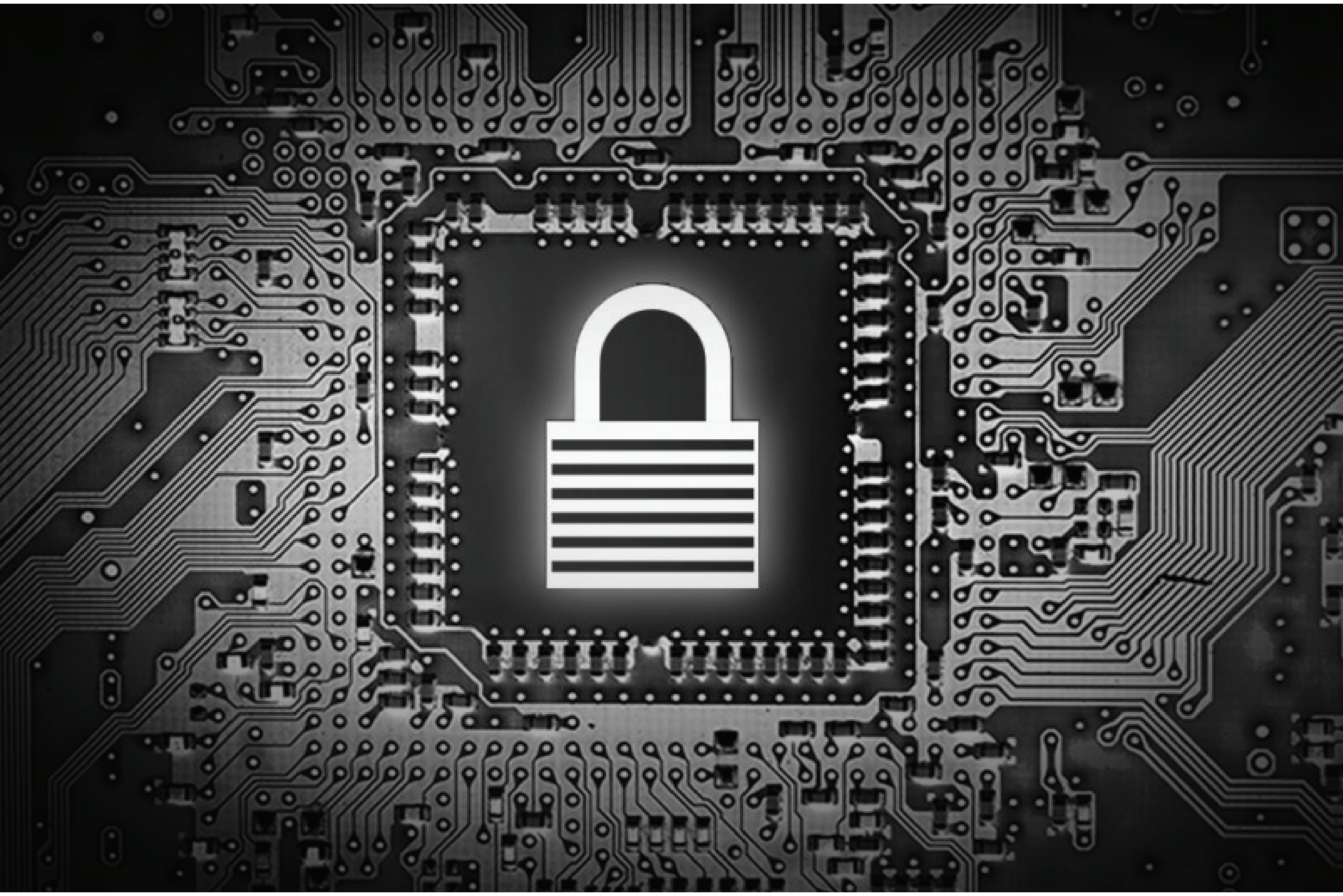
AMD EMBEDDED SOLUTIONS



White Paper |

UNDERSTANDING POWER MANAGEMENT AND PROCESSOR PERFORMANCE DETERMINISM

7.12.18

UNDERSTANDING POWER MANAGEMENT AND PROCESSOR PERFORMANCE DETERMINISM

Abstract–High-performance embedded systems crave the processing power of modern x86 processors, but current hardware architectures consistently prioritize peak performance over deterministic behavior. Advanced power management methods exploit inherent part-to-part variations, boosting core frequencies in unpredictable ways. Adding to this, PC architectures tend to target specific processor power constraints that can artificially clamp operating frequencies to maintain thermal and electrical specs. This creates scenarios where the power-density of the workload defines the effective operating frequency of the CPU, further reducing predictability. Real-time operating systems are there to help address determinism in the software domain but they cannot address it at the hardware level. Once these hardware implications are understood, designers will know what to look for when choosing processors for embedded systems where performance determinism is an important factor. Discover methods to disable features of modern processors that reduce hardware determinism.

