Minimizing Video System Latency in FACE™ Conformant Systems

INTRODUCTION

Advances in Graphics Processing Unit (GPU) technologies are allowing developers and system integrators to reduce the size, weight, power, and cost of their products. This improvement in efficiency is enabled by GPUs that do more than just render symbols. Today's GPUs also decompress MPEG video, allowing input of raw video. Treating these inputs as automatically updating textures, the GPU can manipulate them as if it is just another graphical resource. Modern GPUs also allow the compression of output video for systems like video recorders, as well as the video simply being sent to a display surface.

The increasingly complex integration of more functions into the GPU has presented new opportunities. One challenge that exists in systems is preserving the low latency requirements of mission critical systems in aerospace platforms. The synchronization of "foreground" symbology with "background" sensor imagery is critical. Large system latencies (>100 milliseconds) between symbology and sensor video for aircraft can compromise mission system accuracies as well as endanger both crew and bystanders.

In 2012, the Future Airborne Capability Environment (FACE) consortium standardized which Application Programming Interfaces (APIs) would be used for graphics processing as part of the FACE Technical Standard Version 2.0. This milestone launched efforts by the aerospace industry to develop software components conforming to the APIs defined by the FACE Technical Standard. CoreAVI is one of those companies, delivering a "paired" OpenGL® and EGL™ graphics driver aligned with the evolving FACE Technical Standards. These products have increased the breadth of applications that use this combined OpenGL/EGL software component and have allowed GPU-centric systems to encompass many of the functional requirements needed by industry; requirements that are not called out by the FACE Technical Standard, nor by the Khronos Group's graphical standards referenced by the FACE Technical Standard. CoreAVI's ability to invest in these products reflects the industry's consensus that the standards referenced by the FACE Technical Standard for graphical processing are sufficiently mature and reusable for system integrators and platform developers.

CoreAVI believes that our solutions meet all goals of balancing high levels of GPU integration and functionality, use of industry-wide standards (OpenGL and EGL), and low system latency. This paper outlines the details associated with technologies that optimize performance while building upon the standardization efforts documented by the FACE Technical Standard that promote modularity, portability, reusability for a convergence of standardized graphical processing APIs.

DISPLAY MANAGEMENT SERVICES

The specification of Display Management Services (DMS) in the FACE Technical Standard v3.0 defines a standard interface to control the merging of external/background video with symbology – oftentimes referred to as “video compositing”. The DMS encourages the use of a compositing architecture using both software compositing and hardware compositing [1]. Newer systems are no longer required to relegate the symbology/sensor-video merge function to a Field Programmable Gate Array (FPGA); they allow the GPU (and its associated driver) to do almost everything. The use of FPGAs perform merging is referred to as “hardware compositing”. To enable software compositing, the EGL_EXT_bind_to_front and EGL_EXT_compositor extensions were published to the Khronos EGL extension registry, the first allowing compositing within the same virtual memory address space and the second allowing compositing across address spaces using a non-blocking API (see Figure 1).

