ARGUS MULTICORE SUPPORT

White Paper

Document # 11-146-10159

sales@coreavi.com
www.coreavi.com

NOVEMBER 29, 2013

Core Avionics & Industrial Inc.
www.coreavi.com

Table of Contents

1. Introduction

The use of multicore processors is rapidly becoming the only available option for system manufacturers, as single-core processors have reached the physical limits of speed and complexity. If implemented properly, there are significant performance benefits to using multicore processors in the military and aerospace markets. The challenge is developing an intelligent software architecture to unlock the potential use of multicore processing, while maintaining a safety critical approach to supporting Federal Aviation Authority (FAA) and European Aviation Safety Agency (EASA) standards.

Core Avionics & Industrial Inc. (CoreAVI) has successfully developed several unique approaches to enabling high performance and safety critical graphics on embedded platforms using multicore processors. CoreAVI's graphics drivers allows customers the flexibility to use multiple processor cores in either symmetric, asymmetric, or bound multiprocessing environments. CoreAVI's embedded graphics driver architecture enables embedded display manufactures to support the following:

- Multiple independent graphics applications residing across multiple secure partitions bound to assigned processor cores
- Multi-threaded graphics applications across a multicore processor
- Multiple independent graphics applications running on multiple independent guest operating systems with their own dedicated processor core

This technical paper provides a high level summary of CoreAVI's architectural approaches to enabling an efficient, scalable and flexible use of multicore processing with embedded and safety critical graphics applications.

2. Definition of CoreAVI's Argus Graphics Driver

Argus is CoreAVI's graphics code base from which drivers / libraries are created that control various GPUs and exposes various OpenGL APIs to application programs.

Argus exists as either a static or shared library, supports one or several graphics applications, and can be considered a device driver.

As a device driver Argus controls the Graphics Processing Unit (GPU), such as the AMD Radeon E4690 or AMD Embedded G-Series SoC, and the device's display controllers.

As an OpenGL library Argus is used by graphics applications in an embedded environment to display graphics on one or more displays that may be utilizing one or more graphics processors.

Figure 1 is block diagram showing the embedded software stack (green) which includes Argus and the embedded hardware (blue) of a typical embedded graphics system.

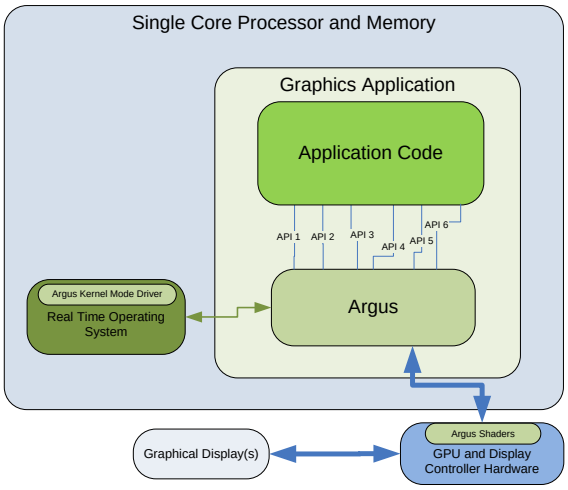

Figure 1 Argus in a Single Core Embedded Graphics System

In addition to OpenGL APIs, Argus exposes an additional five APIs to the application:

- BIT: This API is used to invoke built-in tests and access the fault log.
- EGL: This API is used to create and destroy windows and rendering contexts.
- CoreAVI Display: This API is used to initialize and shutdown the display controller.
- System Initialization: This API is used to initialize and shutdown / un-initialize the driver.
- OS Helper: This API is used to access RTOS features from the application.

2.1. Argus Support for More than One Graphics Application

Two Argus variants are available to support more than one graphics applications. One variant is referred to as Multiple Application / Multi-Threaded support (MAP) and the second is referred to as Multiple Address Space support (MAS).

2.1.1. Multiple Application / Multi-Threaded (MAP)

Argus supports multi-threaded graphics application(s) and multiple single-threaded applications in the same address space as shown in Figure 2.