Keywords – determinism; deterministic; x86; performance; power; management; real-time; AMD;

INTRODUCTION

Embedded system applications span a tremendous range of uses and some of these devices become mission critical equipment where performance behavior must be highly predictable. Embedded system designers in these markets are familiar with the use of real-time operating systems to improve determinism at the software level, but variations introduced by hardware are often overlooked. For this work, hardware determinism is defined as a guaranteed, predictable response time to an event, assuming a fixed sequence of code and input stimuli. Deterministic systems can replicate that predictability across all units. The increased demand for high-performance embedded systems has also driven a trend toward usage of PC-compatible x86 processors from desktop and notebook product lines, though their power and performance architecture are not designed with determinism in mind. Even product variants targeted at embedded markets tend to retain the favoritism toward performance prevalent in the PC models. Power management behavior in leading x86 processors has consistently striven to squeeze the last drop of performance out of every device, including exploitation of inherent part-to-part variations. This paper will review the source of these variations, discuss common power management behaviors that exploit them, and review methods of mitigation. Focus will be on common desktop, notebook, and embedded processors in the 6-65W power range and may not be reflective of x86 server processors.

SILICON BASICS

DEFINING OPERATIONAL LIMITS

Before power management behaviors can be discussed, it is important to understand the fundamental limitations of silicon integrated circuits. In fact, the primary purpose for power management in such devices is to ensure these limitations are not exceeded so that device reliability and functionality are maintained. There are many factors that affect silicon-based transistor performance, but the focus here is to briefly familiarize readers with the most significant factors affecting x86 processors in their typical operating ranges.

Processor frequency is possibly the most obvious of performance limiting factors. Even consumers have become quite familiar with equating frequency to performance. Frequency defines how fast the logic of the device is clocked, and thus how fast instructions are executed. Performance will not be equivalent when comparing two processors of equivalent frequency and different architecture, but it is generally true that increasing frequency will increase execution performance. Frequency in a processor can be limited by several underlying factors, but the most basic are voltage and current. Those familiar with transistor mechanics know that voltage has a key relationship to frequency. Faster switching of the transistors requires increasing voltage to overcome the resistive and capacitive elements of the transistor. However, higher voltage increases ageing effects (Gielen, 2013), putting practical limits on voltage application to ensure product longevity. Faster switching of transistors also generates higher currents as those capacitive elements are charged and discharged. While individual transistor currents may be very small, modern processors can have several billion transistors (Cutress, 2017), so this current adds up quickly. The processor die is typically mounted on a package of some kind and there are also real, practical limitations to how much current can be delivered to the die effectively. Every digital IC must deal with the balancing act of transistor voltage and current to yield a useful frequency.

The combination of Ohm's and Joule's laws teach us that all this voltage and current generates power, and that both parameters have a direct relationship with power. In fact, the reality is that most processor frequency limitations also boil down to power or current limits. Faster switching of transistors increases current and may also require increasing voltage, and doing either will increase power. Integrated circuits of every kind must provide designers with a maximum power consumption limit so that systems can be adequately designed to handle the current and cooling requirements. Power limits are often the most significant performance limiting factor, especially at the lower end of a device family's power range. Modern processors based on the x86 architecture tend to be power limited rather than frequency limited with heavy workloads. The reasons will be discussed in later sections.

Die temperature is a simple factor to consider, though not the most obvious. As the processor operates, consumed power is converted to heat. Heat affects transistor operating characteristics, as well as the rate of diffusion of the doping elements in the silicon that form the transistor junctions. Eventually, diffusion will change the electrical properties of the transistors until they fail to operate correctly and the processor will reach the end of its life. Limiting junction temperature in the device is critical for maintaining its expected longevity. Manufacturers will set maximum die temperatures for their products that must be followed. Maintaining this temperature limit is an important task for the power management entity in the processor.

LEAKAGE POWER

Another basic principle of silicon transistors is that they leak current across junctions and to the substrate (Kaushik, 2003). The amount of leakage current in a processor of a particular process type will vary largely by applied voltage and temperature and it can become quite significant in today's high-performance processors. This is because the same factors that are required to make transistors switch faster (i.e., achieve higher frequency) also increase leakage. All this leakage current creates additional power that must be counted as part of the device's total power consumption. Naturally, leakage power effectively reduces the amount of the device's total power envelope that can be consumed as active power (i.e., power used in transistor switching that does work). Figure 1 below shows the leakage power distribution for a current, undisclosed AMD processor based on a 14nm FinFET process as a percentage of total processor power.

