Is Your Digital Cockpit Telling You the Truth?

WRITTEN BY

Mustafa Ali
Sr. Product Manager
arm

Christopher Giordano
Vice President UX/UI Technology
DiSTI

Neil Stroud
VP Marketing and Business Development
CoreAVI
INNOVATING A SAFER TOMORROW

Is Your Digital Cockpit Telling You the Truth?

Modern automotive applications are becoming more complex with the disruption of vehicle architecture, which requires heightened safety and managing of mixed-criticality applications. This development facilitates the merging of IVI and cluster, HUD, and mirror replacement displays. This paper discusses moving the automotive digital cockpit to a safer and more reliable architecture while leveraging industry open standards and best-in-class partner collaborations, tools, and hardware to deliver a seamless workflow to developers. Using the latest technology innovations in hardware and software stacks, developers can rapidly create and iterate on a mixture of safe and standard embedded graphics and compute on new safety-critical GPUs. This approach enables faster time to market, reduces development costs and certification effort, and reduces risk, resulting in a safer user experience on and around the road.

1.0 Importance of Safety-Critical Rendering in the Modern Digital Cockpit

Modern automotive applications are becoming more complex with disruption of vehicle architecture, which requires heightened safety and managing of mixed-criticality applications. For example, the merging of IVI and cluster, HUD, and mirror replacement displays may require different levels of safety. One application may need to be engineered to an ISO 26262 ASIL B level, while another, more critical application may need to meet more stringent ASIL D levels. When both applications are running on the same system, they may not interfere with one another or cause any false information to be displayed that may ultimately affect the driver’s safety. This approach could also extend to use cases where applications that are considered safe sit alongside applications that are ‘non-safe,’ such as infotainment information. This scenario is where the importance of safety criticality comes into play.

Using the latest technology innovations in hardware and software stacks, developers can rapidly create and iterate on a mixture of safe and standard embedded graphics and compute on new safety-critical GPUs. This approach heightens the end user’s safety and enables faster time to market, reduces development costs and certification effort, and reduces risk, resulting in a safer user experience on and around the road.

This paper discusses moving the automotive digital cockpit to a safer and more reliable architecture while leveraging industry open standards and proven partner collaborations, tools, and hardware to deliver a seamless workflow to developers. We show that this can be achieved by using a combination of GPU hardware, driver stack, and app development tooling that are fit-for-purpose to handle functional safety requirements and support for mixed-critical workloads. Furthermore, we explain that a pre-certified GPU, Driver, and Graphics Library in a Mixed Criticality Stack is a newer method to help with development and cost efficiencies.

For the purposes of this paper, a generic system topology has been defined **(Figure 1)** that outlines how the various elements must come together to achieve a functionally safe system. This approach can be taken as a baseline and tailored to a multitude of application use cases requiring different safety levels ranging from Quality Managed (QM) to ASIL D. For simplicity, some elements are not shown but assumed to exist within the system such as operating systems and a hypervisor where necessary.

This paper is organized as follows. Section 2 discusses how the safety-critical human-machine interfaces (HMIs) in vehicles have evolved to being digital. Section 3 introduces the considerations in moving from legacy to all digital safety-critical HMIs. Section 4 presents the details of hardware and software components that make up the proposed solution for flexible, fully digital SC HMIs. Section 5 talks about the benefits that partnerships for SC HMI solution development bring to the automotive supply chain and Section 5 presents the conclusions.

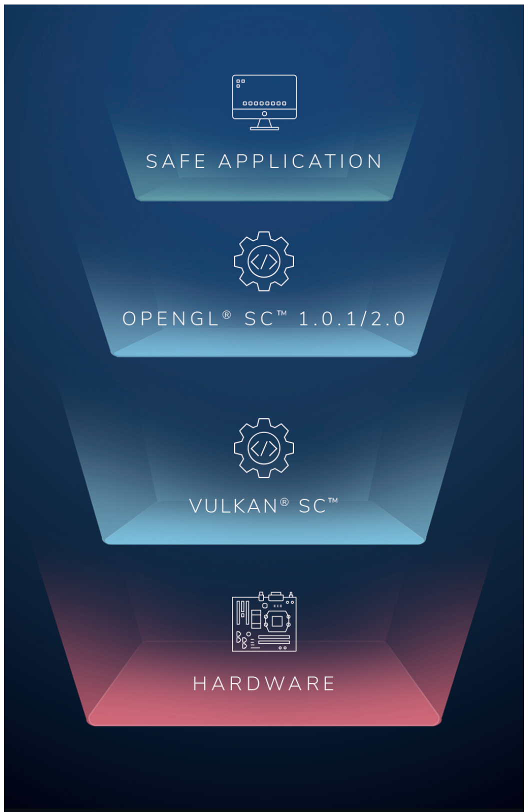


Figure 1 Generic System Topography

2.0 Evolution of Safety-Critical Human Machine Interfaces

Human Machine Interface (HMI) can be any interface, not just digital. The National Institute of Standards and Technology (NIST) defines the term HMI as "*The hardware or software through which an operator interacts with a controller. An HMI can range from a physical control panel with buttons and indicator lights to an industrial PC with a color graphics display running dedicated HMI software.*" [1] Concerning digital interfaces in automotive, a great example of the progression of HMI with respect to functional safety, might be the telltales. Previously, telltales were an LED that would turn on or off, so measuring voltage to the LED and showing it lit was easy from a verification and validation standpoint. Today, the HMI for the same telltale systems can be digital and hence anywhere on the screen **(Figure 2)**.

The automotive market continues to search for the best methods to validate these objects in digital space. However, one needs only to look at parallel industries like aerospace that have been handling digital functional safety graphic user interfaces for decades to gain insight and ideas.

