Vulkan® SC
Safety-Critical Graphics and Compute Library

Abstract—The Human Machine Interface (HMI) in embedded processing systems continues to become more complex. Aerospace, automotive, rail, and industrial control markets are pushing on all the edges of the technology "box". Displays are getting larger and of higher resolution, information content is becoming more complex and diverse, and images from sensors are now processed with sophisticated levels of Artificial Intelligence (AI) and Machine Learning (ML) algorithms. Further complicating these advances are pressures to make things more "open"[1], more portable, and safety-certifiable. This paper examines the evolving requirements for embedded processors that interact with pilots and operators and suggests a solution that may bring harmony to these seemingly disparate requirements. A software architecture centered around the Vulkan® ecosystem will be described that holds the promise of more efficient processing for graphics, image processing, and autonomous decision making (through AI), while being bundled in an environment that is hardware agnostic and capable of being certified to the most stringent safety levels.

INTRODUCTION

In certain industries, systems that interact with humans must be both sophisticated and highly reliable. Displays and controls for a pilot in a cockpit, a driver in an automobile, an engineer running a train, or an operator managing complex systems in a refinery, all require real time responsiveness, clarity in presentation, and high integrity operations. These diverse industries have a lot in common. They need to continuously develop and improve their systems to allow their human operators to operate at peak efficiency. They are pushed by economic drivers that include short product development cycles and low recurring costs, which produces a need for reuse and/or portability and modularity. Increasingly, these systems are coming under the scrutiny of government regulators with new and evolving requirements for safety certification. Fortunately, technology innovations are providing many solutions to these complex problems. Newer hardware and System-on-Chip (SoC) processors with multi-core architectures and embedded graphics accelerators are realizing significant gains in computing horsepower in smaller size and lower power packages. Real Time Operating Systems (RTOSs) are helping coordinate the use of these heterogeneous resources (CPUs and GPUs) with partitioned and virtualized software foundations. HMI and application software development tools are becoming more user-friendly and more accessible throughout the software environment. Graphics generation libraries continue to provide developers with great capability and flexibility. Even the rapidly evolving world of AI and ML is being incorporated into this ecosystem.

What is missing, however, is a unifying environment that can bring all these separate pieces together in a fashion that promotes efficient operations, abstracts the application software from the target hardware, and brings determinism and testability into the equation in support of safety certification. This paper will explore these challenges as well as discuss a set of standards and products that support graphics, inference engines running AI/ML routines, and general purpose computing libraries. Information will be provided on where and how to get started followed by some concluding remarks.

HMI CHALLENGES, 2021 AND BEYOND

For decades, humans interacted with machines using mechanical indicators, warning lights, and gauges. As cathode ray tube displays were introduced to cockpits and industrial control panels, greater flexibility was provided for systems, in that different types of information could be selectively rendered on the same display surface. The flat panel display technology that we know so well today really broadened the places and applications for diverse information presentation. These displays are now in our cars, in our phones and watches, and just about everywhere we go. We are in the information age largely because of displays used for HMI.

The flexible and expansive capabilities of these display devices tend to spawn new ideas and new applications which, in turn, creates a need for higher resolution displays with faster update rates and more complex rendering challenges. These requirements drive a greater need for processing power, but in environments where space, power, and cost can be at a premium. New systems have to always manage the “do more with less” requirements of technology churn. The content presented on these displays has radically evolved as well. What started out as the rendering of what was formerly rendered on lights and “steam gauges” has evolved into an art form where realism dictates performance capabilities.

One might think that with all of these common factors driving such huge segments in our markets that there would be greater uniformity in the processing platforms that run them, but that has not been the case. The free markets driving silicon processing platforms still support several different Instruction Set Architectures (ISAs)—like x86, Arm, and the up-and-coming RISC-V—which are all thriving. However, getting software applications to easily run on all these constantly evolving processing platforms is a non-trivial exercise. RTOSs have helped insulate software applications from the hardware to a certain degree but there are still challenges. Graphics development libraries and APIs like OpenGL® have also helped, but there are still wide chasms to cross when going from one hardware platform to another.