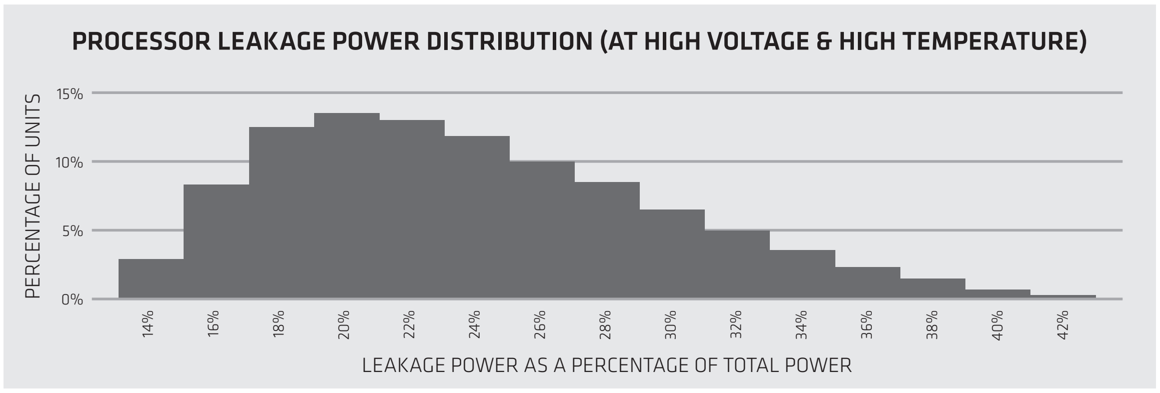


Figure 1- Leakage power distribution for an undisclosed AMD product based on a 14nm FinFET process.

Leakage power is exponentially related to die temperature, often doubling several times over the operating temperature of an integrated circuit (Wolpert & Ampadu, 2012). This means that device power will increase as the device temperature rises, even if the rest of the operating scenario is unchanged (i.e., fixed clock frequency, voltage, and workload). CPU manufactures must either leave enough headroom to accommodate this potential increase in power over the temperature range, or have a power management scheme that is dynamic with device temperature. Figure 2 below shows how leakage power is affected by temperature in that same AMD processor family.

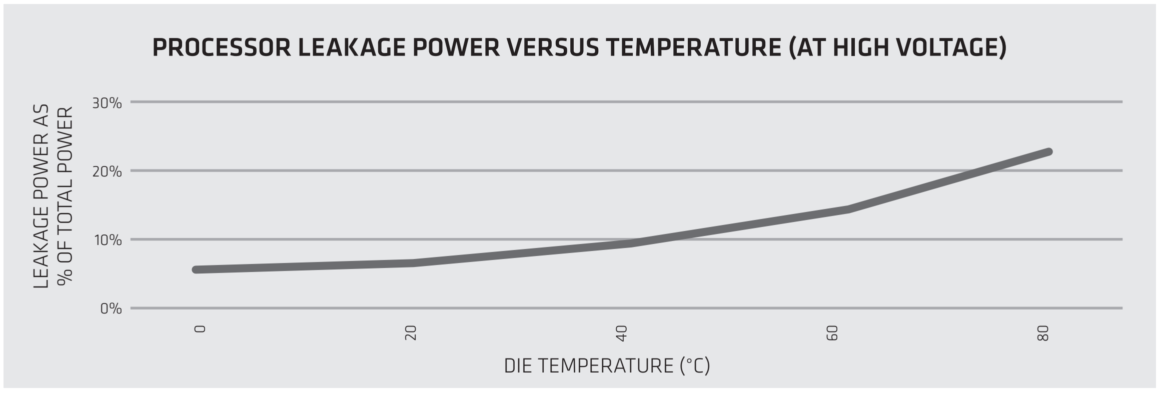


Figure 2 - Leakage power over temperature for a typical sample of an undisclosed AMD product based on a 14nm FinFET process.

PART-TO-PART VARIATIONS

The silicon photolithography process used to create semiconductors has inherent imperfections that manifest as variations in transistor construction and thus affect their operational characteristics. These variations not only exist between batches of silicon wafers, but even across a single wafer. Such variations may require die in one area of the wafer to have a higher voltage to achieve the same frequency than its neighbors, or cause its leakage power to be greater. Figure 1 illustrates leakage power variations quite well. Since power is a key factor in determining achievable performance for a given device, performance variations follow suite.

Processor manufacturers sort these die into groups targeting various product models with different specifications (e.g., 25W vs. 35W) to maximize yield. The amount of variation possible across units is defined by the specific model, and lower cost models will tend to allow wider variance. It is important to understand why these variations exist before discussing how power management exploits them.

