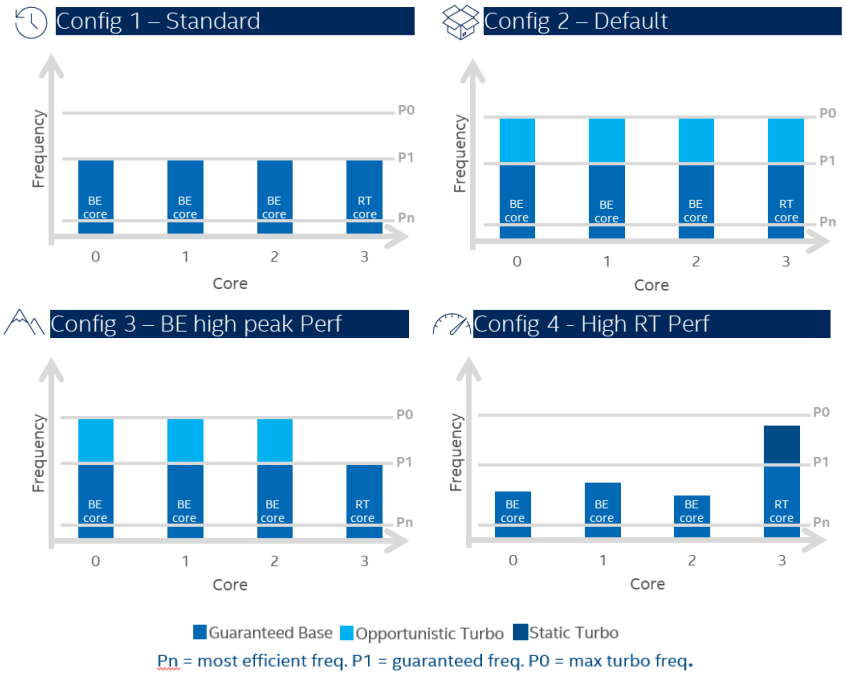
For more information, please refer to intel_pstate CPU Performance Scaling Driver in the Linux kernel documentation[5].

Dynamic Frequency Scaling Configurations

In the earlier sections, we delved into the improvements in DFS features and the essential considerations when working with hybrid architecture platforms. Now, let's explore how this newfound flexibility can be harnessed within the context of consolidating mixed-criticality workloads. showcases examples of four distinct sets of configurations: standard or state of the art, default or out of box, high peak performance for BE and high performance for RT workloads.

[4] https://www.kernel.org/doc/html/v5.19/admin-guide/pm/cpufreq.html?highlight=cpufreq
[5] https://www.kernel.org/doc/html/v5.19/admin-guide/pm/intel_pstate.html

Figure 2. Illustration of Possible Use Cases with and without DFS



The standard configuration serves as a baseline, as it was Intel's recommended approach for achieving the highest determinism before the 11th generation Intel® Core™ processors. In this scenario, all DFS features are disabled, and the core frequency is fixed at the base frequency.

Configurations 2, 3 and 4 leverage the improved DFS features. Configuration 2 is the default configuration with the Intel reference BIOS. With this configuration, Intel's DFS features like Intel SpeedStep®, Intel® Speed Shift/ HWP and Intel® Turbo Boost Technology are enabled, and all cores have the capability to dynamically adjust their frequency based on workload demand, scaling from the most power-efficient frequency (Pn) to the maximum turbo frequency (P0).

In Configuration 3, we adhere to the state-of-the-art recommendation for the RT-Core by consistently locking the core at its base frequency (P1). The BE-Cores now can dynamically adjust their frequency based on workload demand. This ensures that demanding tasks, such as Human-Machine Interface (HMI), AI inferencing, or Computer Vision (CV), among others, can attain peak performance.

In Configuration 4, the High-Performance Real-Time scenario, the RT-Core functions at a static turbo frequency[7], providing enhanced single-threaded performance and a high level of determinism for real-time applications. To safeguard the processor within its power and thermal thresholds, the maximum frequency of the BE-Cores is constrained. This measure is taken to maintain a balance between high performance and the processor's power and thermal constraints.

In the upcoming sections, we will take a closer look at the four configurations, emphasizing the specific factors that require consideration when working with DFS and quantifying their impact on real-time performance. To achieve this, we will simulate real-world scenarios that involve the consolidation of mixed-criticality real-time workloads.

[7] This usage condition has not yet been considered in Intel's "Use Reliability Scenario" reports. Please contact your Intel® Technical Support team for additional information.

Analysis

Setup and Configuration

The following hardware, firmware and software setup was used for the analysis:

- Intel® Core™ i7-1370RE
- Intel® Customer Reference Board (CRB)
- Intel® IoT UEFI Reference BIOS PV+ - version 4081_04[8]
- Ubuntu* 22.04.2 LTS with Intel® kernel overlay for RT kernel - PV+ Release[9]
- cyclictest - version 1.6
- stress-ng - version 0.13.12
- cwlg

To assess the impact of DFS on the real-time performance of the processor, we employed the four configurations discussed earlier, with **Configuration 1** serving as our baseline as it represents the state-of-the-art recommendation.
In general, we used the default Intel reference BIOS settings, making specific modifications detailed in . For all configurations, we enabled the Intel® TCC Mode, a BIOS switch designed to optimize firmware settings for low latency. Alongside the Intel® TCC Mode, adjustments were made to Intel DFVS features – deactivated for **Configuration 1** and activated for the other configurations. The variations between the configurations are highlighted in grey.

Table 1. BIOS Settings

Configuration 1 - Baseline	Configuration 2 - Default	Configuration 3 and 4 - Advanced
Intel® TCC Mode = Enabled	Intel® TCC Mode = Enabled	Intel® TCC Mode = Enabled
Intel SpeedStep® = Disabled	Intel SpeedStep® = **Enabled**	Intel SpeedStep® = Enabled
Intel® Speed Shift Technology = Disabled	Intel® Speed Shift Technology = **Enabled**	Intel® Speed Shift Technology = Enabled
Intel® Turbo Mode = Disabled	Intel® Turbo Mode = **Enabled**	Intel® Turbo Mode = Enabled
	Energy Efficient Turbo = **Enabled**	Energy Efficient Turbo = **Disabled**
	HWP Autonomous EPP Grouping = **Enabled**	HWP Autonomous EPP Grouping = **Disabled**

Apart from the BIOS adjustments, we also used the kernel command line settings detailed in and recommended in the Intel® TCC User Guide[10]. The **Configuration 2, 3 and 4** columns in the table illustrates the specific modifications necessary to activate the intel_pstate driver compared to **Configuration 1**. As mentioned earlier, when Intel® Speed Shift is enabled in BIOS, the intel_pstate driver, in its active mode, automatically activates HWP and delegates p-state control to the processor.

