Achieving Safety-Critical Display and Compute for Automotive Applications

INTRODUCTION

As automotive features and functions continuously evolve, so does the need for solutions for display and Advanced Driver Assistance Systems (ADAS). Displays have changed from analog to mixed analog/digital to completely digital, enabling other enhancements such as digital mirrors. ADAS is evolving from cruise control (maintenance of speed) to auto-pilot-like driver assistance, and beyond. These continuous advancements are increasing the need for GPUs to perform video processing and compute (data-level parallelism through many core SIMD engines) as well as advanced graphics.

GPUs

GPUs help consolidate functions performed by the dozens of application-specific standard products and FPGAs in today's vehicles into a common platform that can be adapted based on need through application software that simplifies vehicle design. GPU programmability simplifies vehicle model differentiation and enables continuous improvement through continuous deployment by way of over-the-air updates. Some of the functions that can make use of GPU capabilities include speed limit recognition, lane departure detection, blind spot analysis, parking assistance, pedestrian avoidance, and active crash prevention and stabilization.

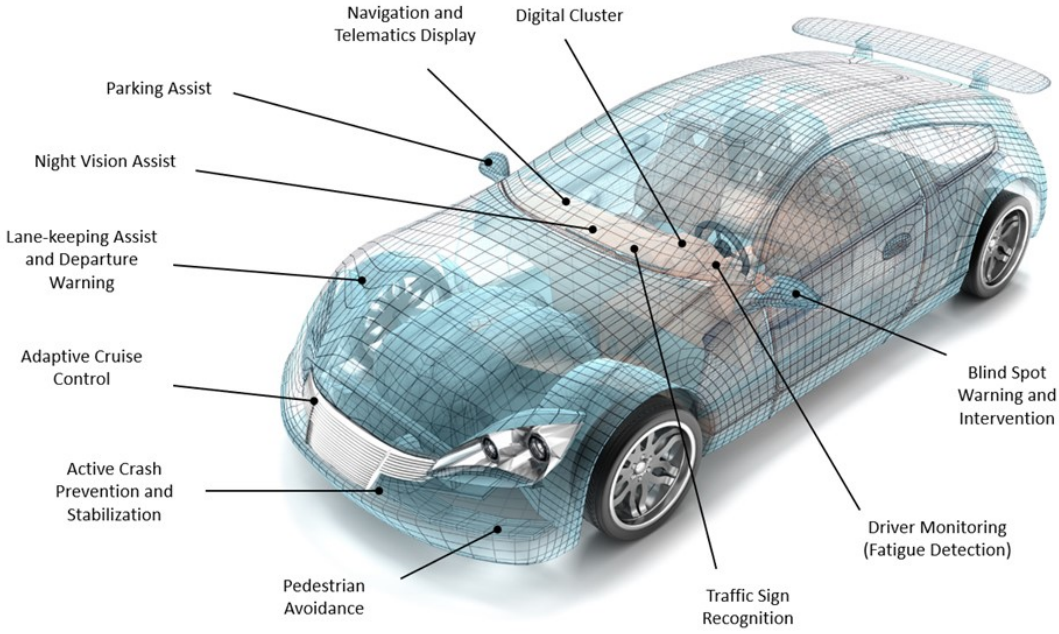

Figure 1: ADAS Functions that Benefit from GPU Capabilities

It is not enough that GPUs can perform these functions; to be deployed in road vehicles these functions need to meet specific requirements described in the ISO 26262 functional safety standard. A key element is the driver and library software between the GPU and the application, as shown in Figure 2.