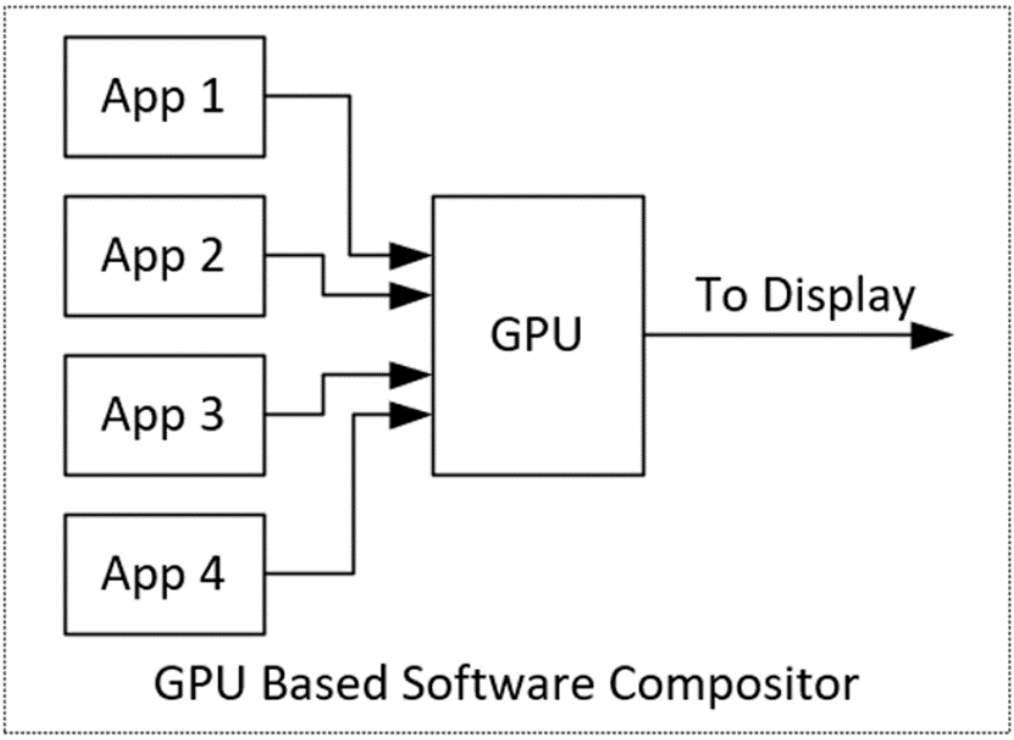

Figure 1: GPU-Based Software Compositor Video Architecture

Software-based compositing can be used to reduce GPU rendering time and display latency by integrating functions that were previously somewhat disparate. GPU accelerated software compositing can reduce the time required for the CPU to generate symbology while reducing the symbology latency. The close coordination of background video processing (through software-based compositing) holds the potential of reducing CPU and GPU processing. The following technology demonstration showed a CPU reduction of 66% and a GPU reduction of 50%. CoreAVI has validated these numbers in technology demonstrations by using a compositing approach that employed a single address space using EGL Pbuffers render to texture capability. No ancillary extensions were required.

This technology demonstration used a 60 Hz input video initially captured by an FPGA. The FPGA transferred the resulting full frame of video to the GPU's video memory (via a PCIe® bus). The GPU merged the background video frame with foreground symbology (overlays) and then displayed the resulting composite image on a display. The data used to drive symbology content (and updates) was received over a network with an update rate of 20 Hz (representing traditional MIL-STD-1553B or Gigabit Ethernet systems). The technology demonstration's primary focus required that the GPU refresh the cockpit display at a rate of 60 Hz.

Another requirement was that no external sensor video frame be "dropped". In a "direct" rendering solution, this approach would result in a system that required the background video and all symbology to be rendered within a 16.6 millisecond time frame. Such an approach unduly burdened both the CPU and the GPU (especially when high levels of symbology renderings are required). On the other hand, a "composited" rendering solution introduced a single thread running at 60 Hz, which takes the background video as a texture and the latest available pre-rendered symbology from the GPU and overlays it at a 60 Hz rate. A second thread running at 20 Hz would render the symbology and make it available to the 60 Hz thread when it finished. This supports a critical system requirement of allowing the symbology content to be updated at a different rate (20 Hz) than the fixed background video rate. The addition of a second GPU rendering "context" was found to have no adverse impacts on the system, and it achieved significant power and throughput savings for both the CPU and GPU. With this approach, the symbology was updated as soon as new data was available from the network. The 60 Hz compositing thread for background video rendered the background video's updates without delaying the output waiting for symbology or dropping any frames.