[8] Intel Resource and Design Center (RDC) Doc ID #779111
[9] Intel Resource and Design Center (RDC) Doc ID #782603
[10] Intel Resource and Design Center (RDC) Doc ID #786715

Table 2. Linux Kernel Command Line Parameters

kernel cmd-line parameters	Configuration 1	Configuration 2, 3 and 4
processor.max_cstate	0	
intel_idle.max_cstate	0	
clocksource	tsc	
tsc	reliable	
nowatchdog		
intel_pstate	disable	active
noht		
isolcpus	5,13	
rcu_nocbs		
rcupdate.rcu_cpu_stall_suppress	1	
rcu_nocb_poll		
irqaffinity	0	
i915.enable_rc6	0	
i915.enable_dc	0	
i915.disable_power_well	0	
I915.enable_guc	3	
mce	off	
hpet	disable	
numa_balancing	disable	
efi	runtime	
art	virtuallow	
iommu	pt	
nmi_watchdog	0	
nosoftlockup		
hugepages	1024	

CWLG

CWLG[11] is the compute workload portion of the Intel Real-Time Compute Performance (RTCP) benchmark.

CWLG is a pointer chasing workload that can take two parameters as inputs; a span size and a buffer size. The span size is the size of the span of continuous virtual memory that should be allocated for pointer chasing. The buffer size is the amount of the span that the workload will access. By configuring these two parameters, the workload can be tuned to mimic a workload that is memory bounded to the L1, L2, LLC, or memory. We use a 256KB span and buffer to stay within the private L2 throughout our testing.

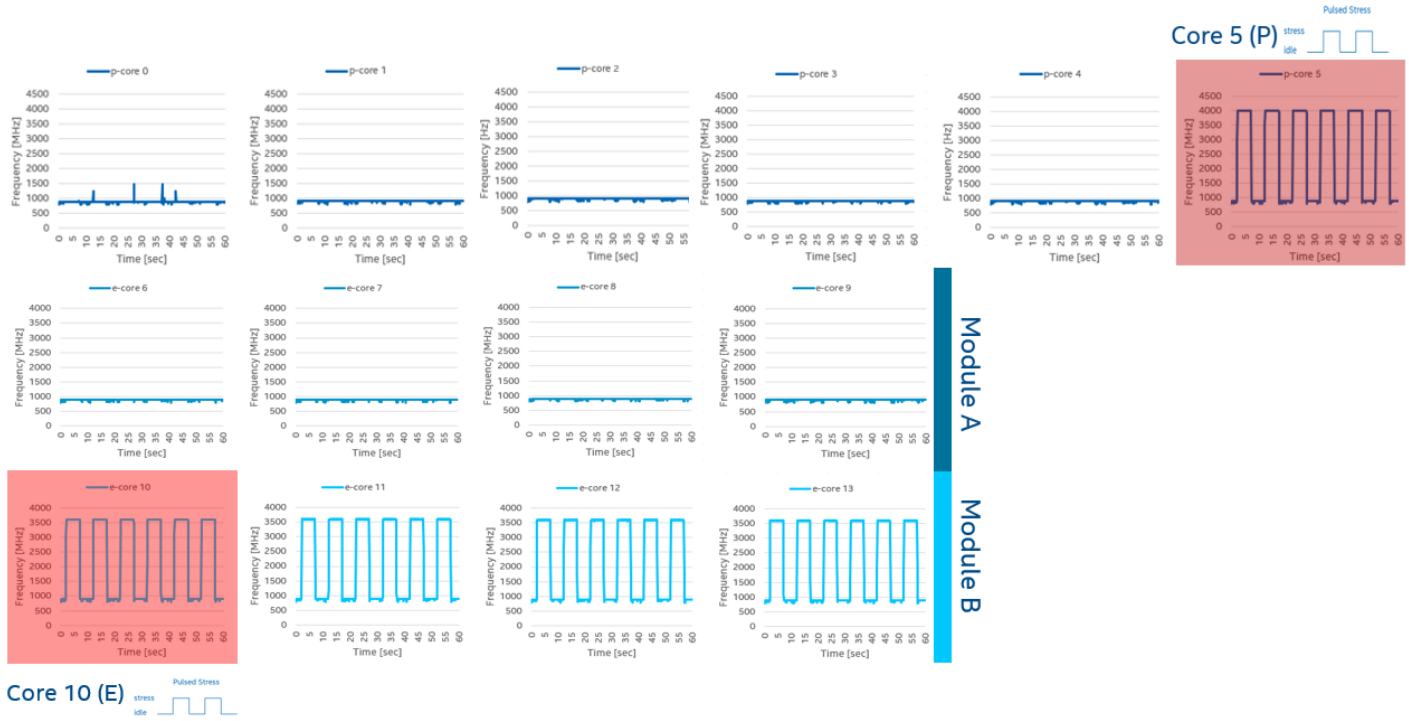
Per Core / Per Module p-state

Before we dive into analyzing the different configurations, let's confirm the basic theoretical considerations of per core and per module p-states, as discussed earlier.
In this experiment, we activated the DFS features of the CPU in the BIOS, disabled HWP Autonomous EPP Grouping, turned on HWP via the OS, and maintained the default HWP configuration for all cores. This default setup optimized the EPP for "power save," with the minimum performance set to the lowest

[11] Please contact your Intel® Technical Support team to get access to https://github.com/OTCShare2/rtcp-xdp.

level and the maximum performance set to the highest.
To induce p-state transitions on core 5 and core 10, we used a script to apply pulsed CPU stress to these cores. Core 5, being a P-Core with per-core p-state control, demonstrated frequency scaling. It adjusted its frequency dynamically, ranging from the most efficient frequency to the maximum turbo frequency in response to the stress workload, while all other P-Cores remained at their most efficient frequency, as depicted in .
On the other hand, core 10 is an E-Core. In this case, all E-Cores within the same module followed the scaling request generated by the pulsed stress workload running on core 10. This is because, in this generation, E-Cores have per-module p-state support. Consequently, all E-Cores within a module adjusted their frequencies together based on the demands of the workload.

Figure 3. Frequency Profile of CPU with Pulsed Stress on Intel® Core™ i7-1370PE Processor



Quantifying the Impact of DFS on Real-Time Performance: Analysis

Now that we've clarified the per-core and per-module p-states, let's continue with the analysis of the four configurations we previously outlined. **Configuration 1** serves as our baseline for all scenarios. We kick off our analysis with **Configuration 2**, reflecting the default or out-of-the-box configuration with Intel® TCC Mode activated in the Intel reference BIOS. Subsequently, our attention shifts to **Configurations 3** and **4** to showcase additional tuning possibilities.
For all scenarios, our objective is to assess the impact of DFS settings on RT performance. To accomplish this, we conducted various measurements. In all scenarios, we opted for a CPU-centric stressor, utilizing stress-ng. Additionally, we integrated stress-ng into a shell script to cyclically alternate between a 5-second "pulse" (i.e., stress activation) and a 5-second pause (i.e., no stress running). This approach was chosen intentionally to induce p-state transitions.
To quantify the impact on RT performance, we chose to use cyclictest to measure system latency and cwlg to quantify compute performance for the different scenarios.
illustrates the general hardware and software setup for our analysis. Core 5 (P-Core) and core 13 (E-Core) have been isolated using the Linux* command line primitives as RT-Cores, while the remaining cores serve as our BE-Cores. Furthermore, we employ Intel Cache Allocation Technology (CAT) to exclusively assign cache to the RT application.

