Growing Importance of Functional Safety

Demand for advanced GPU-accelerated graphics and compute is growing in an increasing number of industries where safety is paramount, such as automotive, autonomy, avionics, medical, industrial, and energy.

In safety-critical systems a compute or display system failure would pose a significant safety risk.
Functional Safety Certification

Safety Certification
Performed at the system level

Development Process defined in safety-critical standards
1) Document system design, safety requirements, software architecture and software design
2) Test and verify at each level against design documentation
3) Provide certification evidence packages to demonstrate documentation and testing

Reducing certification effort and costs
System runtime components should:
1) Be streamlined as far as possible to reduce documentation and testing surface area
2) Have deterministic behavior to simplify design and testing
3) Implement robust and unambiguous fault handling

In the ISO 26262 V-Model system development process testing and verification occur in reverse order from design and implementation

Industry safety-critical standards include
RTCA DO-178C Level A / EASA ED-12C Level A (avionics)
ISO 26262 ASIL D (automotive)
IEC 61508 (industrial)
IEC 62304 (medical)
Safety Certification and Open Standard APIs

Need for APIs to streamline system-level safety-critical certifications
- Streamlined
- Deterministic
- Robust

Growing need for embedded hardware acceleration
- Advanced processing of multiple advanced sensors
- Smart systems through machine learning and inferencing
  - Advanced displays and user interfaces

Growing need for well-defined hardware software interoperability in safety critical industry
- Decoupling software and hardware for easier development and integration of new components
  - Cross-generation reusability
  - Cross-platform reusability
  - Field upgradability

Growing demand for state-of-the-art open, cross-vendor, acceleration API standards that are designed by and for the safety-critical industry
Khronos has close to 20 years experience in adapting mainstream APIs for safety-critical markets
Leveraging proven mainstream APIs with shipping silicon implementations and developer tooling and familiarity

**Vulkan SC targets any systems requiring safety critical graphics and/or compute**
E.g., automotive, autonomy, avionics, medical, industrial, and energy

**Vulkan SC has significantly higher performance and flexibility than OpenGL SC**
Enabling new safety-critical markets requiring graphics and compute AND cross-platform standalone compute
OpenGL SC will continue to be supported by Khronos, but new developments will focus on Vulkan SC
Vulkan SC 1.0 Design Philosophy

Vulkan 1.2 is a compelling starting point
- Widely adopted, royalty-free open standard
- Explicit control of device scheduling, synchronization and resource management
- Smaller surface area than OpenGL
- Not burdened by runtime debug functionality
- Very little internal state
- Well-defined thread behavior
- Ingests SPIR-V IR - no runtime front-end compiler

Streamlined
- Remove non-essential runtime functionality
  - Sparse memory
  - Descriptor update templates
  - Certain types of object deleters

Deterministic
- Predictable execution times and results
  - Offline compilation of pipelines
  - Static memory allocation

Robust
- Removing Ambiguity
  - No ignored parameters or undefined behaviors
  - Enhanced fault handling and reporting functionality
  - Rigorous conformance test suite
  - MISRA C alignment

Vulkan SC enables system implementers deploying GPU-accelerated graphics and compute to meet safety-critical obligations and provide certification evidence packages with reduced cost and effort.

Vulkan SC can also be invaluable for real-time embedded applications, even if not formally safety-certified.

Vulkan SC is a compelling starting point. Widely adopted, royalty-free, it offers explicit control of device scheduling, synchronization, and resource management. It has a smaller surface area than OpenGL and is not burdened by runtime debug functionality. With very little internal state, it offers well-defined thread behavior. Ingesting SPIR-V IR directly, it avoids the need for a runtime front-end compiler.

Streamlined features include the removal of non-essential runtime functionality, such as sparse memory, descriptor update templates, and certain types of object deleters.

Deterministic properties include predictable execution times and results, achievable through offline compilation of pipelines and static memory allocation.

Robust qualities include removing ambiguity, ensuring no ignored parameters or undefined behaviors. It also supports enhanced fault handling and reporting functionality, backed by a rigorous conformance test suite and alignment with MISRA C standards.

Vulkan SC is designed to enable system implementers to deploy GPU-accelerated graphics and compute while meeting safety-critical obligations, providing certification evidence packages at a reduced cost and effort. It is also valuable for real-time embedded applications, even if not formally safety-certified.
Vulkan SC Robustness

Fault Handling and Reporting
Application registers functions at device creation which the driver can call if a fault is detected
Application can interrogate type and level of a fault together with implementation-specific data

Vulkan SC Conformance Test Suite
Freely available to all under Apache 2.0 open-source license
Leverages extensive Vulkan test suite with added SC-specific tests
System integrators can use to confirm and document Vulkan SC implementation compatibility

MISRA C
Vulkan SC 1.0 is aligned with MISRA C software development guidelines
Developed by the MISRA Consortium for embedded system code safety, security, portability and reliability and alignment with safety-critical standards
Vulkan SC Offline Compiled Pipelines

A Vulkan Pipeline defines how the GPU processes data

- SPIR-V
  - JSON
    - Pipeline Description
      - Lists all SPIR-V modules used with related state

Implementation-Specific Pipeline Cache Compiler (PCC)

Pipeline Cache Utility

- Pipeline Cache Containers
  - Extracts information from pipeline cache files to analyze dataflow and the amount of memory used by the processing in the pipeline

Application

- Memory for pipelines is reserved at device at device creation time as fixed size pools. Similarly sized pipelines can be assigned to the same pool to minimize memory size and fragmentation. Avoids need for runtime memory allocation

Offline | Runtime
Vulkan SC Static Memory Allocation

1. Define the upper bound of number and size of objects of all types that will exist at any point

2. Pre-allocate host-side memory structures for the maximum number and size of each object

3. Create and destroy objects as needed within upper bound

4. Store created objects within pre-allocated memory, the driver does not need to perform any runtime memory allocations

Static memory allocation eliminates non-deterministic behavior caused by memory allocations, and possible memory allocation errors, happening at random times at runtime
Porting from Vulkan to Vulkan SC

Developers will be able to use desktop Vulkan for initial development and leverage widely available drivers and powerful development tools.

JSON Generator Layer creates Pipeline Descriptions and memory object data for direct use by the Vulkan SC-ported version of the application.

IN DEVELOPMENT

Vulkan version of application with online pipeline compilation and dynamic memory allocation

JSON Generator Layer

Pipeline Cache Compiler (PCC)

Pipeline Cache Container

Vulkan SC version of application with offline compilation of pipelines and static memory allocation

Memory object counts and maximums for all object types
Call to Action!

- **Vulkan SC** enables many levels of the safety-critical ecosystem
  - Device Manufacturers of GPUs and SoCs
  - Driver Vendors, System Builders
  - Middleware Developers, Application Developers

- **Implementers**
  - [Vulkan SC 1.0 specification](#) and open-source Conformance Test Suite are freely available

- **Middleware, and Application Developers**
  - Ask your hardware vendor for Vulkan SC drivers
  - [Conformant Implementations](#) are running today on CoreAVI, and NVIDIA DRIVE and Jetson Platforms

- **System Builders**
  - Leverage Vulkan SC for high performance safety-critical graphics and compute
  - Use Vulkan SC in embedded real-time systems

- **Everyone**
  - Engage Vulkan SC working group and community at the [Vulkan SC specification GitHub](#)

Vulkan SC will broaden the adoption of GPU acceleration in safety-critical systems and real-time applications