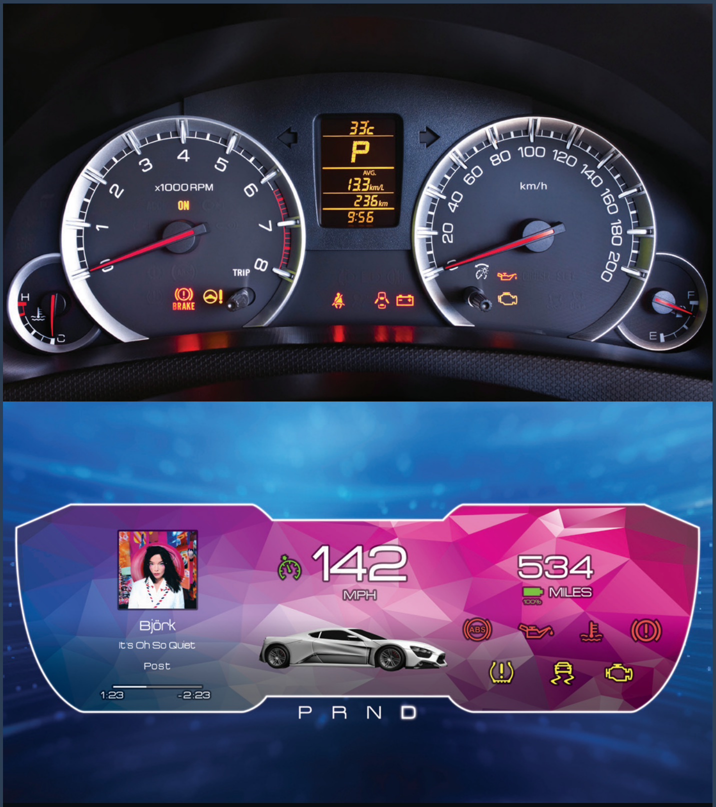


Figure 2 Examples of Legacy and Modern Telltales

1. https://csrc.nist.gov/glossary/term/human_machine_interface

While the automotive market appears to be rapidly progressing in handling functional safety content in digital space, most vehicles on the road today that employ any digital user interfaces rely on a Cyclic Redundancy Check (CRC). The CRC is an error-detecting code used to confirm that the screen contents match what they are supposed to be displaying for the given vehicle data to detect any erroneous changes to the raw data expected to be displayed. This approach has challenges regarding the tolerance of more complex, non-stationary graphical content, for example, flourishes, 3D animations, and other dynamic changes to content, which are discussed further in this paper. To manage the next layer of complexity for FuSa content, which involves adding animations, flourishing and dynamic changes, etc., essentially completely opening up the designer's tool bag for SC content, a fully ISO 26262 ASIL B certifiable hardware and software stack is required to draw graphics and trust the output implicitly, with no need to restrict graphical features to perform a CRC check. Furthermore, to manage the complexity of safety certification, the HMI development process can be automated with the proper commercially available tooling.

There is a transition period where manual drive is moving toward autonomous driving. In this case, where there is still human intervention, there is a need for FuSa in the visual graphics to safely and accurately convey the state of safety-critical data. Once (and if) every vehicle transitions to pure autonomous, this may change.

3.0 Transition from Legacy SC HMI System Design

Like many other sectors, automotive is undergoing a fundamental shift in system architecture driven by integration complexity, cost, form factor, and safety. Two challenges that must be overcome related to the consolidation of functions onto fewer sub-system platforms within the vehicle and, closely coupled to that, the ability to achieve mixed-criticality capability within said platforms.

The first trend can already be seen as the industry migrates from the traditional Electronic Control Unit (ECU) approach to a Domain Controller or Zonal Controller architecture. This method has broad-reaching implications at the vehicle level; it results in a dramatic reduction in the number of processing nodes but also significantly increases the processing capabilities of each node, with each being responsible for multiple functions. In contrast, an ECU would only be responsible for a singular function. These functions that reside on a single hardware platform will likely have different safety requirements, referred to as 'mixed-criticality.'

To put this into context, a modern vehicle instrument cluster displays a plethora of information for the driver. Some of this information is deemed non-safety critical such as information from the media system or navigation information. However, other data such as system 'tell tales' or critical status information would be considered safety-critical. Naturally, the system must ensure that the safety-relevant information is displayed correctly to enable the vehicle operator to react accordingly—the need for mixed-criticality places stringent requirements upon the underlying hardware and software elements.

3.1 Shift from legacy GPU to safe GPU

With the move to richer user interfaces displaying critical information, e.g., speed, lane, gear, etc., and warnings in real-time in modern vehicles, GPU accelerated displays are becoming standard. When displays are used for rendering critical data, these systems are expected to meet safety goals to ensure the vehicle's safe operation.

Until now, safety-critical display elements have been designed to be separate from the GPU rendered part of the display. This approach is because traditional GPUs have not been designed with functional safety goals, making it complex to ensure the correctness of the rendered output. This limits what designers can achieve with the flexibility and computation power offered by GPUs.

To fully use the capability of the GPU for safety applications, the GPU itself and the software stack should be capable of tolerating, detecting, and/or reporting faults that could lead to system failure. Having the capability to detect and report faults that may affect the system's safety goals allows a system designer to design complex user interfaces offering rich visual information, knowing that misleading information will not be presented to the user, which could otherwise endanger a vehicle user relying on the information to make critical driving decisions.

3.2 Shift from Embedded to Safety-Critical APIs

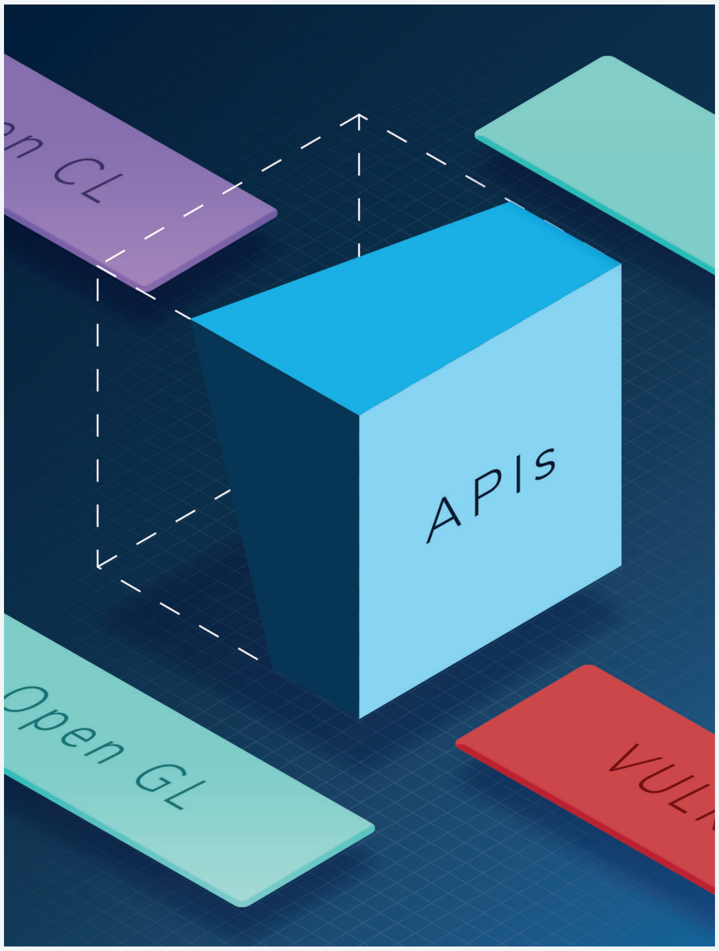
In any visualization-based application, the software has an important role in displaying rich graphical information by maximizing the use of available graphics processing unit (GPU) performance and facilitating the safe display of critical information. To achieve this, all the stack 'layers' must interact correctly to achieve the desired safety functionality or 'goal.' After all, safety is typically about 'divide and conquer.' Naturally, we debate closed and proprietary solutions versus open-standards-based solutions (as opposed to open-source, which is still very nascent from a functional safety perspective). In reality, both must coexist, but the question then becomes about where the hand-off should occur in the stack. Both are important and have associated pros and cons.