Figure 4. General Hardware + Software Configuration on Intel® Core™ i7-1370PE Processor

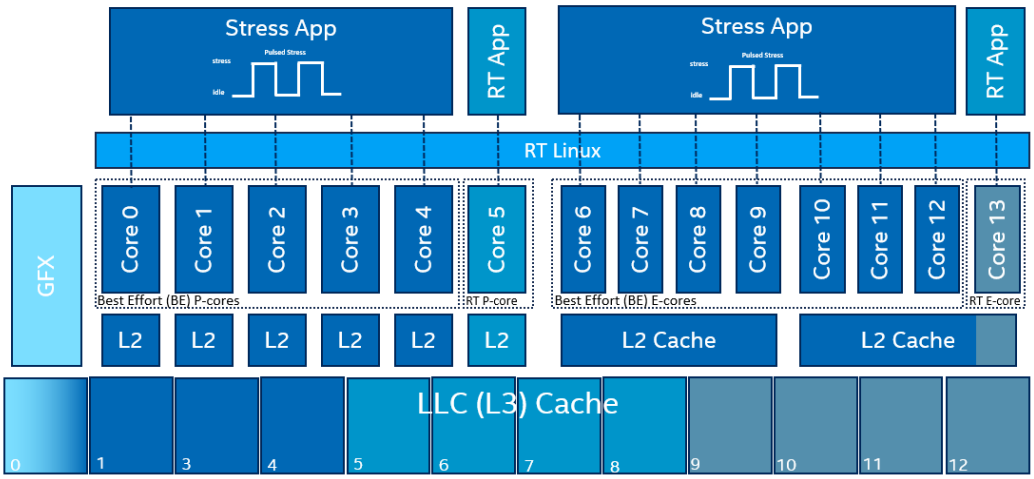


Table 3. General Test Setup

Item	Description
Cache Allocation Settings	L2 CAT: configured for E-Cores • RT App on Core 13: Run on CLOS 2 with Waymask `0xf` • Stress App Core 10-12: Run on CLOS 3 with Waymask `0x3f0` • Stress App Core 6-9: Run on CLOS 0 with Waymask `0x3ff` L3/LLC CAT: configured • RT App on Core 5: Run on CLOS 1 with Waymask `0xf` • RT App on Core13: Run on CLOS 2 with Waymask `0xf0` • Stress App: Run on CLOS 0 with Waymask `0xf00` GT Clos: Waymask `0x800`
Stress Application	`stress-ng -c12 --cpu-load 100 --cpu-method fft --taskset 0-4,6-12`
RT Application	`cyclictest -t2 -a5,13 -D 12h -p99 -m -i250 -h100 -q --latency=1` `taskset -c 5 cwlg -b 256KB -s 256KB -l 10000000`

Configuration 2: DFS BIOS and OS Default Configuration

Now, turning our attention to **Configuration 2**, this setup represents our default or out-of-the-box configuration. In the BIOS, we have TCC Mode, Speed Step, Speed Shift, Turbo Boost Technology, Energy Efficient Turbo, and HWP Autonomous EPP Grouping all enabled. By default, the EPP value for all cores is set to "power save (0x80)," and the HWP autonomous algorithm is capable of scaling the frequency of all cores between Pn and P0 – details can be found in and Table 4.

Figure 5. Core Frequency Configuration for Configuration 2 on Intel® Core™ i7-1370PE Processor

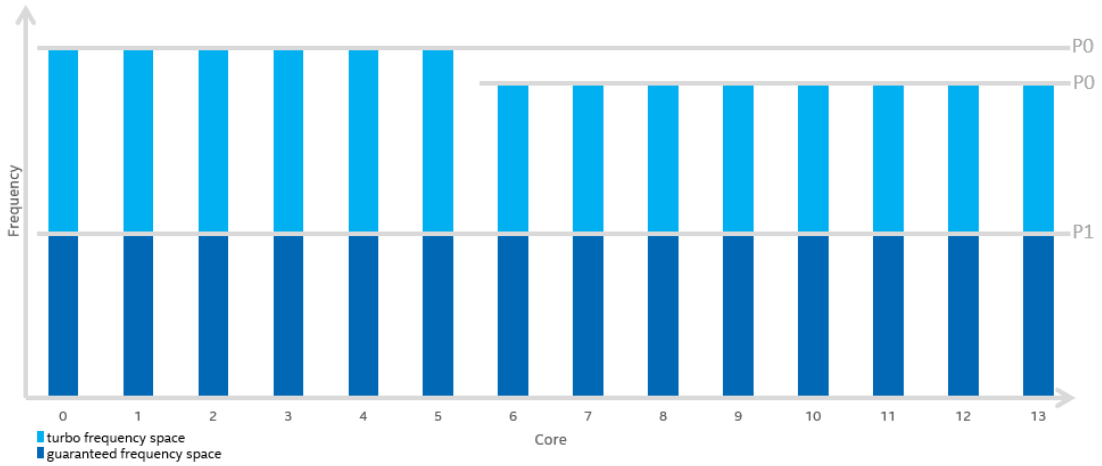


Table 4. Test Setup for Configuration 2

Item	Description
HWP Request MSR (IA32_HWP_REQUEST)	#P-Cores core 0: 0x80003d04 core 1: 0x80003d04 core 2: 0x80003d04 core 3: 0x80003d04 core 4: 0x80003d04 core 5: 0x80003d04 #E-Cores core 6: 0x80002504 core 7: 0x80002504 core 8: 0x80002504 core 9: 0x80002504 core 10: 0x80002504 core 11: 0x80002504 core 12: 0x80002504 core 13: 0x80002504

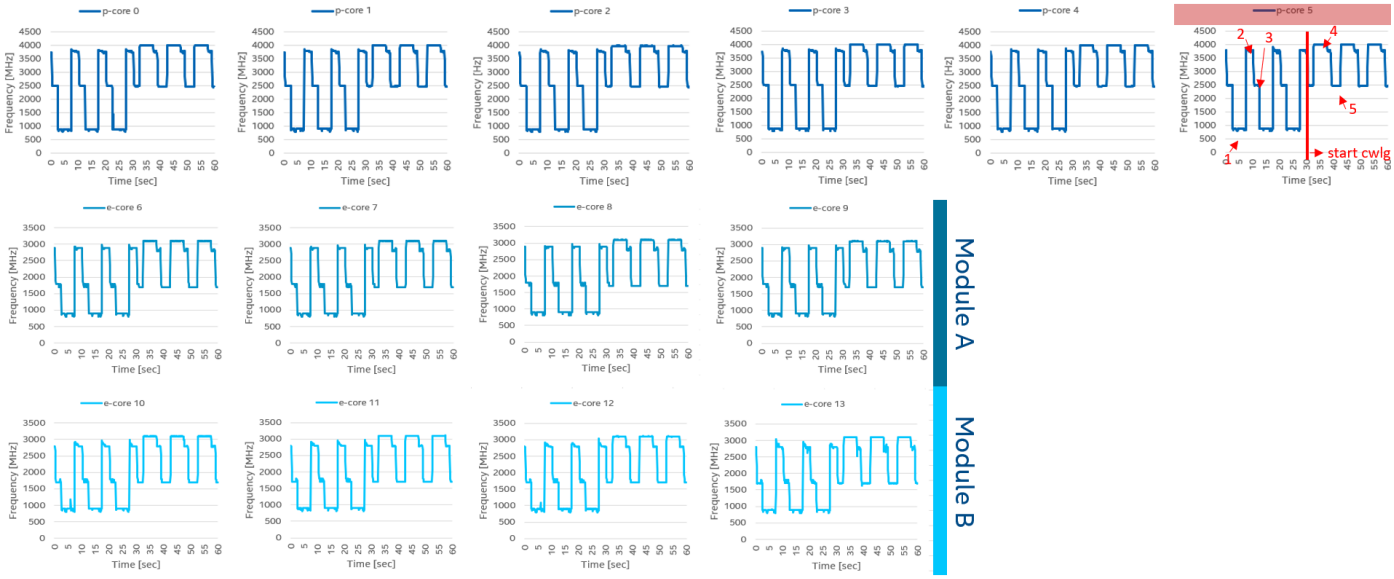
Figure 6 depicts the core frequency profile under a pulsed stress workload on BE-Cores and the execution of `cyclictest` or `cwlg` application on RT-Cores. In the initial 30 seconds, the RT-Cores remained idle, however the core frequency follows the frequency of the BE-Cores, despite having per-core p-states for the p-cores. This behavior is attributed to the enabled EPP grouping, where the HWP autonomous algorithm requests the same core frequency for all cores sharing the same EPP value. Additionally, you can observe that EPP is optimized for "`power save`," as the frequency scales down to the most efficient frequency when core utilization is low.