As sensors are increasingly used to enhance the situational awareness of human pilots and operators, they are being rendered on every display in a cockpit or dashboard or control room console. That said, embedded processing systems are doing more than simply rendering these images; they are doing massive amounts of sensor processing including fusion, stabilization, stitching, target recognition/characterization/tracking ,and facial recognition functions. This processing is also allowing for augmented reality presentations where computer-rendered graphics are overlayed, conformally, on top of the sensor images. This capability presents challenges for the embedded processing environment at both the hardware and software level.

With access to more and more sensors, systems are now empowered with more information and can thus make more decisions on their own. Autonomy is becoming a part of planes, trains, automobiles, and everything in between. The AI and ML algorithms that form the foundation of this autonomy need to run on neural networks and powered by inference engines[2] that process real-time sensor data, to compare with learned (compute trained) data, and then make autonomous control decisions. This capability presents significant challenges to the embedded processing platform developer because it involves so many different facets of the design.

As embedded systems are empowered with more and more capability and control (see Figure 1), they will be challenged to do so in a safe and reliable manner. Because these systems are oftentimes at the heart of critical platforms (airplanes, automobile, transportation systems, or control rooms) they will be expected to undergo rigorous safety certification.

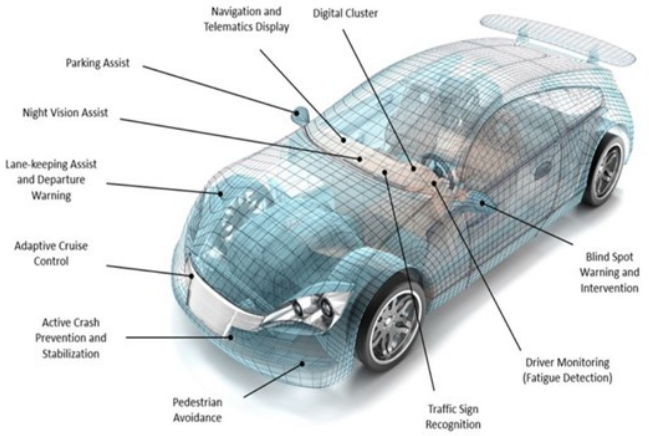

Figure 1: Embedded Systems Grow in Complexity with the Addition of a Multitude of Sensors

VULKAN SOLUTION FOR EMBEDDED

Industry has worked diligently at trying to solve all of these problems with a single ecosystem—something that offers better performance, supports both graphics and general-purpose computing, is agnostic to hardware, and that has a path to safety certification. A standard called Vulkan[3] is the answer to this challenge. Vulkan is managed as a set of open-source specifications that allow industry to both embrace and evolve the standard (and any related products) through an industry group called Khronos. This broad base of support is fundamental to those industries that require safety certification. Allowing industry to participate in standard setting committees and thus influence future specifications is essential. The Vulkan Application Programmers Interface (API) is a whole new world for embedded processing systems that have used OpenGL on their GPUs in the past. Unlike OpenGL, Vulkan allows access to the GPU's general purpose compute functionality, similar to OpenCL and CUDA. Due to the performance increases that were initially achieved, Vulkan started off with a large appeal to video game developers. However, as the benefits of this improved efficiency became more broadly understood, the embedded processing industry took notice. In support of this wider adoption, the Khronos Group is in the process of defining a safety-critical subset of the Vulkan standard. This has captured the attention of aerospace, automotive, and transportation industries where OpenGL SC has been used in the past, and where high levels of safety certification are required.

Vulkan's key components are execution units, work queues, command buffers, pipelines, subgroups, memory buffers, and the Vulkan device or "VkDevice" (to which all commands are applicable). Vulkan also makes use of SPIR-V (Standard Portable Intermediate Representation for Vulkan), which provides the compiler infrastructure for GPU environments. Currently, there are several low-level guides on 'how to program' Vulkan, while high-level information on the Vulkan software architecture continues to evolve.

