PowerVR virtualization:
a critical feature for automotive GPUs
What is GPU virtualization?

Conceptually, virtualization is the capability of a device to host one or more virtual machines (VMs) that each behave like actual independent machines with their own operating system (OS), all running on the same underlying device hardware. In regard to GPUs, this means the capability to support multiple concurrently running operating systems, each capable of submitting graphics workloads to the single graphics hardware entity.

GPU virtualization is now a must-have for a range of next-generation applications, from automotive, to consumer electronics, to the IoT. GPUs that implement hardware virtualization can provide isolation between the various applications/OSs for increased security, as well as maximum utilisation of the underlying GPU hardware.

Imagination’s PowerVR GPUs, from Series6XT onwards, support hardware virtualization and in Series8XT its capabilities have been further enhanced.

In this paper, we will first discuss the fundamentals of virtualization and then describe the specifics of PowerVR’s GPU hardware virtualization solution, highlighting its unique features and their particular relevance for the automotive market. We will also provide several demonstrations of this powerful technology.

Three key entities in a virtualized solution:

- **Host OS**: a virtual machine with an OS, which has a full graphics driver stack and higher control capabilities to the underlying hardware compared to the guest operating systems.

- **Guest OS**: a virtual machine with an OS each, which the hypervisor hosts. There can be one or more guest OSs that shares the available underlying hardware resources. Each of the OSs will have a full graphics driver stack.

- **Hypervisor**: fundamentally the software entity that presents the operating systems with a shared virtual hardware platform (in this case the GPU hardware) and manages the hosting of the operating systems.

Introduction to types of GPU virtualization

There are two types of GPU virtualization discussed in this white paper –

- **Paravirtualization**: where the guest OS ‘knows’ that it is virtualized and shares the same underlying hardware resource with other guests. In a paravirtualization solution, guest operating systems are required to submit tasks via the hypervisor and the entire system has to work together as a cohesive unit. This solution has high hypervisor overhead (running on the CPU), and long latencies in task submission, potentially reducing the effective utilisation on the underlying GPU hardware. There is also a need to modify the guest OSs (add further functionality) to enable them to be able to communicate via the hypervisor.

- **Full hardware virtualization**: where each guest OS is running under a hypervisor and acts as if it has exclusive access to the GPU and has no awareness that it is sharing it with other guests and the host OS. Each guest generally has a full driver stack and can submit tasks directly to the underlying hardware, in an independent and concurrent manner. The advantage of this approach is that there is no hypervisor overhead (running on the host CPU) in handling task submissions from the different guests, and this in turn reduces the latency in task submission to the GPU, ultimately leading to higher utilisation.

A third variation – complete software virtualization, which essentially involves the software emulation of the required system, OS and hardware functionality, is not addressed in this paper, due to its various limitations, making it a non-preferred solution for GPU virtualization.

GPU virtualization use cases

Before we go into detail about Imagination’s PowerVR virtualization solution we will first describe some of the real-world use cases for the technology of which there are a number across a wide array of markets and applications. Most focus on the embedded market and encompass:

- **automotive**
- **DTV/set-top box (STB)**
- **IoT/wearables**
- **mobile/tablets**

This section focuses mainly on two of these applications – automotive and DTV/STB. These are areas today where there is a major need for, and increased adoption of, full hardware virtualization.
**Automotive use case:**

GPU virtualization is becoming a must-have for the automotive industry. As cars become increasingly autonomous, there is an increased requirement to support more advanced driver assistance systems (ADAS) functionality. This is driving the need for more powerful GPUs due to the large parallel computing capabilities demanded by these applications.

At the same time, there is an increasing trend towards having more high resolution displays – for cluster, infotainment (dash and rear seats), heads-up display (HUD), etc.

Historically, these disparate applications were handled by separate chips, or a single chip running software or paravirtualization. In a desire to reduce costs, the Tier-1s/OEMs are moving towards more powerful single-chip solutions. Software or paravirtualization on a single chip, though an option, has the downside of performance/utilisation degradation, higher power consumption and lack of robustness and security, with the latter of critical concern to the automotive industry.