We have placed some markers exemplarily in to explain the behavior more precisely. At marker 1, the core frequency stabilizes at most efficient frequency when the stress application is inactive and core utilization is low. When the stress application is activated at marker 2, the core frequency scales to the maximum supported turbo frequency. At marker 3, HWP initiates throttling on all cores as the turbo budget is fully consumed and power limit throttling is triggered.

After 30 seconds, the `cwlg` workload is launched on p-core 5, one of the isolated RT-Cores. Notably, the cores persistently operate at high frequencies, even during periods when stress application is inactive, due to the constant load caused by `cwlg`. With all cores part of the same EPP group, the frequency of all

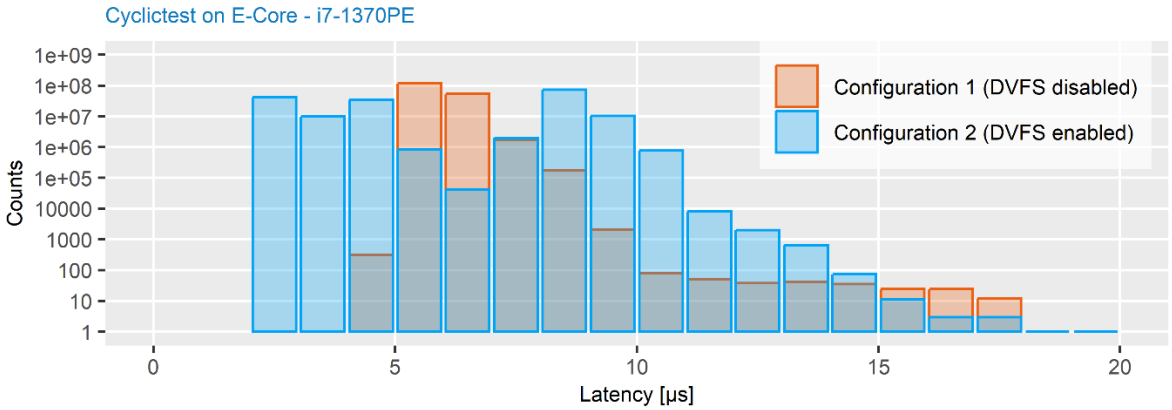
cores follows to the load on p-core 5. At marker 4, when the stress application is inactive, and the turbo budget is available for `cwlg`, the core frequency increases to the maximum supported turbo frequency. Conversely, when the stress application starts, and the turbo budget is consumed, all cores are throttled to stay within the configured power limit of the platform, highlighted at marker 5.

Figure 6. Frequency Profile of CPU with Configuration 2 and Pulsed Stress

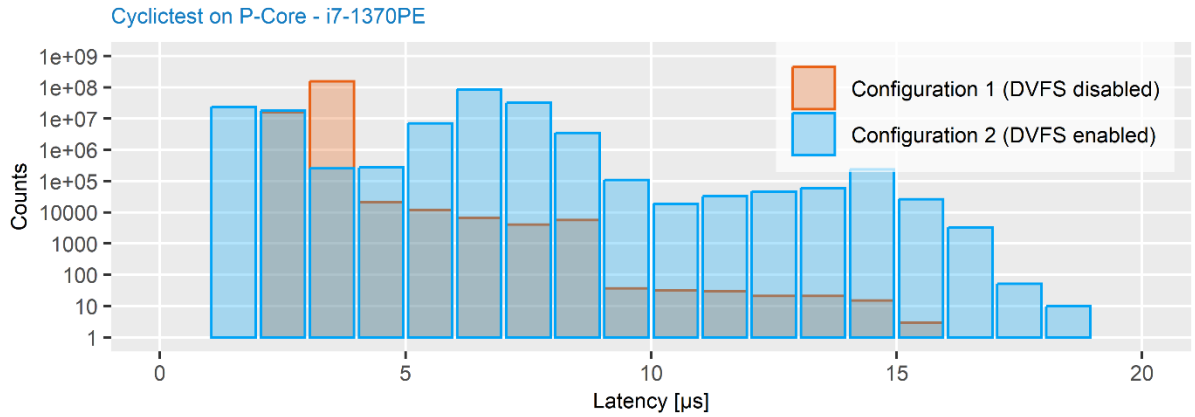


Now that we have a better understanding of the frequency scaling behavior, let's look at the system latency. As depicted in , although the core frequency of our RT-Cores is not locked to a fix frequency, the maximum latency measured for e-core and p-core is only slightly worse than the results with **Configuration 1** which is our baseline configuration with DFS features disabled. The better results for minimum latency with **Configuration 2** are because of the capability to leverage turbo frequency.

Figure 7. Latency Histograms of Default Configuration with Pulsed CPU Stress on BE-Cores[12]

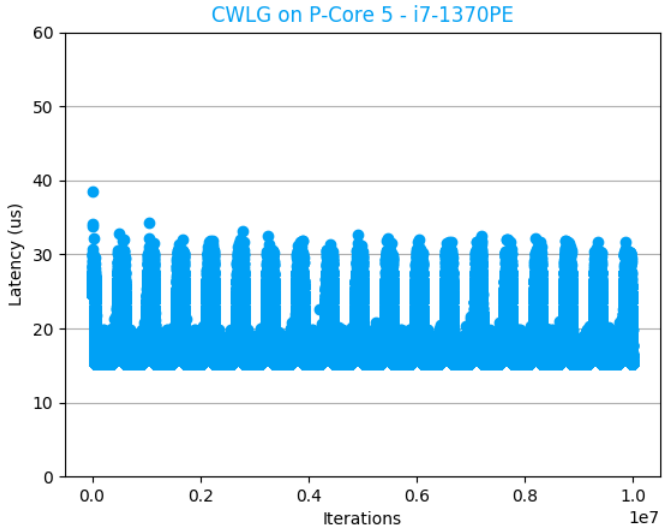


[12] Performance varies by use, configuration and other factors. Learn more at www.Intel.com/PerformanceIndexPerformance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. No product or component can be absolutely secure.