WORKLOAD POWER DENSITY

Understanding power management behavior in complex microprocessors also requires understanding the concept of workload power density. This concept essentially means that different workloads (i.e., executed instruction sequences) will generate different amounts of power consumption in the processor, even at the same utilization level. This is to say that the central processing unit (CPU) core power incurred by two workloads can be significantly different even if the core is 100% utilized (i.e., consistently busy executing instructions) in both cases. This situation can occur because different instructions stimulate different amounts of transistor logic inside the core. As an example, one can image that a complex floating-point calculation will trigger more transistor activity in the CPU than a simple data movement operation. Data movement from one CPU general purpose register to another involves a minor number of gates while a complex AVX or SSE instruction to perform a multiply accumulate operation at 256 bits wide may activate many thousands of gates. Workloads may repeat such operations as part of an algorithm, compounding the power consumption increase. The potential difference in power between workloads becomes even larger when considering that nearly all x86 microprocessors sold today are multi-core, and most have integrated many other functions that were previously external. Integration of the graphics processing unit (GPU) is the most significant, as it is a very large processing core on its own. As consumer use-cases have become increasingly graphical, the GPU in some x86 processors can be even larger (i.e., more transistors) than the CPU cores. This is especially true for companies like AMD, who specifically target high performance integrated graphics in their microprocessors. Mixed workloads that execute a combination of CPU and GPU instructions simultaneously can experience the effects of workload power density differences on both core types. Allocation of the power budget to these various cores is one challenge of processor power management that will be explored further in later sections.

To illustrate the difference in workload power density, power consumption was measured with two different CPU-only workloads on a random sample of an AMD embedded RX-421BD SoC based on the "Excavator" CPU core. Both workloads can saturate a single CPU core while sustaining max frequency, so utilization will stay at 100% for the core under test. The Prime95 workload represents an extreme case (often referred to as a "thermal virus"), and power values have been normalized to that level.

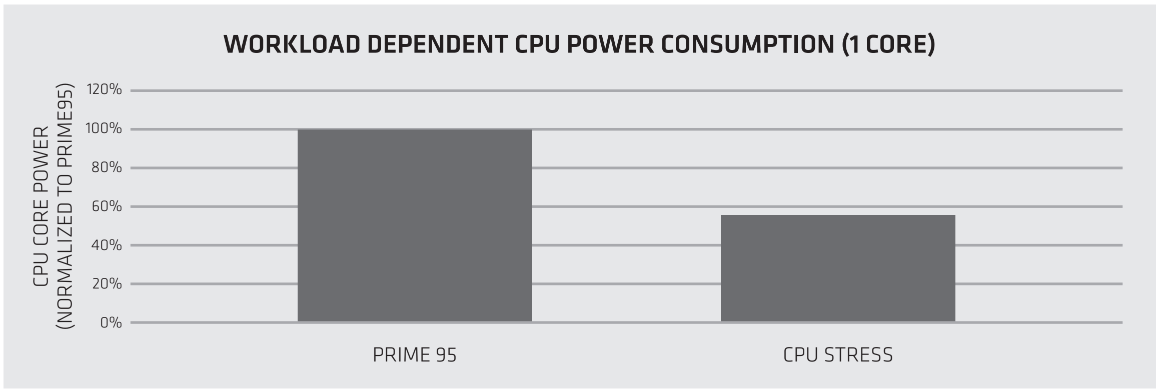


Figure 3 - Prime 95 v29.3 b1 Large FFT, Microsoft SysInternals CPU Stress v1.0

The data in Figure 3 show that the power consumption of the less power-dense workload was only 57% of Prime 95 with a single CPU core active. When extrapolated across multiple physical cores, it is easy to see that power variation by workload can grow quite large. In this test case, the CPU was able to maintain maximum frequency (i.e., 3.5GHz) on the active core without reaching power or current throttling, so no frequency reduction was required.

The power density of GPU workloads can be compared in the same way. The graph below compares a simple 3D workload from the Microsoft DirectX 9 SDK ("blobs") to Furmark, an extreme GPU workload falling in the thermal virus class. GPU frequency was artificially limited to 720MHz to avoid power limit throttling and expose the full potential power consumption difference. A comparison of the RX-421BD processor power for both workloads is shown in Figure 4.

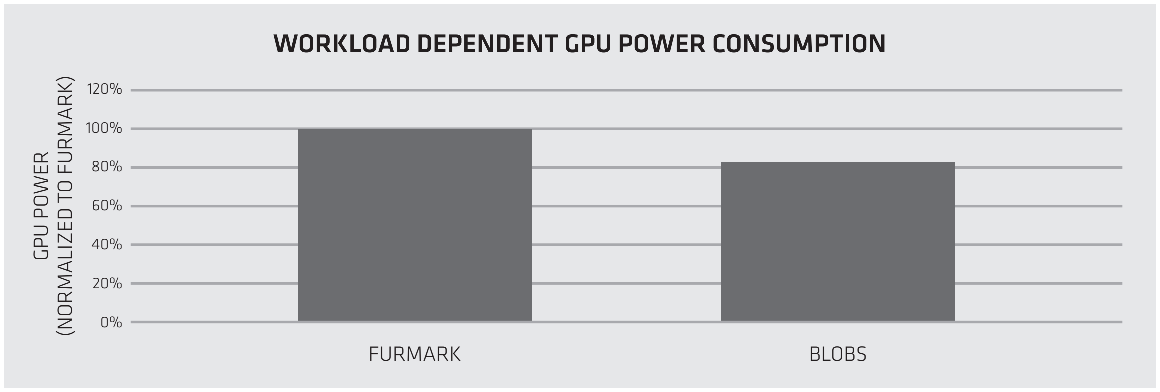


Figure 4 - Furmark v1.18.2.0, Microsoft DirectX 9 SDK "Blobs"

The GPU power data shows the Blobs application consumed only 82% of the power of Furmark, confirming a difference in power density. It is also worth noting that the increase in power dissipation with the heavier workload will raise die temperature in a given system environment. The higher temperature will increase leakage power, adding to the power difference. Truly comparing the power difference caused only by the workload would require tight control of the die temperature which was not attempted in this test. However, the few degrees of difference observed here do not significantly affect the results.