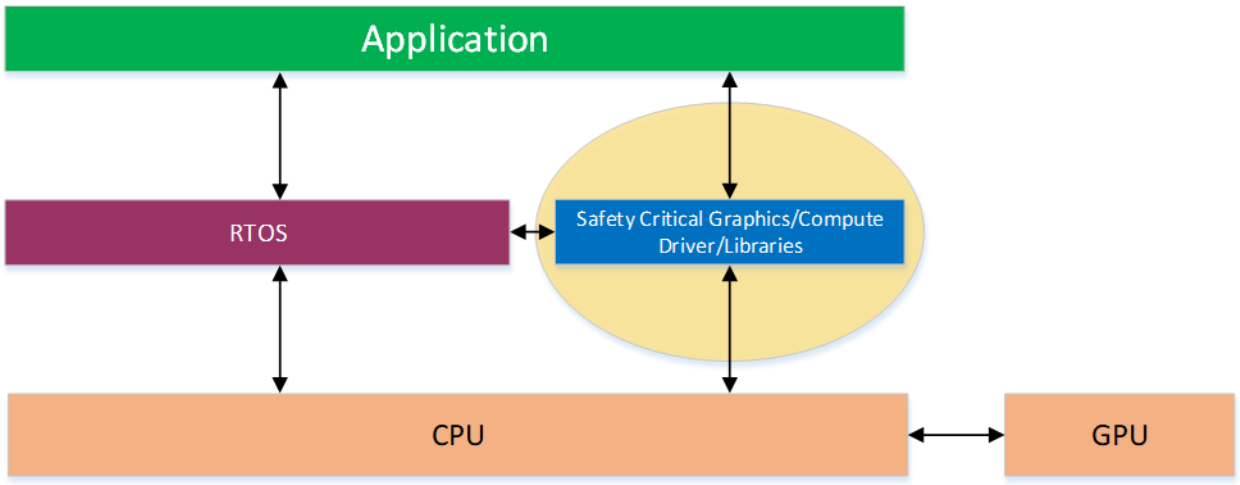

Figure 2: Software Layers Between GPU and Application

Part of the evolution of automotive function implementation was the use of commercial graphics drivers with a software wrapper to enable use with different RTOSs coupled with safety monitors. The early digital displays needed simple safety monitors, as the safety-critical information was in the form of defined symbols at defined locations. These displays could be monitored with efficient readback and check, which permitted the use of commercial drivers and GPUs, because the rendered result was checked (which can be acceptable for ISO 26262 ASIL B). However, the situation becomes more challenging for ASIL D applications like those associated with ADAS, as the software and GPU tool chains need to support more stringent ISO 26262 objectives.

THE EVOLUTION OF COMMERCIAL GPU SOFTWARE

Given the increasing safety functions allocated to GPUs, this paper will examine the evolution of commercial GPU software solutions that enable this progress. The focus is on third-party software solutions within an ISO 26262 context to achieve the functional and safety requirements, referred to as Functional Safety (FuSa). Functional safety is the absence of unreasonable risk due to hazards caused by malfunctioning behavior of electric and electronic systems. What follows is a different commercial solution approach.

While software for GPUs can be developed new in-house, this would require expertise in how GPUs operate to render graphics and how to apply an API to provide the functionality to an application. Given the API is typically an industry standard, the decision to invest in developing something that could be found in the market, and is generally the same as what a competitor would use, may not make the best business case unless there is differentiation. Finding a commercial solution is cost-effective, not to mention that application developers can focus on the differentiation and advancement rather than developing all the building blocks (i.e. typically they would not develop an in-house RTOS, so

why develop a complex driver such as a graphics or compute driver?).

THREE APPROACHES

There are three available options when developing an ISO 26262 FuSa application. The first approach is the qualification of safety-related elements that have not been developed according to the ISO 26262 standard and are usually limited to low and medium complexity elements (e.g. math library). Qualification of such elements requires a specification against which requirements-based testing is completed to verify functional behavior. For most GPU software, an API specification available from the Khronos Group that covers at least 80% of the typical software. (There are vendor-specific functions for hardware initialization, etc.) Passing a Khronos conformance suite would provide evidence of compliance with the API specification. The remaining gap is that the API specifications are not complete in terms of defining failures or anomalous behaviors, which also need to be addressed. Then there is still no guarantee that the solution is safe because key principles for safe software design, like those listed in Table 1, may be lacking. This could affect the behavior of other functions to the point of loss of function and reliance on safety monitoring rather than the safety monitor being an additional layer of defense (“belt and braces”).

a	One entry point and one exit point in subprograms and functions
b	No dynamic objects or variables, or else online test during their creation
c	Initialization of variables
d	No multiple uses of variable names
e	Avoid global variables or else justify their usage
f	Limited use of pointers
g	No implicit type conversions
h	No hidden data flow or control flow
i	No unconditional jumps
j	No recursions

Table 1: Software Safety-Critical Design Principles (ISO 26262-6)

The second approach is proven in use, which applies to reusing or modifying a fielded element that was developed before the release of the ISO 262662. As new functions require modern GPUs with supporting software—available before the existence of ISO 26262—this approach is outside the context of this paper.

The third approach is Safety Element out of Context (SEooC) which is developed according to ISO 26262 as generic products for vehicle applications outside of the context of a specific system; it applies to elements of any complexity. These elements are accompanied by a Safety Manual that provides information on the safety concerns addressed, as well as those requiring consideration at the application and system level when putting the SEooC into context.