Finally, let's examine the execution latency on the RT-Core for the default DFS settings. In this analysis, we measured the execution time of the cwlg application on the isolated p-core concurrently with the pulsed stress on the BE-Cores. The core frequency profile aligns with the last 30 seconds shown in . As depicted in , there is a noticeable jitter in the execution time. This variability is attributed to the fact that the core frequency is not fixed and can scale to the maximum supported turbo frequency or undergoes throttling when the power limit is exceeded.

Figure 8. Execution Time of cwlg with Configuration 2[13]



To conclude this section, the analysis of **Configuration 2** with the default DFS settings reveals that, due to EPP grouping, all cores maintain the same core frequency. The observed jitter in execution time, attributed to continuous core frequency scaling of the RT-Core, provides further valuable insights into the challenges associated with DFS in the context of real-time workloads. The maximum system latency is only slightly higher compared to our **Configuration 1** with DFS features disabled and core frequency locked to base frequency.
However, the results with DFS feature enabled show a significant improvement compared to platforms before 11th Intel® Core™ processors[14] and already meet a performance level for a broad spectrum of real-

[13] Performance varies by use, configuration and other factors. Learn more at www.Intel.com/PerformanceIndexPerformance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. No product or component can be absolutely secure.

[14] "Dynamic Frequency Scaling for Mixed Criticality Real-Time Scenarios on 11th Generation Intel® Core™ Processors" - RDC Id #723446

world applications. The next sections will delve into additional tuning options, aiming to improve the system's compute performance and determinism even further.

Configuration 3 – High Peak Performance for BE-Cores

As introduced earlier, **Configuration 3** simulates a standard scenario with peak performance for the BE-Cores and high determinism for the RT-Cores. Table 5. **Test Setup Use Case 1** illustrates the configuration of the core frequency for the analysis. Core 5 (P-Core) and core 13 (E-Core) have been isolated as RT-Cores, with the core frequency fixed to the base frequency, while BE-Cores can scale the frequency between Pn and P0, except for the E-Cores within the same module as our RT-Core 13. Here, all cores within the module are locked to the base frequency. One reason is that we have only one PLL per module as stated earlier, and another reason is that all cores within a module always follow the request with the highest requested and supported frequency.

As shown in the `IA32_HWP_REQUEST` MSR values in , the RT-Cores are optimized for performance with an EPP value of 0, while the BE-Cores are still optimized for power save, but each core has a different EPP value to avoid EPP grouping. Furthermore, we set the valid bits to prevent the register from being overruled by `IA32_HWP_REQUEST_PKG` MSR, and the `Desired Performance` bitfield is configured for the RT-Cores to make an explicit performance request.

Figure 9. Use-Case 1 Core Frequency Configuration on Intel® Core™ i7-1370PE Processor

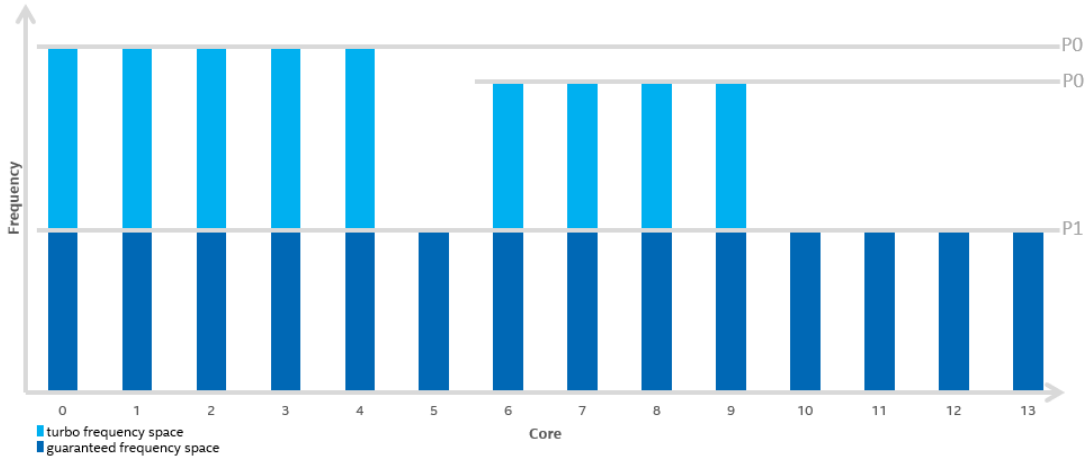


Table 5. Test Setup Use Case 1

Item	Description
HWP Request MSR (`IA32_HWP_REQUEST`)	#P-Cores core 0: `0xe000000080003d01` core 1: `0xe000000081003d01` core 2: `0xe000000082003d01` core 3: `0xe000000083003d01` core 4: `0xe000000084003d01` core 5: `0xe000000000191919` #E-Cores core 6: `0xe000000085002501` core 7: `0xe000000086002501` core 8: `0xe000000087002501` core 9: `0xe000000088002501` core 10: `0xe000000000000c0c` core 11: `0xe000000000000c0c` core 12: `0xe000000000000c0c` core 13: `0xe0000000000c0c0c`

As depicted in , the cores which can scale show the same behavior as discussed in the section before with our default settings. Notably, the core frequency of the RT-Cores, highlighted in red, remains consistently locked to the base frequency even when BE-Cores are throttled.

Upon comparing the `cyclictest` results of **Configuration 1** and **Configuration 3**, showcased in , it becomes evident that there is no significant impact of DFS on real-time performance for the test setup. In fact, there is even an improvement observed for **Configuration 3** compared to the results with the default settings in the previous section.