The driver and libraries layer of the stack has enjoyed a great deal of industry standardization addressing both commercial and embedded applications. This method includes standards-based APIs such as OpenGL®, Vulkan®, OpenCL™, and many more that enable the application developer with a standard API to interface to the underlying platform. This approach, in turn, facilitates differentiation at the application level along with an element of hardware abstraction and software reuse and scalability. This factor is increasing in importance given the levels of investment in software development and maintenance.

Given the increase in functionally safe applications, the industry has been working to define safety critical derivatives of these same drivers and libraries. The development of the Vulkan® SC™ standard by The Khronos Group is a strong proof point of this and their safety-critical versions of the OpenGL® standard – OpenGL® SC™ 1.0.1 and OpenGL® SC™ 2.0. But what does that mean in reality? What must be taken into consideration to turn a commercial software driver or library into one that is fit-for-purpose in a safe system?

There is a two-part answer to that question. Firstly, safety development is governed by stringent processes and procedures, including requirements capture and traceability, reviews, and development practices. This approach relates to how the product is developed. Secondly, the specific features of the software driver come under the microscope. To give an example, safety systems rely heavily on deterministic behavior. One must know precisely how long a function will take to execute so it can be scheduled accordingly. Some functions that exist within commercial drivers are non-deterministic in nature because, for many applications, some variance in execution time can be tolerated. Obviously, this is not acceptable for a safety-based system, and therefore these functions have to be deprecated from the safety-critical version. This can be seen when comparing the functions available to the developer between OpenGL® ES™ and OpenGL® SC™ variants.

The outcome of this means that the system developer can rely heavily on the benefits of using open-standards-based drivers and libraries even when functional safety is design constrained within mixed-criticality graphics systems. The Vulkan SC and associated OpenGL SC example depicted in **Figure One** delivers on the benefits discussed earlier. This approach can be used for functionally safe graphics applications. However, by utilizing the Vulkan SC driver, the developer can also address safe compute applications prevalent in many ADAS and autonomous use cases.



3.3 Design Automation with Mixed-Criticality

While certification is a requirement to save lives, it can be a costly endeavor. If an efficient development process can minimize the code required to be certified by having the OpenGL draw code well abstracted from the object data code and the user-created code, vehicle OEMs and parts suppliers can save on the certification process. Using pre-certified tools can further reduce the overall cost of certification. Avoidance of monolithic applications during the certification process can ensure a modular and well-structured design that is efficient for the purposes of the graphics stack.

Pre-certified GPU, Driver, and Graphics Library in a Mixed Criticality Stack is a newer method to help with development and cost efficiencies. While the concept of mixing different levels of Automotive Safety Integrity Level (ASIL) with QM content is not new, the automating of that process is unique. **Figure 3** shows both an ASIL layer and a QM layer as an example.

Automation of the creation of separate apps form a single design file, a process known as Mixed-Criticality, lets developers iterate very quickly during the HMI prototyping process and implementation development. This approach helps to see the resulting mixture of QM and ASIL content in the same design and on the target platform in an effort to offset and minimize the cost of certification and the time to develop the graphical layers.

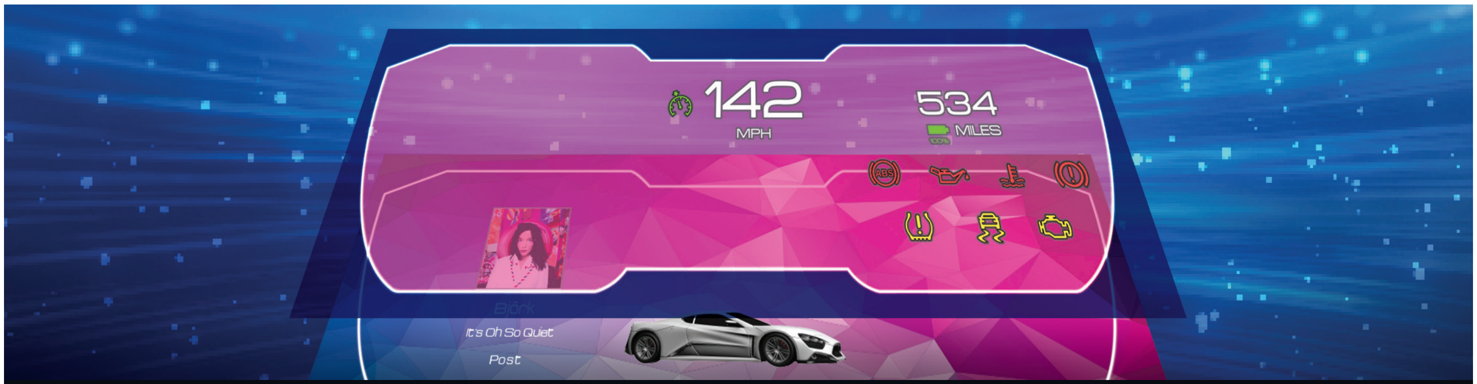


Figure 3 MIxed Criticality Example

4.0 Hardware and Software Solution for Modern Automotive HMIs

4.1 Automotive Targeted GPU Suitable for ASIL B

For a GPU to be suitable for use in a modern digital cockpit system, it must service two fundamental requirements: (1) provide the capability to consolidate the different application workloads that use the GPU hardware while ensuring that there is sufficient isolation between these to avoid any interference, (2) provide the desired safety coverage when running safety-critical workloads so that the hardware can be trusted to detect and report faults.

To meet these requirements, we have developed a scalable GPU as an ISO 26262 safety element out of context (SEooC) that supports the consolidation of multiple workloads and is suitable for ASIL B for detection of random hardware faults and ASIL D for protection against systematic faults. The GPU employs a novel architecture that provides a high degree of flexibility via different ways of mapping multiple workloads to the GPU resources, such as traditional hardware virtualization along with flexible hardware partitioning. Furthermore, it adopts a cost-efficient, industry standard, and application transparent approach for achieving functional safety to minimize the total cost of system development for both hardware and software. Next, we describe further details of these capabilities and how they contribute to meeting the requirements of modern digital cockpits.