Full hardware virtualization helps solve these issues, ensuring that there is isolation between the various applications for increased security, as well as maximum utilisation of the underlying GPU hardware.

To understand more about the need for GPUs in today’s cars, where the market is going and what Imagination’s thoughts are on the market as a whole, please refer to the following pages:

**The benefits of PowerVR for automotive applications**

**PowerVR Automotive market page**

As shown in Figure 1, each of the virtual machines/guest OSs on which the automotive applications are running are isolated from each other, yet are utilising the same underlying GPU. Some are critical and secure applications (running in separated virtual machines) such as dashboard and ADAS applications, which are generally run securely with a guaranteed level of performance. Other applications such as the in-car infotainment apps need to be secure due to Digital Rights Management (DRM). Consumers are also likely to want to run non-critical and non-secure apps downloaded from third-party stores. To avoid malicious attacks and copying of secure content, these therefore need to be kept separate from the applications that require security, all of this can be achieved by PowerVR's full hardware virtualized solution.

**DTV/STB use case**

GPUs have historically been used in DTVs/set-top boxes (STBs) mainly for rendering UIs and casual games. Today, we are seeing increased use of GPUs for video surface post-processing and composition. In the past, this use was not approved due to video protection/DRM content protection concerns (i.e. offering up the possibility of copying video surfaces when passed through a non-secure IP/part of the system). However, some GPUs are now built with hardware implemented support for the protection of DRM content. The GPU works as a part of the system, accessing the DRM content from a protected region in memory and writing the processed content back to this protected memory region. PowerVR GPUs have supported hardware implemented DRM security from Series6 onwards. This enables the isolation of secure and non-secure applications, ensuring that application content in memory can’t be copied to the non-secure memory regions.

Though the solution serves its purpose, there is a growing need to provide further isolation between multiple secure applications. For example, the
content from provider A requires isolation from that of content provider B. This need is driven by the content providers and their suppliers/vendors. There is also a need to separate broadcast TV from other downloadable applications, whether they are secure or not. To achieve this, PowerVR's full hardware virtualization support can be utilised to achieve true isolation between multiple secure applications on a single OS (or multiple OSs). An example of a DTV/STB system with multiple applications, some of which need to be protected/fully isolated, is illustrated in Figure 2.

PowerVR's full hardware GPU virtualization solution

PowerVR GPUs have long led the industry in technical capability, and the inclusion of hardware virtualization is an example of this innovation. PowerVR's GPU virtualization is a full hardware virtualization solution, where the guest OSs have a full driver stack each and can directly submit tasks to the GPU hardware. The solution does not require hypervisor intervention for task submission, resulting in the maximum utilisation of the available GPU resources. PowerVR GPUs can support up to eight virtual machines/operating systems, each of which can be running independently and in parallel.

In the example shown in Figure 3, each operating system can submit one or more tasks to the underlying hardware simultaneously, and each OS is isolated from the point of view of software, hardware and memory content, inherently bringing about robustness and security.

Once the OSs submit their tasks to the GPU hardware, the firmware running on a dedicated microcontroller integrated within the GPU handles the actual scheduling of the workloads on the hardware. The scheduling mechanisms will be explained in detail in the later sections, in terms of the multiple schemes in place to ensure maximum performance, robustness, isolation etc.

There are two mechanisms in place to enable the solution described in Figure 3.

Submission of the task via per OS hardware scheduling interface: Within the GPU there are task submission (kick) registers per OS, which are accessible via the register interface between the host and GPU. These registers are written for task submission to the GPU and each is mapped to its relevant OS address space, making it accessible for job submission to only the correct OS. Each OS can use this mechanism to kick tasks directly on the hardware without intervention from the hypervisor. This inherently brings about security/isolation as the guest OSs can only

© Imagination Technologies Limited
access their own kick register, and cannot access any of the configuration registers, which are only accessible by the host driver/hypervisor.

**Access to memory with OS_ID identifiers:**
With each memory transaction from the GPU a unique ID per OS is propagated and this serves as a unique address space selector per OS. For example, in the case where there is a system memory management unit (MMU) in the SoC, the GPU's physical address becomes the intermediate physical address (IPA). When combined with this unique ID the IPA enables the system MMU to allow or block the access and to convert the IPA to the true physical address. This approach, with the addition of some sort of firewall in the system, inherently provides physical protection of all resources across OSs in memory.