Figure 10. Frequency Profile of CPU with Configuration 3 and Pulsed Stress

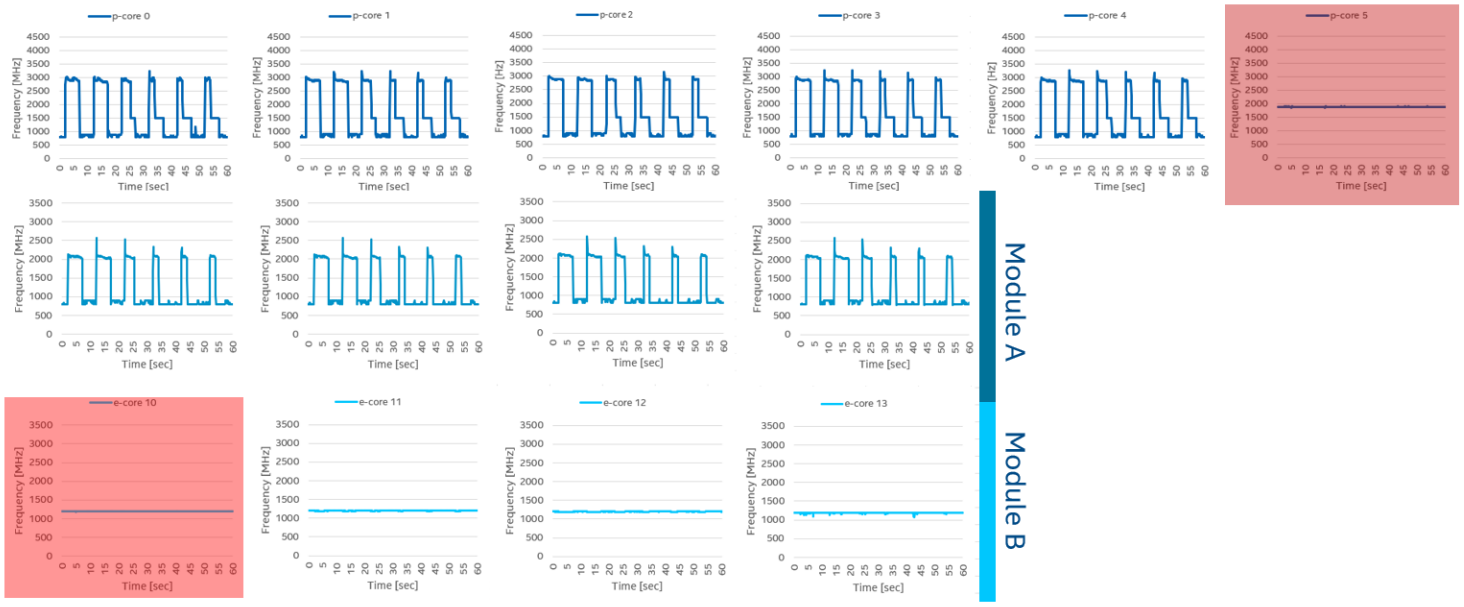
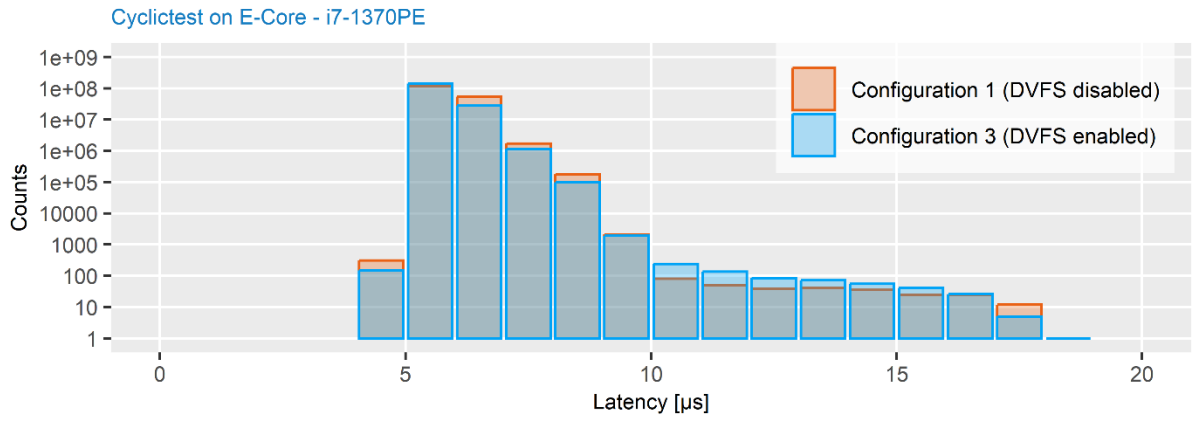
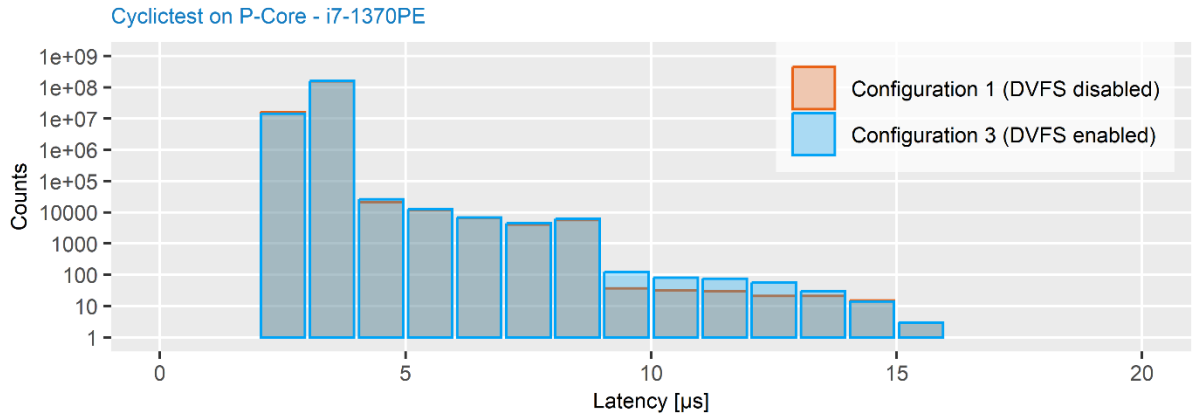


Figure 11. Latency Histograms of Use Case 1 with Pulsed CPU Stress on BE-Cores and Cyclictest on RT-Cores[15]



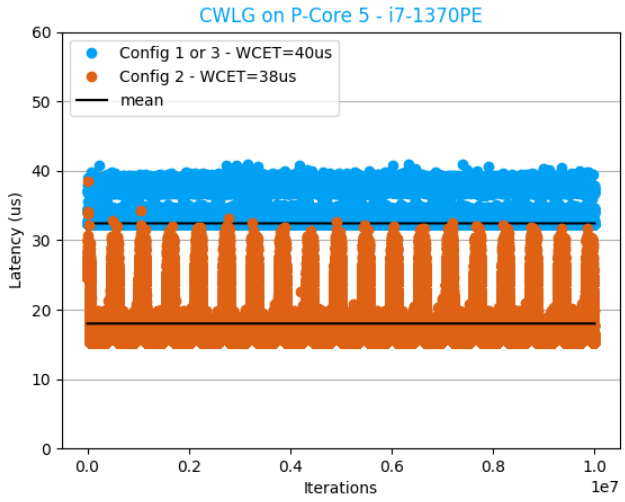
[15] Performance varies by use, configuration and other factors. Learn more at www.Intel.com/PerformanceIndexPerformance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. No product or component can be absolutely secure.



Analyzing the execution latency of cwlg on our RT P-Core 5, as illustrated in , reveals that the WECT is higher for **Configuration 1** or **3** compared to **Configuration 2**. This discrepancy arises because, in this specific scenario, the core frequency of **Configuration 2** remains higher, even as the cores are throttled - see marker 3 in , in contrast to the core frequency of Core 5 in Table 5. **Test Setup Use Case 1**.

With **Configuration 3**, the jitter band is narrower. In conclusion, if the WCET of **Configuration 1** or **3** meets the requirements for your real-time workload, opting for these configurations is recommended. This choice ensures a narrower jitter band and aligns with the state-of-the-art standard.

Figure 12. Execution Latency of cwlg with Config 1 or 3 vs Config 2[16]



As demonstrated in this section, there is no observable difference in latency between the baseline **Configuration 1** and **Configuration 3**. However, **Configuration 3** allows BE-Cores to leverage peak performance for tasks such as Human-Machine Interface (HMI), AI inferencing, or Computer Vision (CV), for example.