4.1.1 GPU Virtualization for Workload Consolidation

Virtualization of GPUs allows different virtual machines (OS) running their own independent graphics stacks to concurrently use available GPU resources in a time-shared manner, as shown in **Figure 4**. GPU virtualization is a key enabler of function consolidation in vehicles. It is key to the design of modern vehicle architecture aiming to reduce the cost of multiple ECUs and wiring harnesses needed.

GPU virtualization needs to ensure adequate separation between workloads executing on the GPU. Time-sharing of GPUs is one of the approaches to enable consolidation. A common way to time share a GPU between multiple workloads is at a frame boundary.

This approach can work well for well-behaved workloads, i.e., release the GPU in a timely manner for other workloads to start in time without missing their rendering deadlines. However, this may not be true when sharing GPU between compute workloads or a mix of graphics and compute workloads.

An alternative way is through flexible partitioning, which provides a higher guarantee of performance isolation. This separation and isolation are essential when executing tasks of differing safety levels on the GPU, referred to as 'mixed-criticality,' such as instrument cluster (safe) and navigation information (non-safe).

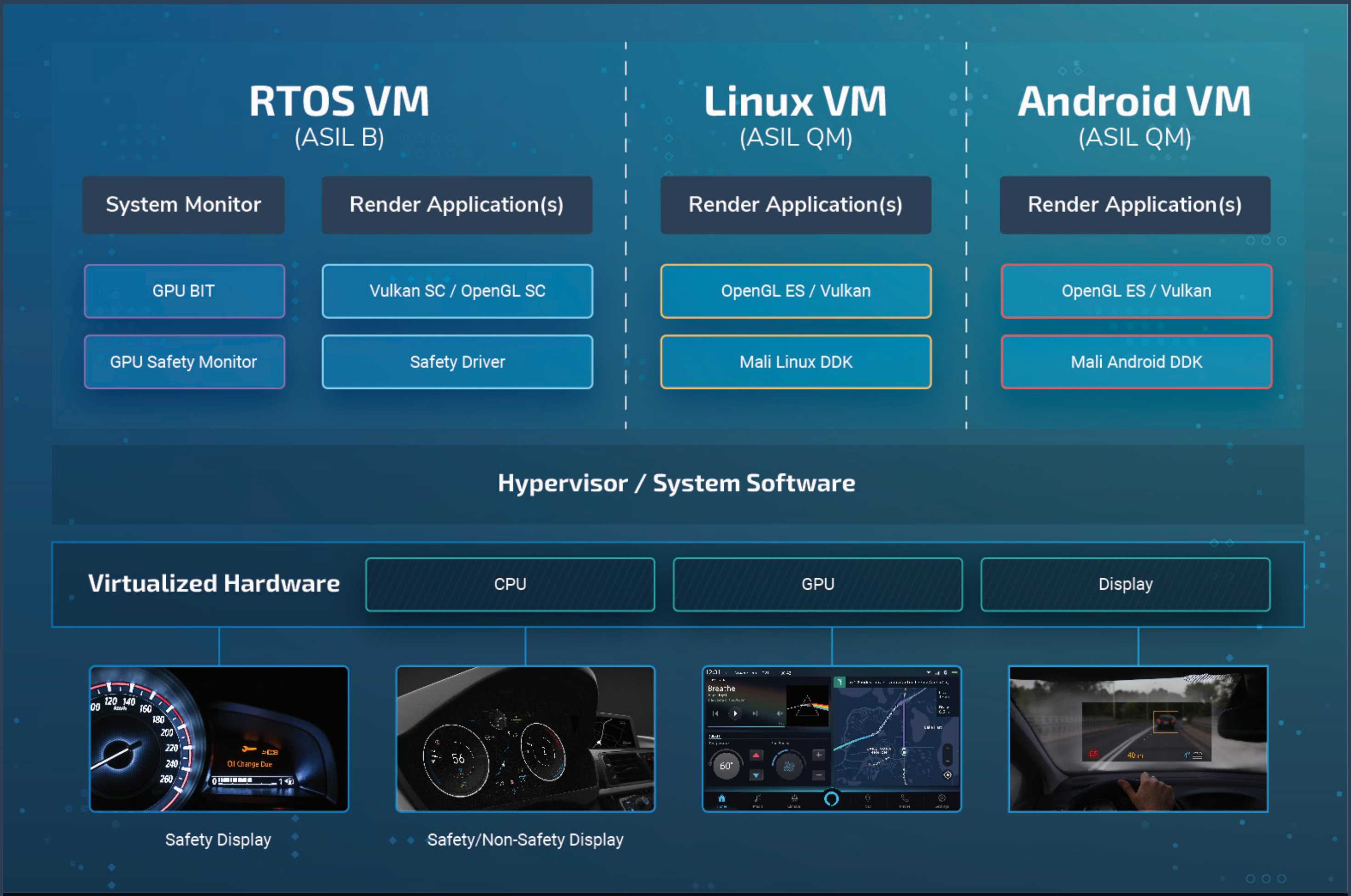


Figure 4 A digital cockpit supporting multiple operating systems as virtual machines running on virtualized GPU

4.1.2 Flexible Partitioning for Workload Separation

Flexible partitioning opens more freedom to system designers to consolidate independent workloads on a given GPU hardware to ensure the highest degree of isolation can be guaranteed. Flexible partitioning allows the GPU to be reconfigured at run-time to support various partitioning configurations depending on various workload demands that need to execute on the GPU. Due to more substantial isolation between different partitions, it is preferable to have a separate partition for the safety-critical workload to ensure deterministic performance, as shown in **Figure 5**.

