Core Avionics & Industrial Inc.
400 North Tampa Street
Suite 2850
Tampa, Florida 33602

T: 888-330-5376
F: 866-485-3199
www.coreavi.com

Introduction

In embedded systems, a graphics application is built on top of a user-level graphics library associated with a graphics driver, OpenGL SC 1.0.1, OpenGL SC 2.0, OpenGL ES 2.0 or OpenGL 1.3, which converts high-level commands into low-level GPU commands which are written to a command ring buffer within a GPU. While multiple graphics applications driving a single, or multiple, GPU can be managed at the application level, modern multi-core processors and Real Time Operating Systems (RTOS) provide support for running multiple applications that improve performance, including graphics application performance. Even though GPUs typically have parallel execution paths to drive multiple displays, there are some shared resources, such as the command ring buffer, which need to be managed by the OpenGL driver to support running multiple graphics applications.

This white paper identifies the key architectures enabled by current multicore processors and RTOS to support multiple graphics applications and describe how OpenGL drivers can support these architectures. The architectures presented are:

1) Multi-Threaded (tasks)
2) Multi-Partition (process)
3) Multi-Threaded Multi-Partition
4) Hypervisor

A summary is then provided on specific available solutions.

The selection of which architecture is the best for a given application is beyond the scope of this white paper. While there may be some graphics performance differences, the approach is best made with an understanding of the system level requirements within the capabilities of the selected hardware architecture and operating system.

Starting with the simplest architecture where all applications share a memory space, multi-threaded.

Multi-Threaded

Threads enhance the process model with multiple threads executing concurrently within a partition. Multiple threads can exist within a partition in the same address space while partitions do not share address spaces. The advantage is that when a thread is waiting for hardware resources, other threads can continue to execute taking advantage of otherwise unused computing resources. Another key advantage is that inter-thread communication is fast. The disadvantages are that there is overhead to thread switching and there is no protection between threads.

If all the threads running graphics applications are in the same address space, only a single OpenGL driver instance is needed. A straight forward method to manage multiple threads accessing the single OpenGL driver is to add a method to allocate command and DMA/data buffers for each OpenGL context. Rendering commands would then be loaded into these context specific command buffers and DMA/data

Core Avionics & Industrial Inc.
400 North Tampa Street
Suite 2850
Tampa, Florida 33602

T: 888-330-5376
F: 866-485-3199
www.coreavi.com

buffers independent of other running threads using the OpenGL driver. Serialization would then be used during critical sections of accessing the GPU or accessing global or shared data.

When graphics applications are in different address spaces, typically different partitions or processes, a different type of solution needs to be added to OpenGL drivers. This is covered in the next section on multi-partitions.

Multi-Partition

The major difference between threads and partitions is that partitions are protected from every other partition by the RTOS kernel using memory management. The advantage is that partitions offer a high degree of protection for both security and safety. The disadvantage is the overhead associated with Inter-Partition Communication.

Since partitions do not share memory, a different approach to using OpenGL GPU drivers is necessary. Instances of an OpenGL driver would then need to exist in each address space (partition) as shown in Figure 1. Then an added communication layer is needed between the OpenGL drivers to synchronize and coordinate the shared GPU resources on behalf of the graphics applications.

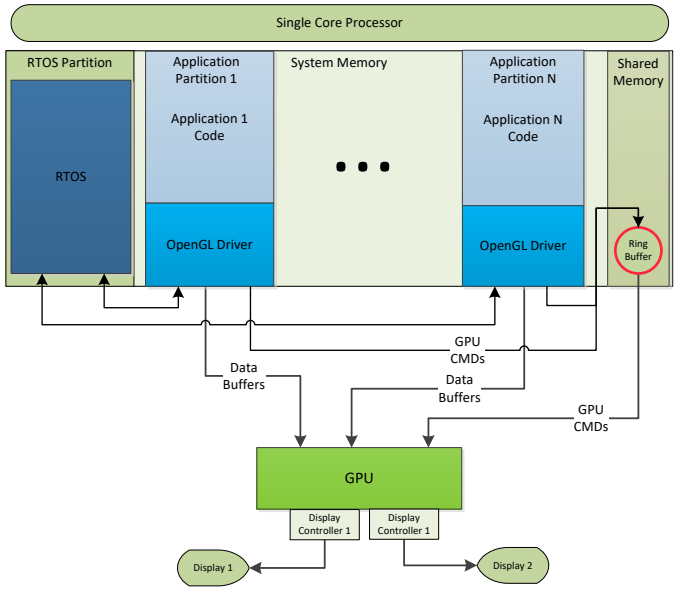

Figure 1 Single Core CPU Multiple Partition Display System

Then multi-threaded and multi-partition can be combined where one or more of the partitions are running multiple threaded graphics applications.

Core Avionics & Industrial Inc.
400 North Tampa Street
Suite 2850
Tampa, Florida 33602

T: 888-330-5376
F: 866-485-3199
www.coreavi.com

Multi-Threaded Multi-Partition

This is a variant of multi-partition where the partitions running multi-threaded applications have an instance of the OpenGL driver with added command and DMA/data buffers for each OpenGL context as described under multi-threaded above.