PROCESSOR POWER MANAGEMENT

Previous sections establish the key observation that power is inextricably linked to temperature, frequency and voltage/current. Power management in modern processors is all about controlling these parameters to control power consumption, while maximizing workload performance. Current processors from AMD and Intel contain dedicated microcontrollers that are independent of the x86 processor cores to administer power management. The firmware in the microcontroller is tailored in some ways for the product's intended use case. For example, mobile products will be more aggressive in the use of power saving features like clock and power gating in the interest of improving battery life. Desktop and server processors that are always wall powered will tend to favor performance and only save power when it has minimal impacts on performance.

DEFINING POWER LIMITS

Definition of the maximum power consumption is a common starting point when defining processor models. Manufactures choose power levels to address various use-cases with differing power restrictions, and performance (i.e., frequency) is largely derived from that. X86 processors are largely marketed by their Thermal Design Power (TDP), even though it is a specification related to the thermal solution requirement and not a maximum electrical power that the device can consume. Maximum sustainable power levels will be equal to or greater than TDP, depending on the product. This paper will focus on the maximum sustained power of the processor when discussing it as a limit.

SCENARIO DEFINED PERFORMANCE

The power management controller of the processor monitors key parameters to ensure the processor specifications for maximum power, current and temperature are not exceeded. If changes in the operating scenario cause any one parameter to approach its limit, the controller must throttle the processor's performance to compensate. This throttling usually takes the form of reducing operating frequency of the core(s) consuming the largest amounts of power (i.e., CPU and GPU), as they have the biggest impact. Reducing frequency often allows voltage reduction for additional power savings. Reductions in power consumption will reduce temperature and current, helping the processor to stay within these specifications. These adjustments can happen as much as every millisecond for very quick response to changes in the operating environment or even the workload (Howse, 2015). Previously, x86 processors moved between discrete "performance states" (specific combinations of voltage and frequency at which cores can reliably operate) that differed by hundreds of megahertz and required suspension of execution during transitions. Newer Intel 7th Generation Core Processors and AMD Ryzen Processor architectures allow much more granular frequency changes for better efficiency and, at least in the Ryzen case, uninterrupted execution.

Since power consumption varies with the workload, one can recognize why achieving maximum frequency of a core may not always be possible. What if a very power dense workload is run on a CPU core at maximum frequency and causes the device to exceed its power limit? What if that workload is then run on multiple cores further exceeding the limit? What if a graphics workload is suddenly introduced on the integrated GPU simultaneously? In these cases, the power management controller has no choice but to throttle frequencies to maintain power and current limits. Many system designers erroneously assume that processor manufacturers configure their products to ensure that cores can sustain maximum frequency for any workload in all configurations. This is definitely not the case. Doing so would require these vendors to continuously search out the worst-case (i.e., most power dense) workload in existence, characterize the power usage on their architecture, and set the product's maximum frequency low enough to accommodate it safely across all units of that model (including their part-to-part variations). This “fixed frequency” model is no longer used by most x86 processors in the PC and embedded spaces. Ignoring the fact that the worst-case workload could keep changing over time, the reality is that defining the max frequency in this way would be extremely limiting and easily reduce the operating frequency of a CPU core to a fraction of its potential because of the wide variance in workload power density. The consequence would be that lighter workloads with less power density would also be limited to this reduced frequency, even if it would have been safe to execute them much faster. An artificial performance limitation would be created to guarantee a predictable maximum frequency that is achievable for all workloads. A better approach for general-purpose processors is to define the max frequency by silicon capability and allow the power management controller to dynamically provide the best performance possible for the specific operating scenario in real-time.

Designers should remember that the operating scenario not only includes the workload (i.e., the exact instruction sequences running on processor cores) but also its timing and usage of integrated peripheral functions and I/O. With high levels of integration in modern processors, I/O power cannot be ignored (in this instance, logic power for the I/O interfaces will be put in the same category as the power used by the physical I/O pins). Interfaces like system memory, Serial-ATA, Ethernet, PCI Express, audio, and USB are commonly integrated and they all consume power. I/O power is largely dependent on the system configuration and usage model. For example, a network gateway device may not implement any SATA devices, while a network attached storage (NAS) system may have many. The NAS unit use-case will involve lots of ethernet activity (increasing power used in that logic), while a machine controller may have very little. The portion of the total power envelop consumed by I/O can’t be used by compute cores, so changes in configuration or usage model can impact achievable core performance when processors are power limited. Including the system configuration and I/O usage model in the workload definition is key when attempting to improve performance determinism.