For graphics applications, Vulkan allows for direct control over the GPU and it supports the incorporation of higher-level libraries built on top of traditional libraries, such as OpenGL (e.g. OpenGL SC 1.0 and 2.0). Bundling options within this environment allow software development companies to add GPU managers, compositing support, and video capture mechanisms. The Khronos group sponsors display management with the compositing extension, which minimizes application efforts for the instantiation of multiple windows within a multi-partitioned graphics system (e.g. "VK_KHR_DISPLAY" extension, "EGL Compositor" extension for OpenGL, etc.).

When properly configured, compositors in a Vulkan environment can:

1. Enhance information assurance by preventing any non-primary contexts and surfaces from rendering to the display.
2. Allow for management and control of GPU allocation to specific contexts, including how much GPU memory a specific application is allowed to use.
3. Support application-dependent off-screen window drawing, ensuring that one application's data will not be overwritten with another application's data.
4. Prevent Vulkan or OpenGL from rendering unless express instructions from the compositing application allow it.
5. Run in hypervisor environments enabling secure and independent virtualized GPU partitions.
6. Allow system designers to mirror off-screen windows to multiple windows on physical displays (without having to invoke multiple applications.

A by-product of implementing higher level APIs on top of Vulkan (such as OpenGL and EGL), is that Vulkan can easily work with popular existing HIM auto-code generating tools. This environment provides a standard windowing API that can be used in mixed criticality systems – ideal for embedded processing applications like aerospace, defense, automotive, rail, and industrial.

One of the most important improvements over previous APIs & Libraries (e.g. OpenGL/OpenCL) is that Vulkan opens up the world of shader and General-Purpose Graphics Processing Unit (GPGPU) Kernel compute (see Figure 2). Shader compute gives an application the ability to repurpose the shader engines for GPGPU computation, which means that a Vulkan application can reduce the workload of the CPU by utilizing the GPU more. Vulkan also provides a much lower level of access to the hardware, which means it can have significantly less overhead than standard OpenGL, which in turn improves performance.

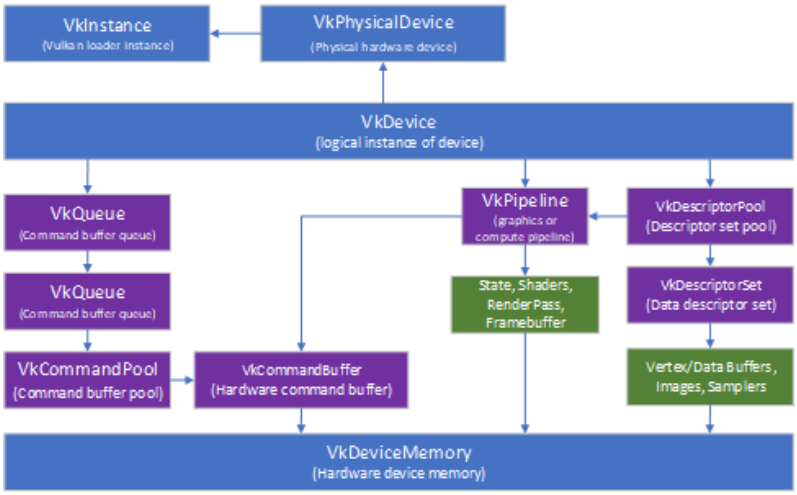

Figure 2: The Vulkan Standard Provides a Rich Set of Components for Graphics, Image Processing, and AI/ML

Vulkan SC is a new specification being created to specifically support embedded platforms that require human machine interaction: aerospace, automotive, industrial, and transportation industries. Standard Vulkan already supports a wide range of GPU devices across most processor architectures. Vulkan SC currently supports a smaller number of platforms but there is still support for AMD GPUs, NXP SOC's (with Vivante's GC7000 SXVS GPU) and most recently, support for the Arm Mali-G78AE (Automotive Enhanced with ISO 26262 capabilities). The Vulkan SC ecosystem is expected to rapidly expand and improve as more and more safety-critical projects adopt it as a standard.