Advancing beyond traditional multi-threaded and multi-partition methods to run multiple applications, most processors and RTOS now support a hypervisor.

Hypervisor

A hypervisor is software layer between guest operating systems and either a host operating system or the actual hardware. The hypervisor virtualizes the underlying resources to make it appear to the guest operating system as if it has exclusive access. That is, the hypervisor creates virtual machines for guest operating systems. When it comes to graphics, there are three base methods to virtualize the GPU in a hypervisor environment:

1) Emulation: Every read/write access to the GPU is managed by the hypervisor. While technically possible, this approach would severely impact graphics application performance and is generally not done.
2) Pass-Through: Each guest operating system has unfettered exclusive access to the memory registers. While fast, this approach has challenges when it comes to sharing.
3) Para-virtualization: The device driver in the guest operating system is hypervisor aware and works through a GPU virtualization manager module in the hypervisor. This is a similar approach to multi-partition except a separate GPU virtualization manager is required to be included in hypervisor.

With the background on the architectures for multiple application execution and associated needs from an OpenGL graphics driver for each, the discussion now looks at what the available OpenGL driver solutions look like.

Core Avionics & Industrial Inc.
400 North Tampa Street
Suite 2850
Tampa, Florida 33602

T: 888-330-5376
F: 866-485-3199
www.coreavi.com

How to Implement Multiple GPU Applications into an Embedded System

As you can see, the various multi-application execution configurations described above require customization of standard OpenGL drivers. While the industry standard OpenGL API does not change, the underlying communication path to the GPU(s) changes relative to a single application configuration. There are standard COTS drivers available for these configurations that minimize your development effort and cost while speeding time to market. For multi-threaded, multi-partition and multi-threaded multi-partition, Core Avionics and Industrial (CoreAVI) have standard factory build options for embedded and safety certifiable ArgusCore™ OpenGL drivers. A summary of the options is shown in Table 1. There is also a parallel standard factory build option for Symmetric Multi-Processing (SMP) or Asymmetric Multi-Processing (AMP) multi-core processor architectures.

Table 1 Supported ArgusCore Graphics Applications

ArgusCore				
	ArgusCore	ArgusCore MT	ArgusCore MP	ArgusCore MTMP
Single-threaded Graphics Application	X	X	X	X
Multi-threaded Graphics Application		X		X
Combination of single-threaded and multi-threaded Graphics Applications in the same address space		X		X
More than one Graphics Applications each in its own address space			X	X
Multiple GPUs	X	X	X	X

It is recommended that the selected ArgusCore configuration matches exactly the system configuration to minimize driver overhead and thus optimize performance. That is, CoreAVI does not recommend the multi-threaded multi-partition configuration as a catch all for future use as this would impact performance when all the features of this execution configuration are not needed. Since the API presented to the application remains an industry standard OpenGL API, the impact to port an application to another execution environment is minimal and mainly consists of rebuilding the application against the new libraries.

In the case of a hypervisor based architecture, CoreAVI also has a build option for the ArgusCore OpenGL drivers to enable hypervisor awareness. This, coupled with the CoreAVI's HyperCore™ GPU virtualization manager, provides a complete para-virtualization for GPUs in an embedded application. HyperCore is compatible with all leading RTOS hypervisors and includes additional health monitoring features to monitor multi-application GPU usage.

Core Avionics & Industrial Inc.
400 North Tampa Street
Suite 2850
Tampa, Florida 33602

T: 888-330-5376
F: 866-485-3199
www.coreavi.com

With these solutions, your applications continue to interface with industry standard OpenGL graphics Application Programming Interfaces (API) and the CoreAVI drivers manage all the multi-application graphics interfaces to the GPU(s). Furthermore, the ArgusCore OpenGL drivers and HyperCore GPU virtualization manager are available with optional Avionics DO-178C, Automotive ISO 26262 and Railway CENELEC EN 50128 certification data packages supporting up to and including the most stringent levels.

One last item to keep in mind is that the overall factor limiting the number of supported graphics applications is typically each application's requirement for system and video memory.

Where to Find Additional Information on OpenGL Drivers Supporting Multiple Graphics Applications

CoreAVI provides high Technology Readiness Level (TRL) OpenGL SC 1.0.1, OpenGL SC 2.0 drivers, OpenGL ES 2.0 and OpenGL 1.3 with HyperCore GPU virtualization manager with optional certification evidence for avionics, automotive and railway safety critical graphics functions.

More information about CoreAVI's ArgusCore OpenGL drivers, with extensions, along with a comparison of driver features can found here: CoreAVI safety certifiable graphics drivers.

More information about CoreAVI's HyperCore GPU Virtualization Manager can be found here: CoreAVI HyperCore

Contact CoreAVI to find out what we are working on and to discuss your demonstration/evaluation requirements: Sales@CoreAVI.com