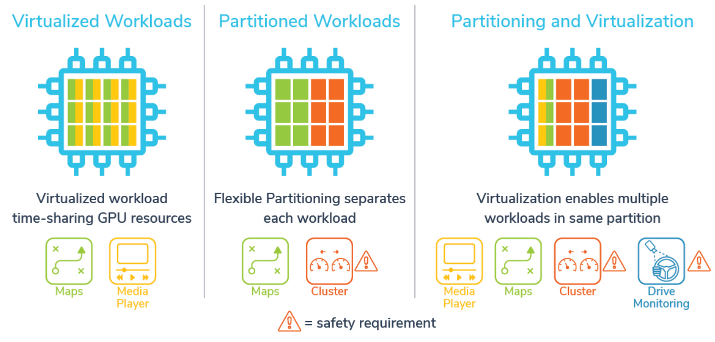
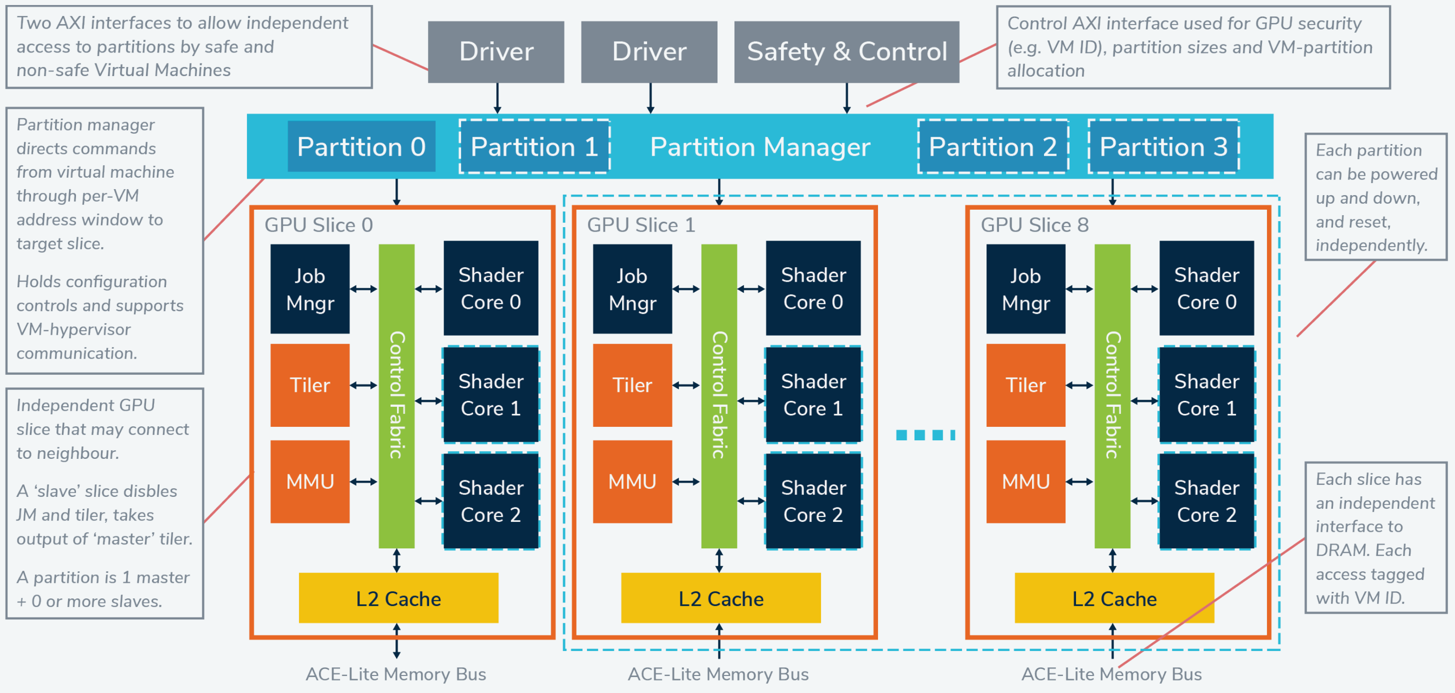

Virtualized Workloads
Partitioned Workloads
Partitioning and Virtualization
Virtualized workload time-sharing GPU resources
Flexible Partitioning separates each workload
Virtualization enables multiple workloads in same partition
Maps
Media Player
Maps
Cluster
Media Player
Maps
Cluster
Drive Monitoring
= safety requirement

Figure 5 GPU virtualization vs partitioning and hybrid approaches to workload consolidation



Two AXI interfaces to allow independent access to partitions by safe and non-safe Virtual Machines
Driver
Driver
Safety & Control
Control AXI interface used for GPU security (e.g. VM ID), partition sizes and VM-partition allocation
Partition manager directs commands from virtual machine through per-VM address window to target slice.
Holds configuration controls and supports VM-hypervisor communication.
Partition 0
Partition 1
Partition Manager
Partition 2
Partition 3
Each partition can be powered up and down, and reset, independently.
Independent GPU slice that may connect to neighbour.
A 'slave' slice disbles JM and tiler, takes output of 'master' tiler.
A partition is 1 master + 0 or more slaves.
GPU Slice 0
GPU Slice 1
GPU Slice 8
Job Mngr
Tiler
MMU
Control Fabric
Shader Core 0
Shader Core 1
Shader Core 2
L2 Cache
Each slice has an independent interface to DRAM. Each access tagged with VM ID.
ACE-Lite Memory Bus
ACE-Lite Memory Bus
ACE-Lite Memory Bus

Figure 6 GPU architecture for supporting flexible hardware partitioning

This flexible partitioning is enabled by a novel architecture that divides the GPU compute resources into slices, as shown in **Figure 6**.

Each GPU slice can operate as a fully functional GPU having its own job manager, tiler (geometry), shading, MMU, and L2 Cache resources, and a separate interface to system memory. The GPU uses a hardware Partition Manager, which provides the interface between the virtual machines and GPU slices. It is responsible for the configuration of GPU slices into partitions.

A partition can have one or more slices connecting to neighboring slices to scale up the number of shader cores. Each partition can also be powered down and reset independently, providing true isolation and freedom from interference to meet ISO 26262 requirements.

Going further, a GPU provides additional flexibility by featuring multiple independent interfaces for virtual machines to submit GPU workload. These various interfaces allow safety-critical VMs to use a dedicated interface while the second interface may be shared. Also, since a safety island may manage the safety and configuration, a separate interface helps to ensure freedom from interference.

4.1.3 Enabling ISO 26262 ASIL B Functional Safety

GPUs support hardware safety mechanisms, as shown in **Figure 7**, that facilitate meeting the requirements of ASIL B systems.

Since GPUs constitute a significant investment in silicon area and power consumption, these functional safety mechanisms have been architected to be largely configurable while ensuring an optimum balance between silicon area and performance. This is ensured by using industry-standard built-in self-test (BIST) approaches that can be tailored to meet the safety requirement vs. area and performance overheads.

As GPUs can be used for different types of rendering and general-purpose compute workloads, it is also ensured that the hardware safety mechanisms used to detect faults are agnostic of the workload. This means that the hardware safety check is transparent to the application executing on the GPU such that the portability of the application is not affected.