Although Vulkan SC builds upon standard Vulkan (1.1) there are some differences that ensure a path to safety certification. These differences include:

1. SPIR-V shader compiling and linking is offline and can only be performed on a host development system.
2. Pipeline cache and pipeline derivative functionality is more restrictive.
3. Freeing of memory is not allowed in Vulkan SC.
4. Vulkan SC will have additional callback mechanisms to deal with command buffer memory exhaustion and fatal error handling.

BEYOND GRAPHICS - VULKAN FOR IMAGE PROCESSING AND AI/ML

In addition to Vulkan 1.1 and Vulkan SC, the Khronos Group has released a standard for computer vision and sensor processing. OpenVX 1.3 is an industry standard API that provides a feature set for implementing a variety of image processing, computer vision, and artificial intelligence functions. It is a high-level compute API that fits into the Vulkan ethos of building higher level abstractions on top of a low-level driver. And, because it has broad industry support, it is an ideal replacement for systems that previously used OpenCL, CUDA, and/or OpenCV. OpenVX 1.3 supports the deployment and design of neural networks, vector machines, Gaussian filtering, optical flow, and more. Users can build libraries on top of Vulkan 1.1 (non-cert) and Vulkan SC (cert) to implement both graphics and general-purpose compute capabilities, all within the same framework.

The Khronos group has also released a safety OpenVX SC feature set for automotive embedded vision applications using GPU acceleration for vision-based embedded machine learning applications. This "Safety-Critical Deployment Feature Set" provides algorithms for performance that are crucial to pre-processing and post-processing tasks on sensor data flows. This collection of computer vision algorithms—edge detection, Sobel filters, etc.—provide an infrastructure for neural network inferencing engines to not only perform efficiently, but to do so as part of a safety-certifiable ecosystem. This architecture supports a wide range of AI/compute applications such as augmented vision systems, object detection/tracking/identification, signal processing, image processing for degraded visual environments, security monitoring, encryption, Advanced Driver Assist Systems (ADAS), and more. This software architecture can be hosted on virtually any RTOS and supports flexible GPU compute capabilities, but in a deterministic, runtime-state-management environment (no undefined behavior is present). This approach allows conformance to the most stringent levels of RTCA DO-178C, EUROCAE ED-12C, and ISO 26262 (see Figure 3.) The Vulkan ecosystem also allows software developers to create their own mathematical libraries like Basic Linear Algebra Subprograms (BLAS)[5] and FFT for accelerated vector and matrix calculations in a safety-critical manner. BLAS and FFTs form the mathematical foundation for most AI and computer vision platforms.

Another Khronos-supported standard that can work within the Vulkan ecosystem is the Neural Network Exchange Format (NNEF)[7]. NNEF was developed to support the seamless transfer of a wide variety of trained networks into various inference engines. The compute expense of many convolutional neural networks can be prohibitive for the embedded processor developer. Many in the industry are using low-power devices to accelerate neural net-based inferencing, but their approaches vary considerably from one company to the next. This chaos in the industry runs the risk of creating unnecessary barriers and fragmentation to solutions that might otherwise work across multiple platforms. NNEF enables a standard set of neural network training tools and inference engines that reduces machine learning fragmentation. NNEF provides stability to a rapidly changing and evolving AI/ML environment by providing an extensible standard that industry can rely on. NNEF defines a complete set of structure, operations, and parameters of a trained neural network with a degree of independence from the training tools used to produce it. As embedded systems adopt AI/ML routines into their ecosystem, the differences between the cloud-based training tools and the processor implemented inference engine can be challenging. NNEF, when deployed within the OpenVX ecosystem (see Figure 4), brings determinism and discipline into the equation, which will allow embedded system developers to both reduce development time and to create a path to safety certification.