EXPLOITING DEVICE VARIATIONS

The natural result for the power limited (versus fixed frequency) model is that performance is maximized for each workload, but frequency is not predictable with workload changes. Any scenario where the workload reaches temperature or power-limit throttling, performance can be degraded from the fixed-frequency model. System designers can avoid temperature throttling by developing enough headroom into the thermal solution to ensure maximum temperature is never reached. After all, the maximum sustained power level is a known quantity and airflow and ambient temperature limits can be specified for the final system. Power throttling is a more difficult challenge due to the part-to-part variations discussed earlier that affect power consumption. Two samples of the same processor model could have differences in their leakage power, causing one unit to reach its power limit at a lower average frequency even when running an identical workload under identical operating conditions. Vendors happily exploit this difference by allowing the lower leakage units to spend more time at higher frequency, yielding better performance. Earlier discussions of voltage dependencies reveal why different processor units of the same model can also have different voltage requirements to achieve a given clock frequency. This difference can be exploited by fusing unit-specific voltage vs. frequency curves into each part that enable the power management controller to minimize core voltage. Reductions like this to active power allows those units to further increase average frequencies before reaching power limits. Fortunately, lower leakage devices tend to also require higher voltages to reach the same frequency as a higher leakage device, so these two factors work to cancel each other out rather than compound. Despite this, material differences in the consumed power can remain.

Many real-world PC use-cases have been found to be bursty, where applications often sit idle waiting for user input and then perform some activity before waiting again. This could be a user starting a program or loading a new web page. Periods of inactivity will naturally coincide with low power and lower die temperature. Some processors take advantage of this situation by defining a maximum power limit that is greater than the sustained power limit. The processor can be allowed to reach this higher power consumption for a short amount of time that is “thermally insignificant”. Thermal solutions have a relatively large thermal inertia, meaning it takes a while for the processor to raise their temperature to a steady state value. Increasing the power limit in this way allows for short periods of increased performance benefiting bursty workloads, but at the cost of performance determinism. The operating environment now has another mechanism by which to affect performance, and workloads may have to run for several minutes to reach a steady state behavior.

REGULATOR TELEMETRY

Since processor performance limitations boil down to power in so many ways, accurately determining power consumption is critical to maximizing performance. Measuring processor power in real time requires accurate current sensing which is not practical for implementation on high-speed digital process technologies.
Until recently, processor power management technology relied on power curves derived from actual power measurements at manufacturing test time with a reference workload. Values were programmed into the processor and combined with run-time data from complex activity monitors in the logic. Management algorithms calculated power usage to ensure power limit adherence. This method allowed some exploitation of part-to-part variations but still required moderate guard-banding due to inaccuracies of the activity monitors to estimate power. Conservative estimations of power consumption leave performance headroom untapped. A recent change seen with AMD processors is use of power telemetry data from the regulators powering the primary voltage rails. Real-time voltage and current data allows the power management unit to be much more accurate in its total power calculation. Doing so enables every variation of the unit that affects power consumption to be factored in along with instantaneous environmental circumstances (i.e., temperature) and exploited for performance gain. Naturally, maximizing performance in this way increases non-determinism across units.

REDUCING EFFECTS OF POWER MANAGEMENT ON PERFORMANCE DETERMINISM

Maximization of performance at the cost of determinism works well for consumer use-cases where the user does not rely on repeatable performance across multiple systems. Enterprise and embedded systems can be quite different and may not be able to tolerate performance variations across units. However, it important to differentiate the need for a minimum performance versus true performance determinism. For digital signage or casino gaming machine examples, a minimum performance need likely applies. Functionality and user satisfaction are not affected if the frame processing time varies slightly across units as long as it is fast enough to meet the level of the content (e.g., 60fps) in all cases. Units that complete work faster may simply spend more time idle between frames, which would be unobservable to the user. Special cases like industrial machine controllers or military applications may require truly repeatable performance due to sensitive timing interactions. Even some datacenters desire such repeatability so that job execution time can be predicted regardless of which system it is scheduled on. It should be clarified that true hardware determinism is not possible with modern x86 CPU architectures. Small timing variations can exist because of interactions between hardware and software, and hardware interrupts can occur with unpredictable timing. Some amount of variation will always exist, but there are ways to improve the situation, particularly for variations caused by power management. One thing that system designers can count on is that improving performance determinism will come with a cost to peak performance.

MINIMUM PERFORMANCE LEVEL

Ensuring a minimum performance level begins with testing in a worst-case environment. The specific workload of interest must be run on a worst-case processor sample operated at maximum temperature. The frequency behavior of the processor and the resulting performance of the workload should represent the lowest level of any sample in the distribution. If the performance is still acceptable, then the processor model choice is sufficient. System designers can have confidence that all samples of the chosen model will perform at this level or better. Of course, changing processor models or modifying the workload means testing must be repeated. Unfortunately, worst-case samples are rare and processor vendors don't usually supply them upon request. Holding the processor die under tight temperature control while running an active workload can also be difficult, and usually requires a specialty thermal equipment like thermal stream blowers or oil baths. Many embedded designers will need an alternative method to ensure their minimum performance level.