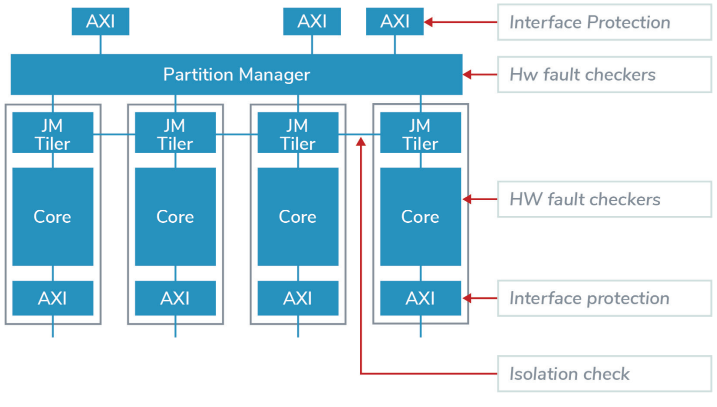


Figure 7 GPU architecture and hardware safety mechanisms for supporting ASIL B

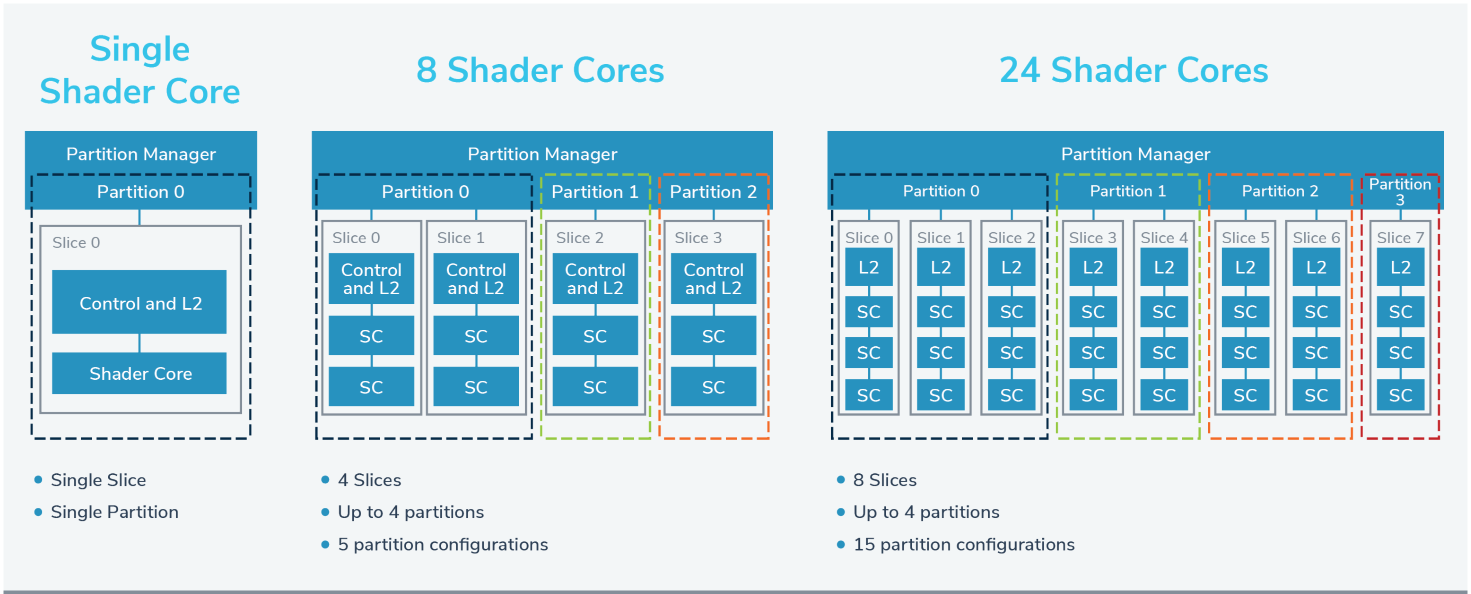


Figure 8 Scalable GPU architecture

4.1.4 Scalable Design for Entry-level to Premium Cockpit

The highly scalable architecture of GPUs, as shown in **Figure 8**, helps system designers select the right amount of computing resources depending on the performance and cost target. It also allows silicon vendors targeting different products catering to entry-level to high-end requirements to use the same hardware and software architecture across the products family, which is key to saving costs. The ability to create up to 4 flexible partitions out of an available GPU slices allows given hardware to be reconfigured according to the needs of different workloads executing on the GPU.

4.2 Scalable Safety-Critical Software Stack

A fundamental way to reduce the total cost of ownership for software is through the adoption of open standards. This allows a developer to leverage a standard interface while giving the freedom to differentiate at the application level. This is not a new concept and can be seen across the software landscape. However, the industry has been looking at the benefits of open standards and working to apply them to the safety domain with proof points such as OpenGL SC, a safety-critical implementation of the popular OpenGL standard, and the recently ratified Vulkan SC standard, a safety-critical version of the popular Vulkan API.

4.2.1 Vulkan SC as the Safe GPU Interface

The Vulkan SC API can be used as a foundational GPU driver layer to enable the application to interact with the safe GPU facilitating rich, high-performance, safe graphics for the modern vehicle displays. This approach can also be extended to applications where the safe GPU is being used for computing applications such as ADAS and autonomy functions, but that falls outside this paper's scope.

4.2.2 OpenGL for Safety

Where the application does not natively interface to the Vulkan SC API, it can still be leveraged as an abstraction layer. In this case, the OpenGL SC library sits ‘on top of’ the Vulkan SC API, exposes the OpenGL API to the application, and translates it via the Vulkan layer to the GPU. This is especially useful if the system developer needs to deliver safe graphics but is perhaps migrating from applications that utilize the embedded OpenGL ES.

4.2.3 Software Flexibility and Safety

Adopting this approach affords the developer the vast benefits of open standards while delivering a clear path to safety certification. While most safe graphics applications within today’s vehicles only need to achieve ASIL B, the Vulkan SC and OpenGL SC elements can be certified up to an ASIL D level. Additionally, this stack can utilize the flexible partitioning features of the underlying safe GPU leading to high performance, mixed-criticality systems such as cockpit domain controllers.

4.3 HMI Tool for SC Graphics

As UX and UI design continues to develop during your program and considering the rapid response required to the competitive landscape, there will be many iterations. Selecting the appropriate HMI graphics tool is critical to the success of any program. Below are a series of guidelines to consider when choosing the right tools for HMI development, specifically pertaining to safety-critical requirements.