[16] Performance varies by use, configuration and other factors. Learn more at www.Intel.com/PerformanceIndexPerformance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. No product or component can be absolutely secure.

Configuration 4 – High-Performance RT-Core

In this section we will go into details of **Configuration 4**, High-Performance RT.

As introduced in the earlier section, **Configuration 4** simulates a scenario in which the RT-Core(s) operate at a stable turbo frequency and still maintain a high-level of determinism at the expense of the BE-Cores. Figure 13 illustrates the core frequencies of **Configuration 4**. Core 5 (P-Core) has been isolated as the RT-core and fixed to the multi-core max turbo frequency. While the BE-Cores will scale between Pn and P1 depending on available power budget as demonstrated in Figure 14.

Figure 13. Config 4 Core Frequency Configuration on Intel® Core™ i7-1370PE Processor

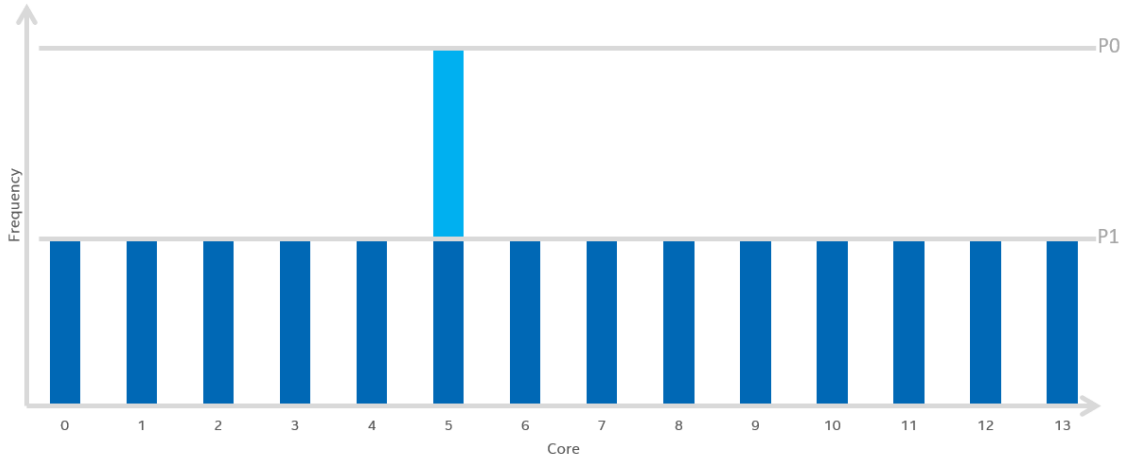
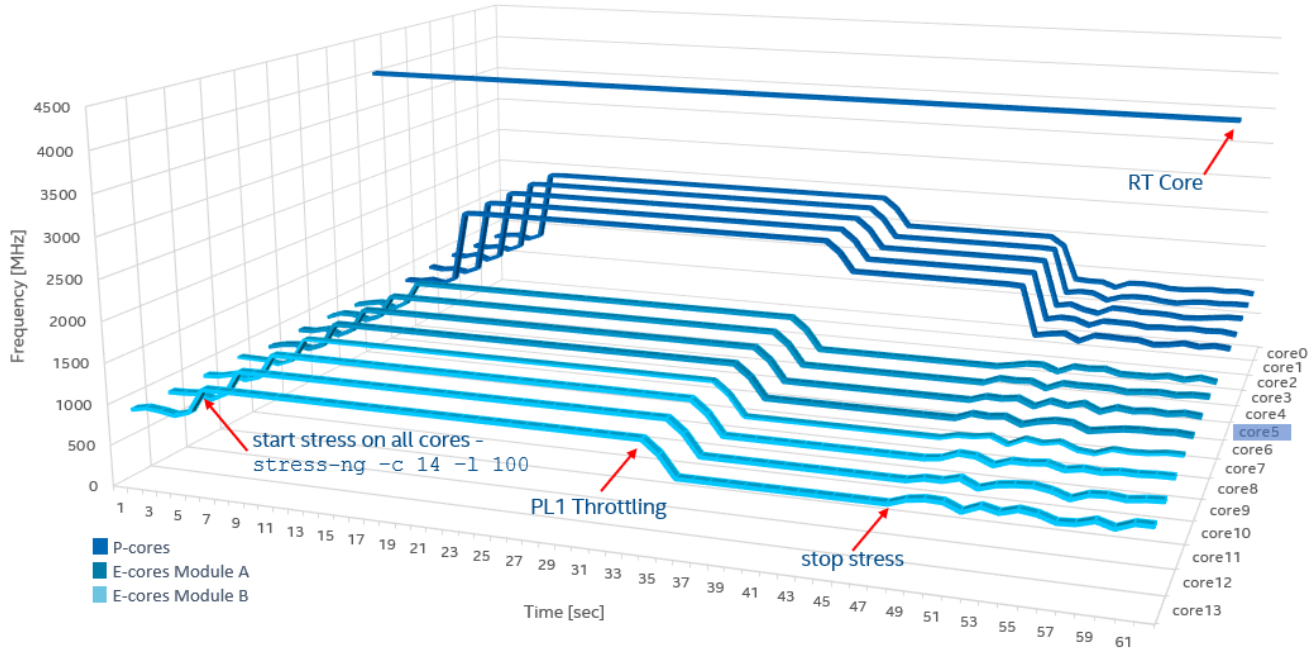


Figure 14. Core Frequency over Time on Intel® Core™ i7-1370PE Processor



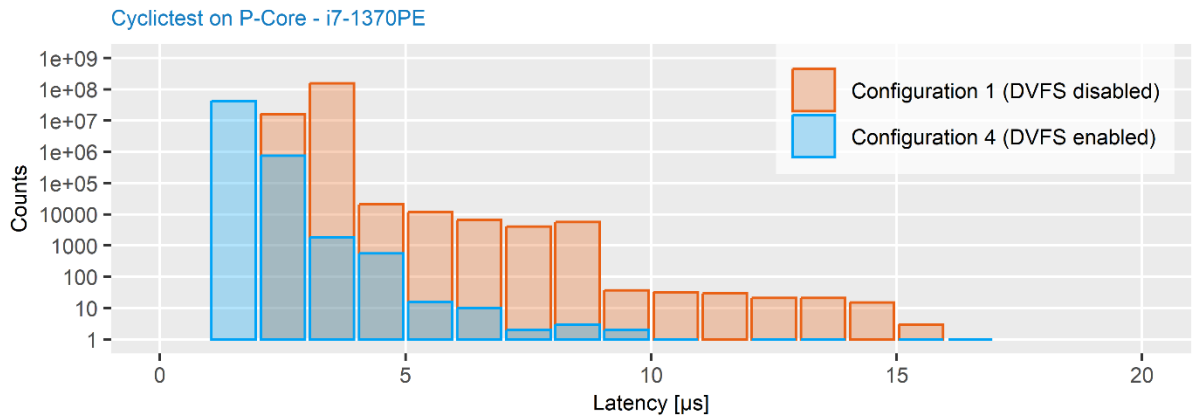
With the EPP considerations discussed previously, in the Energy Performance Preferences section, Table 7 shows the `IA32_HWP_REQUEST` MSR configuration along with any test parameters for testing during **Configuration 4**. Note, like **Configuration 3**, the RT-Core is optimized for performance, BE-Cores for power save, all cores have different EPP grouping values, and valid bits are set.