**Demonstrating maximum utilisation**
As mentioned previously, the locally present firmware handles the scheduling of the workloads submitted by the different OSs. The main goal of the firmware is to achieve maximum utilisation of the GPU resources, hence leading to higher performance, while adhering to the different mechanisms in place for robustness and security. A video demonstrating this, i.e. maximum utilisation of a fully hardware virtualized PowerVR GPU is available on the following link:

**Video 1: Two Dashboards Equal Priorities**

This demonstration is carried out on a reference platform that uses a PowerVR Series6XT GPU. In this example, there are two applications submitted from two separate OSs, running on the same virtualized PowerVR GPU hardware.

In the initial part of the video, both of the OS's workloads are given the same priority and hence are scheduled on the hardware by the firmware in a round-robin fashion. It can be seen that they are sharing the GPU resources equally as they are achieving the same performance (quantified in frames per second – fps). Further, on analysing the hardware counters within the GPU, it can be seen (in Figure 4) that the hardware is utilised to its maximum extent (no significant gaps), demonstrating that PowerVR’s virtualization solution does not add overhead that would result in reduced utilisation.

This hardware counter-based utilisation analysis is done using PVRTune, Imagination’s performance analysis software. This is a publicly available, world-class GUI-based software tool used to access, collate, and visualise the GPU’s performance in real-time, using hardware counters and timers.

It provides high-level information such as the fps, GPU and CPU utilisation and frequency, but can also provide in-depth data such as texture and ALU pipeline utilisation, bandwidth consumption, per context ID, vertex and pixel processing cycles, etc. These are based on hardware counters present within the GPU and can be sampled at a regular interval and configured by the user. This is a useful tool for developers to check their application performance by identifying bottlenecks if applicable and optimising accordingly.
In the second part of the video, (starting at 16 seconds) the performance of the app from OS1 (displayed on screen 1) is artificially limited to a maximum frame rate of 120, 90, 60, 30fps etc. It can be seen that there is a corresponding proportional increase in the performance (fps) of the OS2 app, which doesn’t have a fixed framerate set. This demonstrates that in this scenario too the firmware is able to schedule efficiently and achieve maximum utilisation.

The last part of the video (from 52 seconds in) emphasises this further. The OS1 app framerate is limited to 10fps, and the free running OS2 app framerate is 247fps. When the OS1 app framerate is then limited to 20fps, the free running OS2 app framerate is 237fps, as expected. The hardware counter plot of this is shown in Figure 5.

This proves that when virtualizing PowerVR GPUs there is negligible overhead and maximum utilisation of resources.

**PowerVR virtualization – geared for automotive**

PowerVR's GPU virtualization solutions cater to the complex needs and strict restrictions of the automotive industry.

**Basic isolation**

One of the benefits of virtualization, in general, is the isolation it can provide between the different OSs and their corresponding applications at a software level. This is a basic requirement for automotive applications, where a non-critical OS's application failure can't impact a critical OS application. This is demonstrated in Video 2, available at this link:

**Video2: Kernel Panic and OS Reboot**

It shows a reference platform that uses a fully hardware virtualized PowerVR Series6XT GPU. There are two applications, submitted from two OSs: the critical OS running the cluster application and the non-critical OS running the navigation application.

The non-critical OS application is made to artificially crash, followed by a kernel panic and a full reboot of the guest OS. Because of virtualization in the GPUs, this doesn’t affect the critical application running from the critical OS — it continues to render uninterrupted. Furthermore, once the guest OS reboot has completed, it is able to again seamlessly submit jobs to the GPU.