4.3.1 Pre-Certified

An HMI tool with pre-certification to a higher level of certification than your initial requirements can save precious reengineering time as requirements morph throughout the program. For example, if your requirement is for ASIL B and you are presented with ASIL D support options at no extra effort or cost, that option should be the top pick. There are several permutations of the process to become pre-certified that the following V model **(Figure 9)** topically sums up.

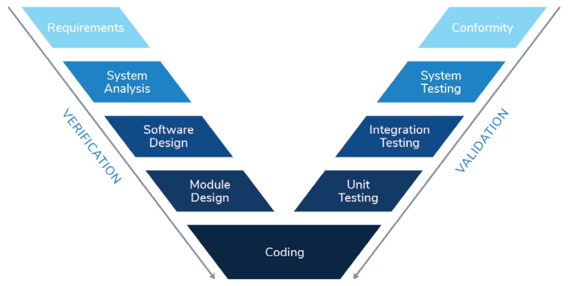


Figure 9 V-Model

While this works for the runtime software, depending on the level of functional safety certification, the process may also require tool qualification. Higher levels of functional safety certification require a more rigorous approach. For automotive specifically, functional safety is a function of Severity, Exposure, and Controllability, as seen in the graph below **(Figure 10)**.

Severity	Exposure	Controllability		
		C1 - Simple	C2 - Normal	C3 - Difficult
S1 - Light	E1 (very low)	QM	QM	QM
	E2 (low)	QM	QM	QM
	E3 (medium)	QM	QM	A
	E4 (high)	QM	A	B
S2 - Severe	E1 (very low)	QM	QM	QM
	E2 (low)	QM	QM	A
	E3 (medium)	QM	A	B
	E4 (high)	A	B	C
S3 - Fatal	E1 (very low)	QM	QM	QM
	E2 (low)	QM	A	B
	E3 (medium)	A	B	C
	E4 (high)	B	C	D

Figure 10 Defining the Automotive Safety Integrity Level (ASIL) that is to be applied to the element in question.

4.3.2 Strong History and Experience

The more experience an HMI tool vendor has in functional safety, the less risk your program will take on in safety-critical requirements. In looking at other markets, software functional safety processes and procedures will have different levels of rigor. Always consider the industries with more rigor than required when taking technology and design techniques from those industries. In aviation, for example, the software does not only need to be certified to the DO-178C standard, but the tools that create the software require qualification to corresponding DO-330 TQL levels. Leveraging tools that have been through this level of rigor can give the developers the confidence that the runtime and content generated from the tool will meet lower levels of functional safety.

4.3.3 Target Portability for Rapid Iteration Cycles

The HMI software development environment should be agnostic to the target hardware and OS or RTOS/OS. This is another effort to futureproof the HMI development effort. HMI tools can be well segmented and abstracted to have a Model View Presenter approach that keeps the content as modular and agnostic as possible. Reusability from target to target is essential to the long-term maintainability and supportability of the program **(Figure 11)**.

Suppose the HMI is developed to a specific platform, and in 6 months, a superior target SOM or CPU is released for the same or lower price. You should have the flexibility to move to the superior target without adding software risk to the program.

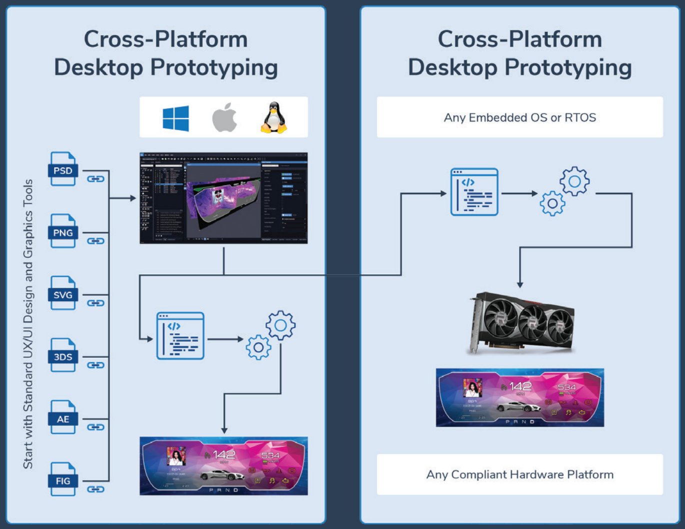

Cross-Platform Desktop Prototyping
Start with Standard UX/UI Design and Graphics Tools
PSD
PNG
SVG
3DS
AE
FIG
Cross-Platform Desktop Prototyping
Any Embedded OS or RTOS
Any Compliant Hardware Platform

Figure 11 Rapid Portability from Desktop to Target

4.3.4 Mixing ASIL Levels with QM

A pure safety software HMI approach, while significantly more reliable, may also prove to be more fiscally challenged depending on the level of tools you use. Here again, precertification is vital to cost savings, but further cost savings could be found in segmenting the ASIL parts of the HMI to separate layers and compositing those layers together on the target.

Having both ASIL and QM in the same display is a process called mixed criticality **(Figure 12)**. There are also tools available that can take the ASIL and QM contingent together in the same design canvas providing a smooth workflow process for mixed-criticality HMI. This, in turn, provides a very rapid iteration capability with both safety-critical and standard QM embedded content.

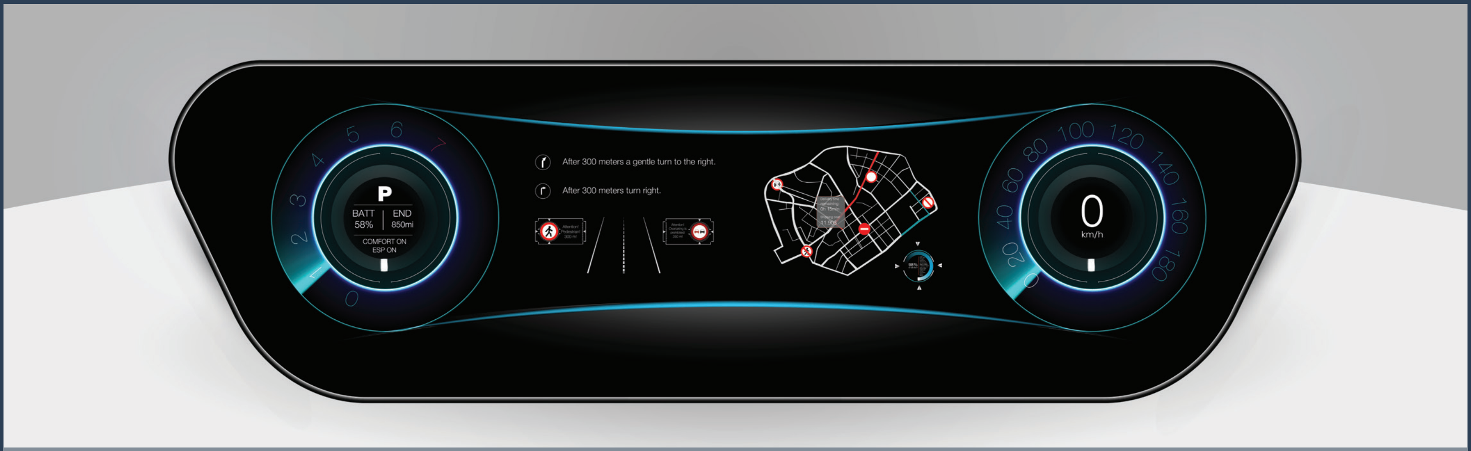

P
BATT 58%
END 850mi
COMFORT ON
ESP ON
After 300 meters a gentle turn to the right.
After 300 meters turn right.
0
km/h