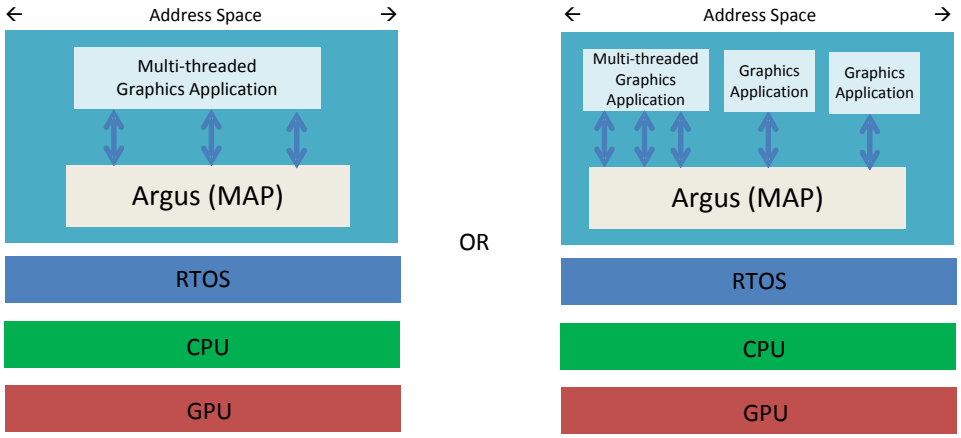

Figure 2 Multiple Application Support

To support this, Argus allocates personal command and DMA buffers for each OpenGL context. Rendering commands are loaded into these context specific command and DMA buffers independent of other running threads using Argus. Serialization is used during critical sections of accessing the GPU or accessing global or shared data.

2.1.2. Multiple Address Space (MAS)

Argus Multiple Address Space (MAS) supports more than one single-threaded graphics application with each graphics application in its own address space (partition) as shown in **Error! Reference source not found.**.

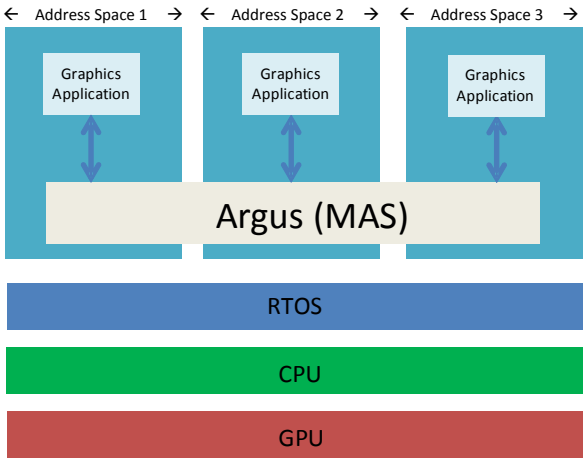

Figure 3 Multiple Address Space Support

CoreAVI's Argus MAS solution is a peer to peer implementation and exists in each application's address space, performing synchronization and coordination on behalf of the graphics applications. If a second partition tries to acquire a shared resource that a first partition has already locked, then the second partition waits until the first partition has finished with the resource and has unlocked it.

Argus allocates personal command and DMA buffers for each OpenGL context ensuring each context can load rendering commands to its own buffer without waiting on other OpenGL contexts.

Argus may exist as a shared object in a multi-partition environment.

3. Support for Embedded Multicore Systems

3.1. Symmetric Multiprocessing (SMP)

In typical Symmetric Multiprocessing a single RTOS manages all processor cores simultaneously and applications can float to any processor core. Utilizing the Argus MAS architecture described above and shown in Figure 4, Argus is able to support any number of partitioned applications in any use of the assigned processor cores.