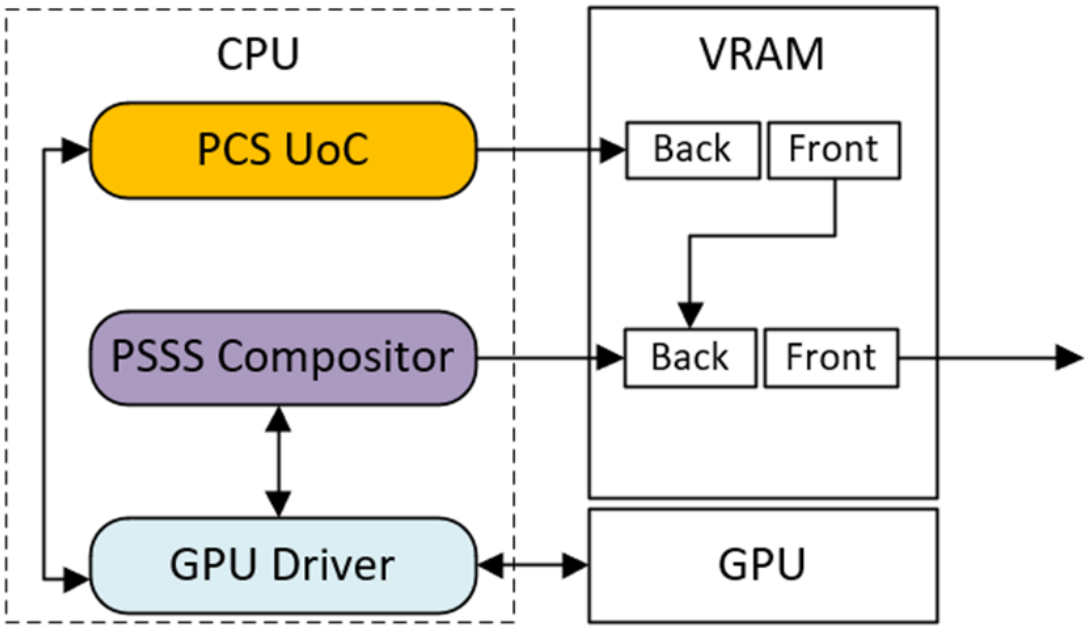

Figure 2: Management of Background Video and Symbology at Different Rates

MULTIPLE THREAD, PARTITIONED ADDRESS SPACE, AND COMPOSITING

CoreAVI has observed industries using video compositing systems with both multi-threads and non-homogeneous address spaces. With the publication of the FACE Technical Standard v3.0 and the EGL_EXT_compositor extension, the need for multi-threaded and multi-address space aware drivers became apparent. EGL_EXT_compositor allows for employing multiple address spaces to do video compositing, resulting in the requirement that the GPU driver be able to support execution from multiple address spaces.

At the same time, the EGL_EXT_bind_to_front extension, which allows double buffering of Pbuffers within a single address space, lead to a requirement to support the execution of multiple graphical rendering threads within a single address space. These requirements are fully embodied within CoreAVI GPU drivers that are both thread and address space “aware”. The extensions provide the capability for off-screen asynchronous updates while ensuring secure graphical information data flows. They also define the compositor application such that it can prevent other applications from rendering to the display without going through the compositor.

The role of the compositor in the system includes the ability to enable management and control of GPU resource allocation for all the applications using the GPU. The compositor extension supports the minimization of application porting efforts by enabling composition of multiple windows within a multi-partition or hypervisor-based graphics system. The off-screen applications can be written without the need to manage the display or have fixed rendering surface sizes as these are set by the compositor, making these applications highly portable.

VIDEO PIPELINE LATENCY

Avionics systems have been merging background video with foreground symbology for decades. To achieve low latency requirements for these deployed systems, these Augmented Reality (AR)–like products have been developed with proprietary interfaces and architectures. These systems have met mission requirements but have failed to produce solutions that are modular, reusable, or portable. The commercial industry has only recently come to the realization that video system latency is a significant issue that must be solved. Commercial solutions are still evolving. In avionics, GPU driver providers (like CoreAVI) have been working to satisfy a wide swath of customer needs by minimizing video pipeline latency.

Some of CoreAVI’s optimizations to address the Video Pipeline Latency problem—made possible by building a reusable GPU driver product—are detailed here.

OVERALL VIDEO SYSTEM LATENCY

For mission critical systems, the overall system latency is often calculated from the time the first pixel of information is sent from the sensor to the time when that same pixel is rendered on a display. The components of this system latency calculation consist of the following elements:

1. Sensor Video Network/Interface Delays
2. Capture/Conversion Logic (outside the GPU) Delays
3. External Capture/Conversion Circuitry Video Decompression (if used)
4. Frame/Line/Pixel Transfer to GPU local RAM
5. Time until GPU reads the image and begins processing
6. Time GPU spends processing the image and combining with symbology
7. Time to transfer complete frame from GPU to output signal conversion element
8. Time to convert from GPU output signal to desired output signal type
9. Time to send image to the display over the desired output signal type
10. Time until the display displays the frame

Items 1 through 4 and items 8 through 10 are beyond the control of software and, for the most part, contribute very little (1 or 2 microseconds) to overall video system latency.