Standard product SEooC is commonly provided with an accredited Safety Assessment Certificate. A standard product SEooC would also have a standard Development Interface Agreement (DIA) from the supplier. Suppliers are usually willing to configure the SEooC for different targets and even customize an SEooC based on a customer provided DIA, an adaptation of a Statement of Work (SOW) as defined by ISO 26262.
Not only would the software be designed and verified against the principles listed in Table 1 (and more), but the basis for the design is also an API specifically designed for safety-critical application use. When it comes to GPU support, there are several industry-standard APIs defined for safety-critical applications such as Khronos' OpenGL™ SC 2.0, Vulkan® SC, and OpenVX™ SC. These industry open-standard-based safety-critical APIs are designed to:

- Reduce driver size and complexity
- Exhibit deterministic behavior
- Remove online compile to offline (enable certification of the GPU software)
- Manage static memory
- Eliminate undefined behaviors
- Enable robust error handling

These APIs are especially powerful when combined with application access to graphics (Vulkan SC and OpenGL SC 2.0 simultaneously), compute (Vulkan SC), video processing libraries, compute libraries, video decode and video encode simultaneously, fully integrated and designed to work together.

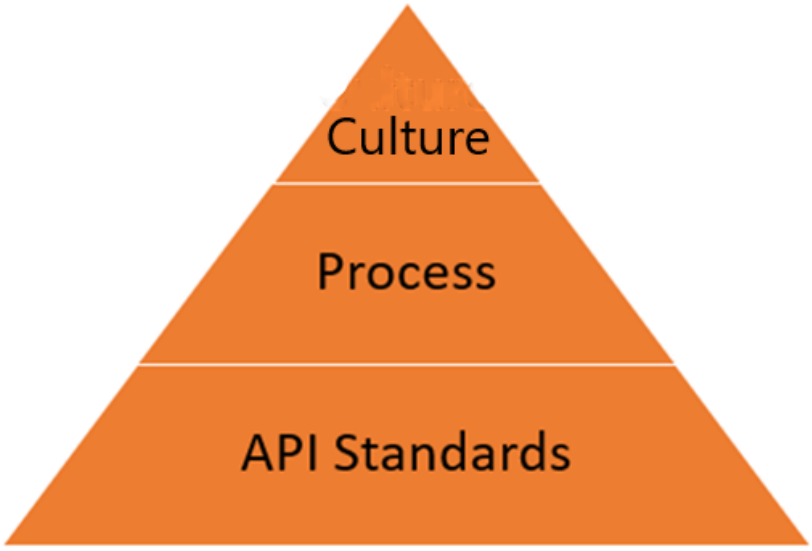

Figure 3: Safety-Critical Software Development Pyramid

API standards considerations include:

- Are the organization's solutions using industry open standards?
- Does the organization participate in and initiate industry open standards working groups?
- Does the organization provide complete solutions?

Process considerations include:

- Does the organization hold an internationally accepted quality certificate?
- Does the organization have and follow an ISO 26262 development process supporting up to ASIL D?
- Does the organization work with a safety assessor and certification body, such as TUV?

Culture considerations include:

- Is the organization's primary focus on safety?
- Does the organization work with partners to educate the industry?
- Does the organization innovate?

At CoreAVI, we do not develop applications, we make your applications safer with our innovative drivers and libraries for high performance computing, imaging, and AI, so you can focus on new applications to advance state-of-the-art vehicles.

Further related topics are presented in the following white papers:

- *Safety Element out of Context (SEooC) Solution* – This white paper describes what an SEooC is, how to select an SEooC, and how to put the SEooC into context.
- *Next Generation Graphics GPU Shader and Compute Libraries* – This white paper discusses Vulkan graphics and compute libraries.
- *Vulkan: The Future of Embedded Graphics* - This white paper introduces Vulkan, with discussion on its benefits and how it differs from OpenGL.
- *Certification of Compiler Generated Object Code* – This white paper discusses the safety certification of object code generated by a compiler tool chain for both CPU and GPU targets.

For more information on CoreAVI's solutions, please visit our website or contact sales at Sales@CoreAVI.
:

AUTHOR

Gregory Sikkens
Director, Safety Solutions Architect

Gregory Sikkens has over 30 years of experience holding a wide range of technical and management positions within the rugged embedded COTS market. Prior to joining CoreAVI, Gregory was the senior product manager at Curtiss-Wright responsible for directing and launching a number of advanced graphics and FAA DO-254 certifiable COTS modules. Furthermore, Gregory has extensive hands-on experience in developing and managing OpenGL SC graphics driver development including DO-178 DAL A software development.