Most x86 processors sold today specify a base and boost frequency for CPU cores. A few models even do the same for integrated GPUs. A good rule of thumb has been that base frequency should be sustainable for all processor samples, but designers must understand when this can be broken. Processor vendors generally do intend for base frequency to be sustainable on all cores of a CPU under "real world" workloads. Differentiation is made because worst-case power density of real versus synthetic applications can be very large. The example provided earlier used two synthetic workloads for uniformity of power usage but it still illustrates the point. Real applications tend to have a mixture of compute, memory, and I/O operations while synthetic “power viruses” can intentionally loop on small, power dense instruction sequences. As previously discussed, defining base frequency with the most power dense workload available would be extremely conservative and would artificially limit performance of more typical workloads. The catch is that definition of “real-world” is subjective and varies by vendor and product. Vendors will choose reference workloads to represent the worst-case of the real-world, and then use it to define base frequency of the product. If the reference workload is known, both it and the custom workload can be compared on a random sample. If the power density of the custom workload is less than the reference workload, then it should be able to sustain base frequency across all units of the model distribution. To get an accurate measurement, testing should be performed on a specific sample/system in a fixed configuration and environmental conditions. CPU core frequency boost should be disabled to prevent reaching temperature or power/current limits (boost disable is commonly available in system BIOS[1] firmware options of most x86 platforms). If either workload can reach these infrastructure limits then results will be skewed. Both AMD and Intel provide tools to log processor power, but they do not publicly disclose reference workloads. Such information must be obtained under non-disclosure agreements. If comparison is successful and the custom workload’s power density is less than the reference workload, then the performance of the boost-disabled scenario should be achievable across all units. Capping frequency in this way does reduce peak performance, but that is the sacrifice required for consistency.

If characterization of the custom workload reveals it is more power dense than the reference workload, then further frequency reduction is required to ensure minimum performance across units. In addition to disabling boost states, frequency should be reduced until power/current measurements are below those of the reference workload on the test unit. Once a suitable frequency limit is determined, performance can be evaluated for acceptability. A challenge with this case is that setting a CPU frequency limit below base frequency can require more invasive software modification. The Linux kernel supports a simple software daemon (e.g., cpufreqd) that can set core operation at a specific P-state and this mechanism can be used to limit CPU frequency with some processor architectures. For Windows operating systems, custom modifications must be made to the ACPI[2] PSS table in BIOS that communicates supported CPU P-states to the operating system (OS) (Unified Extensible Firmware Interface Forum, 2017). Higher, unwanted frequency states can be removed, and the table rebuilt. The same method can be used for other operating systems that support the ACPI _PSS table. Modification of the table takes significant BIOS expertise and access to source code. Once a P-state is identified that brings power density of the custom workload below that of the reference workload, performance can again be evaluated for acceptance. It should be noted that this method of comparing power density is less exact than the ideal method of using a true worst-case sample. Including some reasonable margin into the operating point is wise.

Mixed workloads that highly utilize both CPU and GPU cores simultaneously complicate the ability to confirm if a custom workload is more power dense than the reference workload. If a GPU base frequency is defined at all, reference workloads for CPU and GPU are likely measured independently so there is not significant interaction. Using the workload power density comparison method would also require tools to provide power data separately for each core type, which processor vendors do not typically provide. Workloads of this type cannot establish a reliable minimum performance operating point without assistance from the processor manufacturer. In cases where GPU performance is not critical, its maximum frequency could be set to a very low value in the interest of limiting its contribution to processor power and possibly avoid power/current throttling of CPU cores (AMD integrated GPUs use a vendor-specific “PowerPlay” table in BIOS to define frequency states for the GPU, much like the ACPI PSS table for CPUs; they can be edited, but this approach is not universal to other vendors). However, without a way to quantify the power usage to a reference point, ensuring repeatable performance requires adding a large, arbitrary guard-band to core frequencies. There will always be some uncertainty about coverage for worst-case samples.

DETERMINISTIC PERFORMANCE

Any system that desires performance determinism from the processor will need to start disabling power management features to get as close as possible to the old fixed-frequency model. The first to go are those features that provide temporary performance improvements based on real-time environmental factors. Examples discussed earlier include temperature-based boosting or time-based power excursions (e.g., AMD STAPM[3] and

1 Basic Input / Output System
2 Advanced Configuration and Power Interface Specification
3 Skin Temperature Aware Power Management

sub-features of Intel DPTF[4]). These features often don't provide value anyway for embedded use-cases where a workload is run continuously. The ability to disable them may not be exposed in off-the-shelf embedded platforms, but most can be turned off by the system BIOS developer. Review the processor documentation thoroughly to understand which of these features can be disabled.