**Quality of service**

There is a concept wherein a critical OS's applications (one or more) are required to be protected against malicious applications and also have a guaranteed level of performance. This can be achieved by supporting context robustness. This involves having mechanisms in place to protect against denial-of-service (DoS), i.e. protection from attack of a malicious app, where, for example, it deliberately consumes all the resources, preventing the critical OS app from executing. These mechanisms are supported in PowerVR GPUs and are described overleaf.
**Prioritisation**

To cater to the guaranteed level of performance required by many automotive systems, there is a need for prioritisation mechanisms. The firmware, running on a dedicated microcontroller within the GPU, handles the scheduling of workloads on the hardware as previously described, with the intent of maximising the utilisation and performance of the GPU. This entity also helps honour the prioritisation criteria set for each OS and further within the OS at a workload granularity level. When a higher-priority OS's workload is submitted to the GPU, the lower-priority OS's workload is context-switched out. Context switching, in simple terms, is the pause of the current operation at the earliest possible point and the writing out of the required data to enable the resumption of the operation at a later point. The earliest possible point, i.e. the minimum granularity of context switching for Series6XT (the first generation of PowerVR GPUs which supported full hardware virtualization) is:

- **Geometry processing: draw call granularity**
- **Pixel processing: tile granularity**
- **Compute processing: workgroup granularity**

Once the higher priority OS's workload has completed, the lower priority workload is resumed. This feature helps ensure that the critical higher-priority OS's workloads get the GPU resources needed to guarantee the required performance. This is demonstrated in the video available here:

**Video 3: Performance and Priorities**

The demo is running on a reference platform using a fully hardware virtualized PowerVR Series6XT GPU.

At the start of Video 3 there are two OSs, of which one requires a guaranteed performance level of 60fps. This is the critical OS and is accordingly marked as high-priority. When the required performance level is further increased to 90fps the performance of the navigation application (set as the lower-priority OS) decreases. This is due to the priority mechanism described above; i.e. the navigation application is context-switched out whenever the high-priority workload is submitted to ensure the latter achieves its required performance. Again, when the required fps of the high-priority OS is further increased to 120fps; the navigation app performance decreases even further to ensure the critical OS has the GPU resources to achieve the required performance.

In the second part of Video 3 (at 50 seconds), for functionality demonstration purposes, the high-priority OS's required fps is reduced to 10fps and then 20fps, and, as expected, there is a corresponding increase in the navigation app performance. Furthermore, when (at 1m 10) the high-priority OS is set without a required fps and is hence free running, it consumes the GPU resource due to its priority being higher, and, predictably, the rendering of the navigation app stalls completely.

In the final part of the video (1m 30), the priority of both OSs is set to be the same. As previously mentioned, the workload submission by the firmware to the hardware for equal priority workloads happens in a round-robin fashion, depending on the submission rate from the host side. In this case, even though the cluster app is set to run without a required fps (hence free running/max) it still shares the GPU resources equally with the navigation app, and this reflected in the navigation app fps change.

**Denial of service**

The above prioritisation works as long as all of the applications running from the different OSs are well behaved. However, a malicious app could be written in such a way to make it impossible to perform a context switch out, in order to schedule the higher priority workload. For example, a malicious pixel shader could be set to be in an infinite loop and as a pixel is below the granularity of a tile a context switch-out will not happen.

To protect against such scenarios, with PowerVR GPUs it is possible to define the the maximum context switch period. If the context switch hasn’t completed, a per data master kill or soft reset is executed (depending on the type of workload). The difference between a per data master kill and a soft reset is that the former does not result in
the disruption of other workloads running on the GPU at the same point in time, whereas a soft reset will disrupt all running workloads. For the reference platform used (PowerVR Series 6XT GPU) in this demonstration, compute workloads can be killed, whereas vertex and pixel processing operations will be soft reset. This feature allows for quality of service, as the GPU can be freed from malicious applications, hence enabling a guaranteed level of performance. The time period for the context switch is configurable, delivering full flexibility and customisation for the Tier 1 or OEM.

**OS isolation**

Though the above solution works to ensure that the GPU can protect against malicious applications, there is an optional additional feature in PowerVR GPUs to take the robustness aspect a step further – OS isolation. Here, only the OS deemed as critical and safe will be run on the GPU in complete isolation. For example, assume the dashboard application was developed in a closed environment (and hence is safe) and needs a guaranteed level of performance and protection from malicious applications. It can be run on the GPU without allowing a third-party and therefore potentially unsafe/malicious application (for example, downloaded from an app store) on a guest OS, to also run at the same point in time. This ensures that there is true isolation even from the hardware perspective. The unsafe application/s can be run once the ‘safe’ application completes, in isolation. The isolated app can run faster, as the GPU and memory subsystem resources (bandwidth) aren’t shared. This solution can also be extended to multiple applications from a safe OS or multiple safe OSs. It is dependent on what the OEMs deem safe; there is total flexibility and configurability from the GPU.