Using a flexible video compositing approach (as discussed previously) provides a method to minimize items 5, 6, and 7 in the video system latency calculation. The GPU driver can work with custom video input transfer circuits to reduce any time associated with the GPU reading of the image. CoreAVI's GPU drivers allow for the timing of external video processing (scale, rotate, translate, zoom, window, etc.) to be done in a just in time fashion, preserving external video freshness. The GPU drivers can be configured to assure no frame drops.

Studies of AR system latencies[2] show that the main delays introduced within the video pipeline are directly correlated to the time it takes to transfer an image between a source and destination. This is because when relying on a standard video interface system the rate at which an entire frame is sent is based on the display update rate. For example, a display with a 60 Hz update rate will take 16.6 milliseconds to transfer a single frame (independent of the resolution), while a display with a 30 Hz update rate will take 33.3 milliseconds to transfer the same frame. This time is constant in a video pipeline when using standard video interfaces. For such an approach, one frame time (16.6ms/33.3ms) is expected for the transfer of an image from a camera source to a video compositing engine (GPU or FPGA), while another frame time (16.6ms/33.3ms) is taken to clock out the composited video to the destination. This means a minimum of two frame times are required to read a frame and send it to the display. These standard video interfaces such as DisplayPort™, HDMI, SMPTE, and NTSC can be grouped together as constant frame time video interfaces, since they take a constant amount of time to transfer one frame of video based on the frame refresh rate.

ENHANCED VIDEO CAPTURE

In some systems, the video latency caused by the video capture can be reduced by using custom logic to do pixel-by-pixel or line-by-line merging of the incoming video with the output from GPUs. This, however, requires that the input video clock and the GPU display clocks be synchronized. In these cases, a single pixel or line comes from the camera and then custom hardware processes or buffers that line of video and sends its out with minimal delay. This approach solves the problem by allowing the FPGA to immediately utilize the output of the GPU. This requires that the video frame be a true "pass-through" video. It also requires custom hardware to develop close synchronization with the incoming video.

However, this high degree of synchronization requires that the GPU allow itself to be synchronized with an external clock source, a feature lacking in most Commercial Off the Shelf (COTS) GPUs.

To get around hardware-specific implementation, CoreAVI offers GPU drivers that support the ability for any external video capture device to provide video frames to the GPU as raw frame data providing a defined interface. This interface can be set up in a non-blocking fashion, to achieve minimal latency for items 4 and 5. Item 4 is minimized by allowing the capture device to write to GPU memory at bus speeds as opposed to over a frame time constant interface. Item 5 is minimized to close to zero by reading the current input video frame pointer immediately before the GPU reads the frame data.

ALTERNATIVE VIDEO OUTPUT

Like most GPU driver providers, CoreAVI has offered software products for years that support the output of video over a GPU's standard display interfaces (DisplayPort, TMDS) for transference of video data to a display. Most recently, an additional software control interface was added to allow the GPU to provide video frames to another device (over PCIe) at system data transfer rates, as opposed to video frame rates. This interface allows the GPU to output video directly to a video transmitter frame buffer, or the video transmitter can access the GPU's memory directly. This process minimizes item 7 of the video pipeline.

VIDEO LATENCY AND SAFETY CRITICAL SYSTEM SOLUTION

Solving the video latency problems in systems must also be done in an environment where more and more defense platforms are complying with Federal Aviation Administration (FAA)/European Union Aviation Safety Agency (EASA) like standards for safety criticality. In 2017, CoreAVI announced an industry first, an OpenGL and EGL GPU driver supporting the EGL_EXT_compositor and EGL_EXT_bind_to_front extensions aligned to the FACE Technical Standard. These drivers provide safety critical graphics capabilities up to the most stringent certifiability levels including FAA/EASA Design Assurance Level (DAL) A. These drivers support most Real Time Operating Systems (RTOSs) used in avionics and have FAA/EASA DAL A safety certification evidence available. GPU drivers including the EGL_EXT_compositor extension not only facilitate complex video merging as well as blended graphics and video windowing but can also be used to support mixed levels of safety certification among various independent partitions drawing to the same display. These drivers are an ideal choice for high reliability and safety critical single-core and multicore display systems.