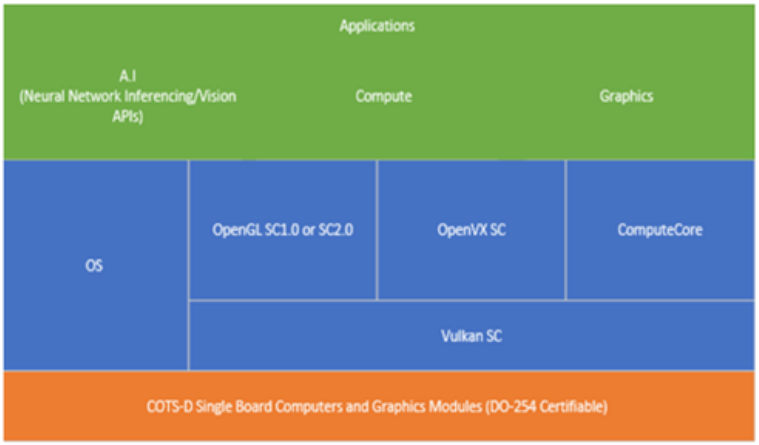

Figure 3: The Vulkan Standard Provides a Rich Set of Components for Graphics, Image Processing, and AI/ML

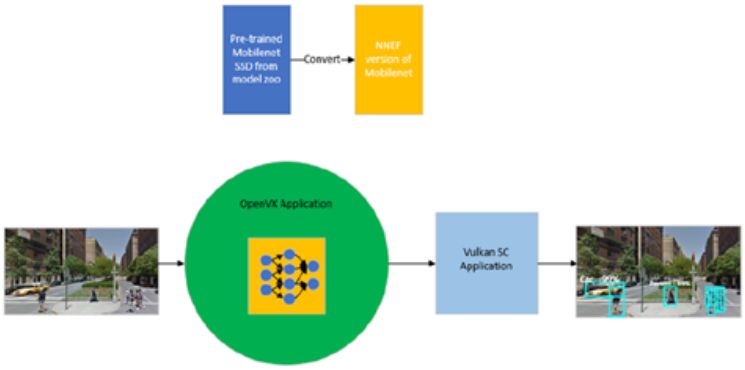

Figure 4: OpenVX Supports Significant Image Processing and AI/ML Capabilities

GETTING STARTED WITH VULKAN

The arguments for employing some or all of the Vulkan ecosystem into an embedded processing system that runs the Human Machine Interface are compelling. For performance, portability, and a path to safety certification there is clearly no better option. In spite of all the variables and options, getting started with Vulkan is straightforward. Most integrators will take the following steps: select hardware, select an RTOS, procure or adapt the Vulkan SC driver to the platform, select graphics driver options, select an HMI tool (if desired), and procure or adapt OpenVX, Compute, and NNEF files (as needed).

One of the great advantages of Vulkan is the broad hardware support it enjoys among hardware providers. Vulkan is supported by all major Instruction Set Architectures (ISAs), including x86, Arm, and RISC-V. Vulkan can run on highly integrated SoC devices as well as on discrete CPUs and GPUs.

By its very nature, Vulkan 1.1 and Vulkan SC run independent of the RTOS. However, as other elements are integrated, RTOS considerations do factor in. Vulkan has been integrated by various RTOS companies throughout industry. Selecting hardware and an RTOS with an established Vulkan integration pedigree will reduce development program risks.

After the target platform (hardware and RTOS) is established, developers should create or procure Vulkan and integrate accordingly. Vulkan's place in the software architecture will allow it to insulate applications from the hardware platform and provide those applications with low-level access to critical silicon resources. Most importantly, it will serve as a host to other add-ons and extensions (OpenGL, OpenVX, BLAS, etc.).