Table 6. Test setup for Use Case 2

Item	Description
HWP Request MSR (`IA32_HWP_REQUEST`)	#P-Cores core 0: `0xe000000080001901` core 1: `0xe000000081001901` core 2: `0xe000000082001901` core 3: `0xe000000083001901` core 4: `0xe000000084001901` core 5: `0xe0000000003d3d3d` #E-Cores core 6: `0xe000000085000c01` core 7: `0xe000000086000c01` core 8: `0xe000000087000c01` core 9: `0xe000000088000c01` core 10: `0xe000000089002525` core 11: `0xe00000008a002525` core 12: `0xe00000008b002525` core 13: `0xe00000008c002525`
Stress Application	`stress-ng -c13 --cpu-load 100 --cpu-method fft --taskset 0-4,6-13`
RT Application	`cyclictest -t2 -a5 -D 6H -p99 -m -i250 -h100 -q --latency=1`
Monitoring Application	Core frequency and package power consumption was monitored with … `turbostat -i 1 -n 60 -s Bzy_MHz,PkgWatt`

Now, let's examine the `cyclictest` results from **Configuration 4** against the results using **Configuration 1** with pulsed CPU stress on BE-Cores and `cyclictest` on the RT-Core (Figure 15).

Figure 15. Use Case 2 RT P-Core Cyclictest Latency Histogram[17]

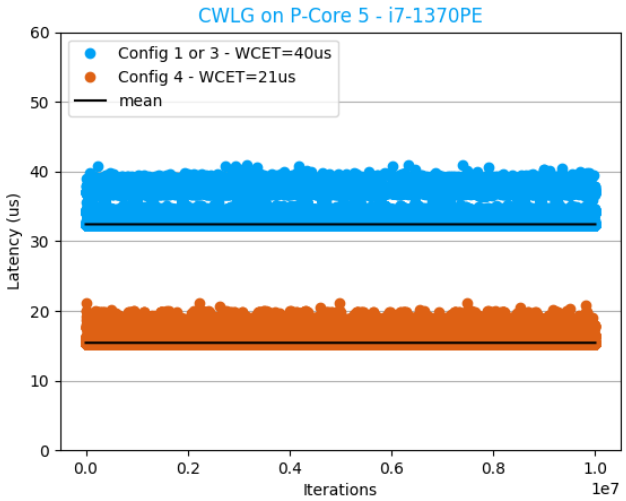


[17] Performance varies by use, configuration and other factors. Learn more at www.Intel.com/PerformanceIndexPerformance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. No product or component can be absolutely secure.

Configuration 4, as shown in , reduces average `cyclictest` latency with little to no impact on max cyclictest latency. This reduction in average latency is due to the increase in the RT-Core frequency.

Next, let's examine the `cwlg` results against the results of **Configuration 1** or **3** (Figure 16).

Figure 16. Execution Latency of cwlg with Config 1 or 3 vs Config 4[18]



Configuration 4, as shown in Figure 16, significantly reduces both the WECT and average execution time of `cwlg` when compared to **Configuration 1** or **3** where the RT-Core is locked to base frequency.

Configuration 4 offers both the fastest execution time and highest determinism for the RT-Core, but these performance improvements come at the cost of BE-Core performance.

Conclusion

The goal of this whitepaper was to explore new configurations for the 12th and 13th generation of Intel performance hybrid architecture processors, arising from enhancements in DFS features, with a specific focus on mixed-criticality real-time workloads demanding bounded WCET.

We initially delved into the distinctions between per core and per module p-states and their implementation in the 12th and 13th generation Intel performance hybrid architecture processors. Subsequently, we introduced four configurations:

- Configuration 1: Standard or state-of-the-art
- Configuration 2: Default or out-of-the-box
- Configuration 3: Peak performance for BE-Cores
- Configuration 4: High RT performance

[18] Performance varies by use, configuration and other factors. Learn more at www.Intel.com/PerformanceIndexPerformance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. No product or component can be absolutely secure.

Configuration 1 served as our baseline, aligning with Intel's pre-11th generation Intel® Core™ processor recommendation to disable all DFS features, essentially locking all cores to the base frequency.

The analysis of our default configuration (**Configuration 2**) revealed that DFS features no longer significantly impact RT performance in terms of system latency and WCET. We also discovered that the execution time has a wider range of jitter because the core frequency is not fixed. Nonetheless, the overall performance has significantly improved, allowing us to serve a wide range of use-cases with our out-of-box configuration.

Configurations 3 and **4** took a step further, showcasing options to fine-tune the system for specific use cases requiring either peak performance for demanding BE workloads or high single-threaded performance for RT workloads.

Configuration 3 follows the state-of-the-art recommendation, avoiding frequency scaling for RT-Core(s) while enabling BE-Cores to leverage peak performance. `Cyclictest` results demonstrated improved temporal isolation in terms of DFS between the cores, with no difference compared to DFS features being disabled. While the WCET of `cwlg` is slightly higher compared to the default configuration, the narrower jitter band makes this configuration a suitable choice for those needing high peak performance for demanding BE workloads and high determinism for RT workloads.

The analysis of **Configuration 4** showcased the potential to reduce average system latency and significantly decrease the WCET through fine-tuning the HWP settings. Overall, the results suggested that this configuration could be an ideal fit for use-cases requiring high single-threaded performance, albeit with a reduced BE performance.

In summary, DFS enhancements in the 12th and 13th generations of performance hybrid architecture offer increased flexibility for future system architectures with mixed-criticality RT workloads. All analyzed configurations showed that the impact of DFS on RT performance has greatly improved and can meet various workload requirements and maintain a high level of determinism for RT applications. With **Configuration 3**, we achieve improved performance for BE applications, albeit with a slightly higher WCET for RT applications compared to the default setup, although it remains comparable to the standard configuration. On the other hand, with **Configuration 4**, we significantly reduce the WCET for RT applications, but this comes at the cost of peak performance for BE applications.

Future Work

Exploring further granularity in DFS tuning for BE versus RT workloads and presenting detailed results for BE metrics like SPEC benchmarks and RT workloads like Intel RTCP would be a promising avenue for future research. This could provide a more comprehensive understanding of the performance trade-offs and optimizations in diverse scenarios.

Investigating the practical implementation of the findings within the industry could also be a valuable direction for future work. Collaborating with industry partners to deploy and validate these configurations in real-world applications could offer insights into the practicality, challenges, and benefits of adopting these optimizations.

Additionally, addressing power balancing and frequency throttling between the cores and integrated graphics unit for instance could be a relevant focus for future research. Explore how the existing tuning knobs can be used to optimal balance to power consumption and performance across these different

components, especially in hybrid architectures, could lead to more efficient and reliable computing solutions for RT applications in a mixed-criticality RT system architecture.

Overall, future work could aim to deepen our understanding of fine-tuning DFS configurations, explore industry implementations, and in general optimize power management strategies in hybrid architectures for mixed-criticality RT system architectures especially in the industrial segment.

Document number: 814793, Revision: 1.0 - February 2024