**Taking it a step further:**

**Virtualization in PowerVR Series8XT**

Imagination’s PowerVR Series8XT GPUs are the first GPUs based on Imagination’s Furian architecture. This powerful new family of GPUs implement Version 3 of PowerVR’s hardware virtualization support. Some of the key advancements are as described in this section.

**Finer grain context switching**

With Series8XT GPUs, context switching can be executed at a finer level of granularity, ensuring even faster context switching out of the lower-priority workloads - and scheduling of higher-priority workloads. The context switch granularity is now at:

- **Vertex processing:** primitive granularity
- **Pixel processing:** primitive block within a tile – or worst case back to tile granularity

**Per data master killing**

In the case where the lower-priority app doesn’t context switch out within the defined timeframe, there is a DoS mechanism as described in previous sections to kill or soft reset the app, depending on the data master (compute, vertex or pixel processing). Previous generations of PowerVR GPUs supported only compute killing, whereas vertex and pixel processing required a soft reset, hence impacting the high-priority workload if it was being run in overlap with the unsafe lower-priority application. In Series8XT, all data masters can be killed, ensuring that even if a high-priority/critical workload overlaps with an application which needs to be evicted, it won’t be affected.

**Tightly integrated second-level MMU**

Previous generations of PowerVR GPUs had a first-level memory management unit (MMU), hence requiring the SoC vendor to design and implement the second level/system MMU or a similar mechanism at the SoC level to support virtualization. Series 8XT has an integrated second-level MMU within the GPU, which brings about the following benefits:

- **Low latency and improved efficiency due to tight coupling with the first-level MMU**
- **Reduced effort and faster time to market for the SoC vendor**
- **Corresponding isolated software for the entity available in the hypervisor**
- **Improved performance and reduced system bandwidth through full/two-way coherency support**
- **A higher level of protection in a virtualized environment and more fine grain (page boundary) security support**

**Per SPU workload submission control**

In Version 3 of PowerVR virtualization, a particular application can be given its own dedicated SPU (scalable processing unit) within the GPU to execute its workloads. This can be beneficial for long-running compute-based ADAS applications, wherein the application can be run on its own dedicated SPU uninterrupted, while other applications use the other mechanisms in place (example prioritisation based on context switching for higher priority tasks) to share the remainder of the GPU resources.

© Imagination Technologies Limited

PowerVR hardware virtualization, 2018
Summary

For many markets where considerations such as robustness and safety are critical, virtualisation support is becoming a key criterion for selecting GPU IP.

The move towards increased levels of ADAS functionality to achieve higher levels of autonomous driving along with the need to drive more high resolution displays in the car is boosting the requirement for fast, power-efficient GPUs with virtualisation support.

As we have shown in this white paper, Imagination’s fully hardware virtualized PowerVR GPU solution is a perfect solution for this. As well as a highly scalable design that can deliver at all required performance points, the hardware virtualization support makes it an ideal solution for those looking to create leading-edge automotive SoC solutions.

The hardware virtualization solution delivers the maximum utilisation of the underlying GPU hardware resources, along with true isolation between applications in both software and hardware, with negligible performance overhead. Furthermore, additional mechanisms such as prioritisation, QoS, DoS and OS separation enable further levels of safety.

The PowerVR Series 8XT, the first GPU series based on the highly performant and power-efficient Furian architecture, takes this virtualization solution a step further, adding new useful features that make it easier for SoC manufactures to create efficient, safe and robust solutions in a highly cost-effective manner.

Further information

Want to know more? You’ll find further information on PowerVR at: https://www.imgtec.com/powervr/ encapsulation graphics, vision and AI.

To access free examples, tools and support for mobile graphics development visit: www.powervrinsider.com

For all other enquiries, simply get in contact with PowerVR: https://www.imgtec.com/about/contact-us/ – we’ll be happy to help.