For graphics applications, the embedded processor developer will then choose to use Vulkan for graphics or to integrate an OpenGL product. The graphics environment will need to include a GPU manager, a compositor, and if required, video encoding/decoding drivers.

Finally, for image processing and/or AI/ML routines, the designer should consider the acquisition of a product like OpenVX, compute libraries/routines, and file conversion tools like NNEF (depending upon the application needs).

CONCLUSION

Systems that interact with pilots and operators in safety-critical environments are seeing new challenges with each passing year. These systems continue to become more complex while markets put pressure on designers to lower costs, reduce development times, leverage reuse and scalability where possible, and conform to safety-critical standards. These pressures are in place for all market spaces: aerospace, automotive, rail, and industrial control. Displays and the content rendered on them are becoming more complex and oftentimes including images from sensors that must now be processed with sophisticated levels of Artificial Intelligence (AI) and Machine Learning (ML) algorithms. Portability concerns are always at the forefront, as there continues to be a wide variety of hardware and RTOS platforms needing support. In conjunction with all of this is the need to comply with increasingly stricter industry safety standards. One solution to all of these issues is the Vulkan software architecture and ecosystem. Vulkan promotes more efficient performance by exposing low-level silicon features to software applications. Vulkan promotes both hardware and RTOS agnosticism with a structure that was built from scratch with this in mind. Vulkan supports both graphics and general-purpose processing while also enabling the hosting of libraries with image processing capabilities, AI/ML routines, and low-level mathematics functions, so that advanced applications can operate at peak efficiency within an embedded processing system. And, because Vulkan is an open standard, there are easy and defined steps to experiment with it, adopt it, and field it for a variety of products.

ACKNOWLEDGEMENT

The author would like to thank Lucas Fryzek (Field Application Engineer) and Ken Wenger (Software Engineer) both of Core Avionics & Industrials Inc. for their help in contributing to this paper.

REFERENCES

1. Karl Wiegers & Joy Beatty. “Requirements for Devices Around Us: Embedded Systems, Part 2.” modern analyst.com. https://www.modernanalyst.com/Resources/Articles/tabid/115/ID/3149/Requirements-for-Devices-Around-Us-Embedded-Systems-Part-2.aspx
2. B.G.M. Vandeginste. “Handbook of Chemometrics and Qualimetrics: Part B.” Data Handling in Science and Technology, 1998
3. Khronos Group. Vulkan 1.1 Reference Guide. https://www.khronos.org/files/vulkan11-reference-guide.pdf
4. Khronos Group. VK_KHR_display(3) Manual Page. https://www.khronos.org/registry/vulkan/specs/1.2-extensions/man/html/VK_KHR_display.html
5. GNU Scientific Library. BLAS Support. https://www.gnu.org/software/gsl/doc/html/blas.html
6. Khronos Group. The OpenVX Specification. Version 1.3 10 Sep 2020. https://www.khronos.org/registry/OpenVX/specs/1.3/html/OpenVX_Specification_1_3.html
7. Khronos Group. Neural Network Exchange Format (NNEF) Specification 1.0. https://www.khronos.org/nnef

AUTHOR

Michael Pyne
Director Strategic Accounts & Solutions Architect

Mike’s role as Director Strategic Accounts & Solutions Architect at CoreAVI allows him to bring together the rapidly evolving world of “open” software infrastructures with the safety-critical requirements of both manned, pilot assisted, and autonomous platforms. Mike has over 40 years of experience in Defense and Aerospace markets. Previously, as an engineering fellow with Honeywell, Mike developed cockpit architectures and systems for a variety of airborne defense platforms like the F-15, F-16, F/A-18, C-130, OH-58D, CH-47, and the V-22. Mike’s focus on all of these platforms was in using open system architectures for high reliability/mission critical roles in rugged military environments.

Mike is based in Albuquerque, New Mexico, has a BSEE from Brigham Young University, and holds two patents in the area of sensor processing.