Frequency boosting is also scenario driven and therefore must be disabled. Even if a frequency in the boost range is sustainable for the custom workload on a worst-case sample, current processors do not allow fixed frequency operation in this range. Operating systems are unaware of boost frequencies, so there are no OS-level mechanisms to set them. To fix CPU frequency, a value at or below base must be used. From there, evaluation of the workload power density compared to the reference workload can be used to determine if CPU frequency must be further reduced below base frequency. The method described earlier still applies.

After the new maximum frequency has been established and set, states below this must also be eliminated to ensure fixed-frequency behavior of cores. Latency is increased when cores transition to low frequency during idle periods, reducing deterministic behavior. As described in the previous section, OS frequency governors can be used with Windows and Linux to set "performance" mode which will ensure hardware does not go below base frequency. This method has been verified on AMD Embedded R-series and G-series processors, as well as Intel 7th Generation Core processors. Excursions below base frequency can still occur if triggered by thermal throttling, but proper design, as outlined here, can prevent it. Despite being effective for Windows and Linux, designers requiring determinism will likely be running a real-time OS. RTOSs with support for ACPI PSS tables can still use that method. Others with no CPU P-state management must rely on the platform BIOS to set the desired state before the OS handoff.

If the workload is mixed for CPU and GPU, the same complications previously discussed apply. Extreme guard-banding could ensure fixed frequency operation across all units of a model distribution, but there is no reliable method to confirm that without manufacturer support. For applications that are willing to go the extra mile to secure deterministic performance, custom screening can be implemented where each sample is pre-tested to ensure operation within specific limits. Obviously, this kind of screening is very costly in both infrastructure and labor but has found use in military markets where cost sensitivity is low.

VENDOR PROVIDED HARDWARE DETERMINISM

AMD has recognized the demand for improved determinism in some enterprise and high-end embedded applications and has introduced dual operating modes in their EPYC line of enterprise processors to address differing needs. A "Power Determinism Mode" offers higher performance by taking advantage of many of the mechanisms described earlier in this paper including part-to-part variations (Fruehe, 2017), though server processors are more conservative in this area. The processor will exploit some of these differences to reach a maximum (i.e., deterministic) power consumption for a given workload (at a given temperature) and thus maximize performance while maintaining infrastructure power limits. "Performance Determinism Mode" offers the unique ability to achieve the same performance with every processor of a given TDP. Creating repeatable performance requires part-specific power and frequency curve data to be fused into the device at production time. This data essentially provides a negative performance offset that can be used to make each individual unit replicate the performance of a worst-case unit of the entire model distribution. The power management controller also uses the predictable calculated-power method based on activity monitors instead of regulator telemetry data. Enablement of the feature is a simple compile-time option in the BIOS firmware. In performance determinism mode, part-to-part variations will only result in differences in power consumption for a given workload (at a given temperature) while performance (derived from frequency behavior) is minimally impacted and remains consistent. This type of reliable hardware determinism can only be provided by the processor manufacturer and it ensures that only the minimum necessary performance sacrifice is made to achieve that determinism. The performance deterministic mode certainly simplifies system architecture for designers looking for improved performance determinism for enterprise applications and its existence is noteworthy given the topic of this paper. However, it is yet to be seen if this kind of feature will find its way into lower-power embedded processor products from AMD or Intel.

4 Dynamic Power and Thermal Framework

REFERENCES

Cutress, I. (2017, February 22). AMD Launches Zen. Retrieved from Anandtech.com: http://www.anandtech.com/show/11143/amd-launch-ryzen-52-more-ipc-eight-cores-for-under-330-preorder-today-on-sale-march-2nd

Fruehe, J. (2017). Power / Performance Determinism. Moor Insights and Strategy.

Gielen, E. M. (2013). Analog IC Reliability in Nanometer CMOS. Analog Circuits and Signal Processing, DOI: 10.1007/978-1-4614-6163-0_2, 23-28.

Howse, B. (2015, Nov 26). Examining Intel's New Speed Shift Tech on Skylake: More Responsive Processors. Retrieved from Anandtech: https://www.anandtech.com/show/9751/examining-intel-skylake-speed-shift-more-responsive-processors

Kaushik, S. a.-M. (2003). Leakage Current Mechanisms and Leakage Reduction Techniques in Deep-Submicrometer CMOS Circuits. IEEE.

Unified Extensible Firmware Interface Forum. (2017, May). Retrieved from Unified Extensible Firmware Interface Forum: http://www.uefi.org/sites/default/files/resources/ACPI_6_2.pdf

Wolpert, & Ampadu. (2012). Managing Temperature Effects in Nanoscale Adaptive Systems. In D. Wolpert, & P. Ampadu, Managing Temperature Effects in Nanoscale Adaptive Systems (pp. 22-24). Springer.

7.12.18

AMD.com/en/corporate/speculative-execution

AMD