Figure 12 Mixed-Criticality Example

5.0 Collaboration Benefits to Automotive Ecosystem for Digital Cockpits

5.1. Automotive Supply Chain for SC Digital Cockpit

5.1.1 Supply Chain Hierarchy

The traditional automotive supply chain is complex **(Figure 13)**, although this is undergoing a shift with the emergence of OEMs looking to develop hardware and software in-house, bringing in vertical integration and potentially shortening the length of the supply chain.

This doesn't show the different ecosystem players in the chain. They play a critical role in supplying various pieces of software, tools, and services to facilitate the system integration by Tier 1s and/or OEMs. These could include (for example) providers of graphics stack and HMI tooling used by system integrators (Tier 1s) to deploy safe HMI in a vehicle.

5.1.2 What Changes with Safety-Critical

The automotive supply chain can play a crucial role in reducing cost and time to market associated with the development of functional safety products. While developing a safety-critical system, it is vital that all the individual elements that constitute the system, when integrated, help achieve the system-level safety goal. This can be achieved by ensuring that the individual components are developed following relevant safety standards to meet safety requirements allocated to these elements based on the system-level functional safety concept. The pre-certified pieces such as GPU IP, software stack, and tooling is a step in this regard. Furthermore, we have ensured that these pre-certified pieces work together to a give a pre-integrated solution for safety-critical graphics rendering that works. This will support system integrators while preparing to develop software and hardware in-house, as they have access to this solution while selecting the hardware itself.

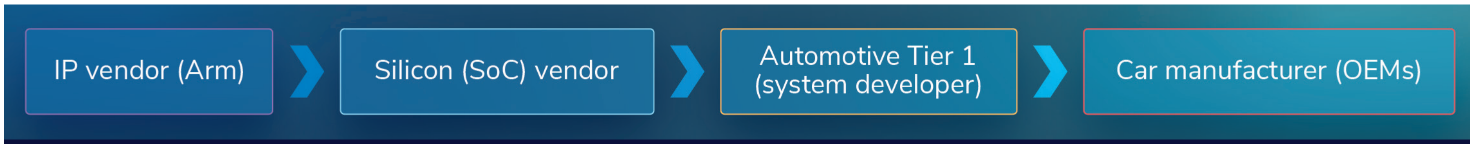


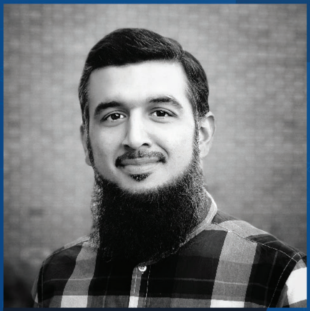
Figure 13 A traditional automotive supply for the deployment of GPU IP

Conclusion

Having the appropriate architecture in place before developing any capable safety system is paramount to the program's success. Finding the partner ecosystem where all sub-specialty parts have been pre-integrated will save any OEM or Tier 1 supplier significant time and effort in developing, testing, iteration, and fielding a new system. This paper covered some of the more critical components of such a system, from the evolution of HMI software safety and considerations of what the software stack should look like to the importance of mixed-criticality and Safe Computing with the supplier interactions within the supply chain. In developing any architecture, it is prudent to consider the latest technology, how it is currently employed and what other markets may be more mature where developers can learn from others' mistakes and triumphs.

As the complexity of modern automotive architecture systems becomes ever-more reliant on emerging technology, implementing safety-critical focused hardware and software stacks is paramount. This approach improves time-to-market and provides a reduction in development costs and certification, but ultimately it reduces risks for the OEM, culminating in a safer driving experience.

About the Authors



arm

Mustafa Ali

Senior Product Manager, Automotive & IoT Line of Business

Mustafa joined Arm in 2018 as a Product Manager in the automotive line of business. He has been working in the semiconductor IP industry for the last 10 years, having held product management roles for systems IP, embedded GPUs, and machine learning accelerator IP. Mustafa holds a Ph.D. in electronics engineering from Southampton University UK.

DiSTI

Christopher Giordano

Vice President UX/UI Technology

Since 1997, Chris has focused on developing UI and HMI Software starting at the U.S. Navy and University of Central Florida. Chris has worked at DiSTI since 1999 as lead engineer or program manager for over 60 different programs, and eventually the product manager for all DiSTI’s UI development tools. Chris managed DiSTI’s HMI/UI programs for Boeing, Hyundai, Jaguar Land Rover, Lockheed, NASA, Nissan Motors, Northrop Grumman and The Space Ship Company to name a few and is currently DiSTI’s VP of UX/UI Technology. He has successfully managed DiSTI's UX/UI business for over a decade developing a global leadership position as experts in HMI/UI and Functional Safety. Chris holds bachelor degrees in Finance from UNCW and Computer Engineering from UCF graduating with honors.

CoreAVI INNOVATING A SAFER TOMORROW

Neil Stroud

Vice President of Marketing, Business Development & Product Management

Neil Stroud joined CoreAVI in November 2020 as the VP, of Marketing & Business Development. He has spent over 25 years in the semiconductor industry in technical, commercial, sales, and strategy roles. For the past 15 years, Neil has been an industry leader in the functional safety domain working closely with the ecosystem, customers, and partners across the automotive, industrial, avionics and transportation segments to build a safer world. Neil has held various positions in world-class corporations including Samsung, NEC, PMC-Sierra, and Intel. Immediately prior to joining CoreAVI, Neil worked at Arm, during which time his roles included Senior Director of Technology Strategy where he led a team focused on functional safety, security, machine learning and real-time compute for the automotive and IoT business unit, as well as leading the activities in industrial automation and robotics.

ISBN: 978-0-578-38940-0