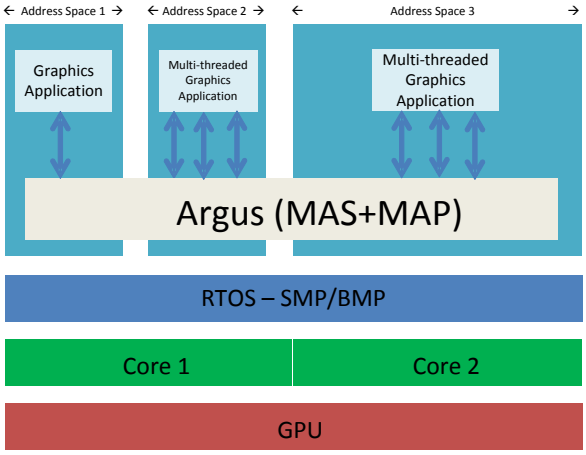

Figure 4 Embedded Multicore System with an SMP RTOS

From an Argus MAS perspective there is no difference between the driver support SMP and BMP. In Bound Multiprocessing (BMP) a single RTOS manages all processor cores simultaneously, with each application locked to a specific core.

3.2. Asymmetric Multiprocessing (AMP)

In Asymmetric Multiprocessing (AMP) a separate RTOS runs on each processor core.

In a multicore system with an AMP RTOS, with or without a hypervisor, CoreAVI offers both indirect and direct rendering OpenGL graphics applications.

3.2.1. Indirect Rendering

CoreAVI employs an ArgusSC MAS / GLX client- server architecture as shown Figure 5. Note the GPU is only visible to RTOS 1.

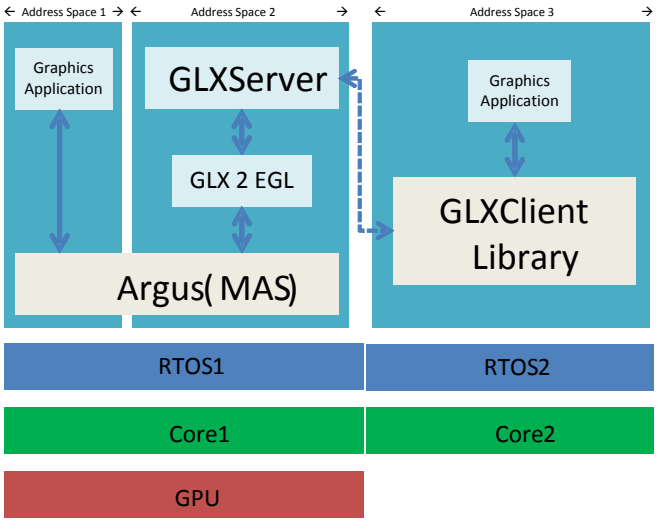

Figure 5 Embedded Multicore System with an AMP RTOS

3.2.2. Indirect Rendering with Hypervisor Support

Argus MAS / GLX client-server architecture in which the GPU is only visible to the RTOS / core with the GLX Server as shown in Figure *6*.

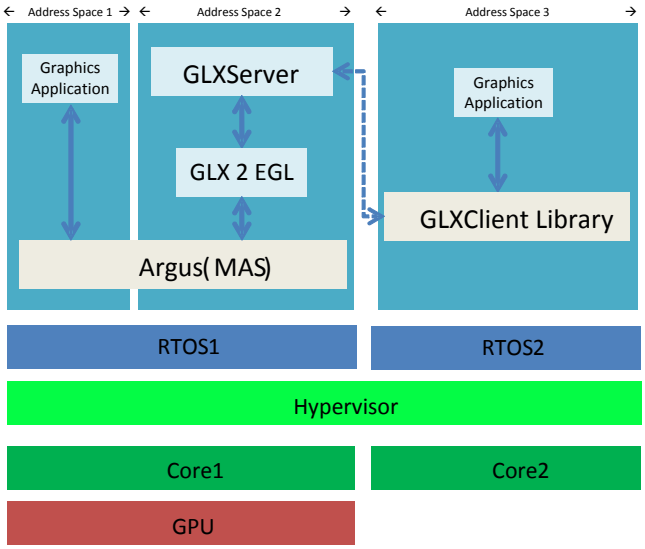

Figure 6 Hypervisor - Argus MAS GLX Client / Server Solution