FACE ALIGNMENT/CONFORMANCE

To get around hardware-specific implementation, CoreAVI offers GPU drivers that support the ability for any external video capture device to provide video frames to the GPU as raw frame data providing a defined interface. This interface can be set up in a non-blocking fashion, to achieve minimal latency for items 4 and 5. Item 4 is minimized by allowing the capture device to write to GPU memory at bus speeds as opposed to over a frame time constant interface. Item 5 is minimized to close to zero by reading the current input video frame pointer immediately before the GPU reads the frame data.

To support the underlying objectives of the FACE consortium, CoreAVI is using business and technical methods made mainstream by the FACE consortium to increase software reuse and safety evidence reuse. Some of those efforts include:

1. Separation of product interdependencies resulting in run-time extension linking
2. Dynamic system configuration through standard interfaces
3. Removal of porting layer code

This results in CoreAVI being able to offer its GPU drivers along with a variety of add-on modules that work seamlessly within the confines of the FACE Technical Standard and the requirements of the Avionics industry. These add-on modules include:

1. Video decompression (H.264, H.265, MPEG, etc.)
2. Video compression
3. A GPU safety monitoring system

All of these add-ons meet the highest DAL of DO-178C. Previously, these add-on modules had to be added to standard packages through a linking process that required every address block to enable the driver, even if the functionality was not being used. A recent change to this family of add-on components, along with dynamic system configuration approach, has allowed CoreAVI to use the same binary objects in all configurations for a specific platform. By using the same binary, the safety certification artifacts are completely reusable. This approach also has the benefit of reducing test cases and test conditions, allowing our software development group to spend more time maturing the GPU driver. Testing concurrent data “race conditions” is simplified along with the testing of multiple run-time configuration mechanisms. Reduced test cases would not be possible without the foundation of a standardized API and standardized and validated extensions as called out by the FACE Technical Standard.

CONCLUSION

When an industry standardizes graphical processing APIs, it allows for providers of those APIs to mature their base software. With a mature software base, developers are freed up to solve related problems. In this case, solutions to the video latency problem were easily solvable by the GPU driver team, and while other solutions may exist, they are not optimal, portable, easily maintained, or reusable. While the solutions presented here are GPU driver-centric, they can be implemented for a wide range of COTS GPU hardware, allowing the application code to be fully portable and reusable between changes in the underlying hardware.

GPUs present great opportunities to reduce size, weight, power, and cost for these systems. However, mismanagement of GPU driver requirements through indiscriminately creating system application specific drivers can result in poor software maturity at both the driver and the application level. By building portable, reusable, and conformant GPU drivers CoreAVI has been able to focus its efforts on evolving the usability of GPUs within a safety critical environment, including expanding the lessons learned in avionics to adjacent markets.

REFERENCES

Links listed below were correct at the time of writing.

[1] Sharing Graphics Resources in a FACE Environment, Proceedings of the 2018 September US Army FACE TIM, August 2018, Daniel Herring, CoreAVI
[2] Teemu Kämäräinen, Matti Siekkinen, Antti Ylä-Jääski, Wenxiao Zhang, and Pan Hui. 2017. Dissecting the End-to-end Latency of Interactive Mobile Video Applications. In Proceedings of the 18th International Workshop on Mobile Computing Systems and Applications (HotMobile '17). Association for Computing Machinery, New York, NY, USA, 61–66. DOI:https://doi.org/10.1145/3032970.3032985

AUTHOR

Daniel Herring
Principal Software Developer

Daniel Herring is a principle software developer at Core Avionics and Industrial Inc. (CoreAVI) developing and integrating OpenGL and EGL graphics drivers for commercial GPUs used throughout the avionics industry. Daniel leads multiple DAL A GPU driver software certification efforts, as well as being responsible for implementing and certifying the advances described above.

Prior to CoreAVI, Daniel worked at Elbit Systems of America developing next generation mission and display computer systems, working on many platforms including the UH-60M, V-22, F-16, AH-64. At Elbit Systems of America, Daniel implemented the Display Management Services using PCIe frame buffer transfers between multiple systems without the use of the EGL_EXT_Compositor extension.

Daniel is a member of the FACE Technical Standard Graphics Subcommittee, co-author of the EGL extension EGL_EXT_Compositor, author of the EGL extension EGL_EXT_bind_to_front and is working on standardizing avionics compositing extensions for use with the upcoming Vulkan® API, to support future Display Management Services implementations in a mixed Vulkan, OpenGL, EGL environment.