Vulkanised 2025

The 7th Vulkan Developer Conference
Cambridge, UK | February 11-13, 2025

Welcome and Vulkan Update

Ralph Potter, Vulkan Working Group Chair
Samsung Electronics

Overview

- **Introduction**
- **Looking back at 2024**
  - Roadmap 2024
  - Vulkan 1.4
  - Additional Extensions
  - Shader Ecosystem
- **Where do we go from here?**
- **Questions & Answers**

So Long, and Thanks for All the Fish!

Vulkan

An explicit API for graphics and compute on GPUs

- Radically cross-platform, from embedded to desktop
- Focus on high performance and user control

Driving the future evolution of graphics hardware

- Setting requirements for new hardware
- Ensuring compatibility with current hardware
- Focus on solving issues raised by industry experts

Developed collaboratively by industry experts

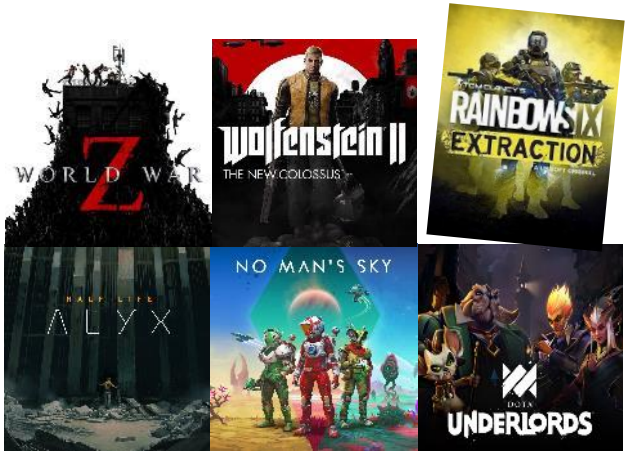
- Input considered from a wide range of sources

Vulkan is Everywhere

Desktop and Mobile GPUs and SOCs

AMD, BROADCOM, arm, Google, HUAWEI, Imagination, intel, NVIDIA, MEDIATEK, QUALCOMM, SAMSUNG, VeriSilicon

http://vulkan.gpuinfo.org/

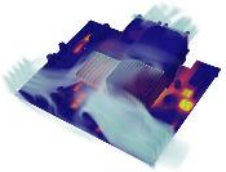


Desktop Games

Mobile Games

AUTODESK FUSION 360

Cross-platform post-processing and display of simulation results



Adobe, Substance 3D Stager

Cross-platform ray tracing

Applications

Engines

CIDER, CINDER, XENKO, GODOT, CRYENGINE, Unity, source, id, UNREAL ENGINE, Croteam Serious Engine, UX3D, NetEase Games

Note: The version of Vulkan available will depend on platform and vendor

Looking Back at 2024

Roadmap 2024

Describes capabilities of mid-high end HW releasing in 2024
Communicating direction, both to WG and developers
Released alongside Vulkanised 2024 (Q1 2024)

Vulkan 1.4

Describes capabilities of current mainstream HW
Released December 2024 (SDK available January 2025)

Additional Extensions

Pipeline Binaries, Device-Generated Commands, Vulkan Video, and more…

Ecosystem Updates

Slang/HLSL

Vulkan Roadmap: Setting industry direction

The roadmap sets forward-looking targets for new hardware, with milestones established for new mid-to-high-end GPUs in the year of their release

Roadmap **2022**

Roadmap **2024**

Roadmap **2026**

Core Specifications: Support for Current GPUs

Features that can be supported on current GPUs are brought into core

Maintenance Updates

Vulkan **Core 1.0**

Vulkan **Core 1.1**

Vulkan **Core 1.2**

Vulkan **Core 1.3**

Vulkan **Core 1.4**

Vulkan **Core 1.5**

2016 2018 2020 2022 2024

Roadmap 2024

Required Extensions

VK_KHR_dynamic_rendering_local_read
VK_KHR_shader_maximal_reconvergence
VK_KHR_shader_quad_control
And more..

Additional Features

Shader half-float and 8/16-bit integer types,
multi-draw indirect,push descriptors, and more …

Everything in Roadmap 2022 plus 14 extensions, 8 required features

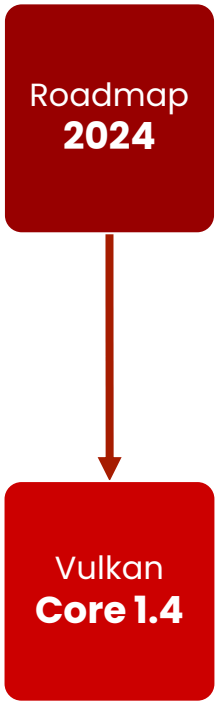
https://github.com/KhronosGroup/Vulkan-Docs/blob/main/appendices/roadmap/Roadmap-2024.adoc

KHRONOS GROUP

Vulkan 1.4

Strengthening the Vulkan Ecosystem

Roadmap Sets Direction, Core Solidifies It



- **Vulkan 1.4 is the first core version derived from the roadmap**
  - Notable benefits both in design and development
  - Enabled huge increase in supported features
- **Most of the tough questions for 1.4 largely already answered**
  - Future direction already set with the roadmap
  - Features already designed, shipped, and implemented
  - Vendors already knew which hardware could support what
- **“Just” had to put the pieces together**
  - Much easier development cycle than previous cores
  - Allowed us to focus on future roadmap items

Vulkan 1.4 Core Specification

Integrates significant requested functionality proven as extensions

Mandated support for new functionality ensures availability on all Vulkan 1.4 implementations

Dynamic rendering local read bringing subpass support to the dynamic rendering API

Streaming transfers via host image copy or mandatory async transfer queue support

Fine-grained control of floating point optimization behavior

Mandating previously optional features such as scalar block layout and 8/16 integer support

Maintenance extensions up to VK_KHR_maintenance6

Several limit increases, including 8K rendering with up to eight separate render targets

And more…

16 extensions in total

Raising the Bar

Vulkan 1.0 was designed to run on GLES 3.1-class GPUs (circa 2014)

Core versions need to run on the broadest set of devices

Vulkan 1.4 raises minimum hardware requirements

Optional functionality and artificially low limits increase complexity for developers

Makes 28 previously optional features mandatory, including scalar block layout and 8/16 bit integer support in shaders

Raises minimum limits on 31 properties

Provides reliable access to functionality across all supported platforms

Streaming Transfers

Streaming image resources without interrupting rendering

Previously required copies on GPU timeline

VK_EXT_host_image_copy (optionally) promoted to core

Enables CPU-side image copies

If host copy is not supported, then a dedicated asynchronous transfer queue is mandatory

https://www.khronos.org/blog/copying-images-on-the-host-in-vulkan

Dynamic Rendering Local Read

Vulkan 1.3 promoted dynamic rendering to core…

- Removed the need for render pass and framebuffer objects
- Greatly simplified the programming model

…but the original extension didn't address input attachments or subpasses

- Critical for performance on tile-based GPUs

Vulkan 1.4 includes local reads for color attachments/storage resources

- Closes the gap versus legacy render passes
- Local reads for depth/stencil/multisampled attachments are optional

https://www.khronos.org/blog/streamlining-subpasses

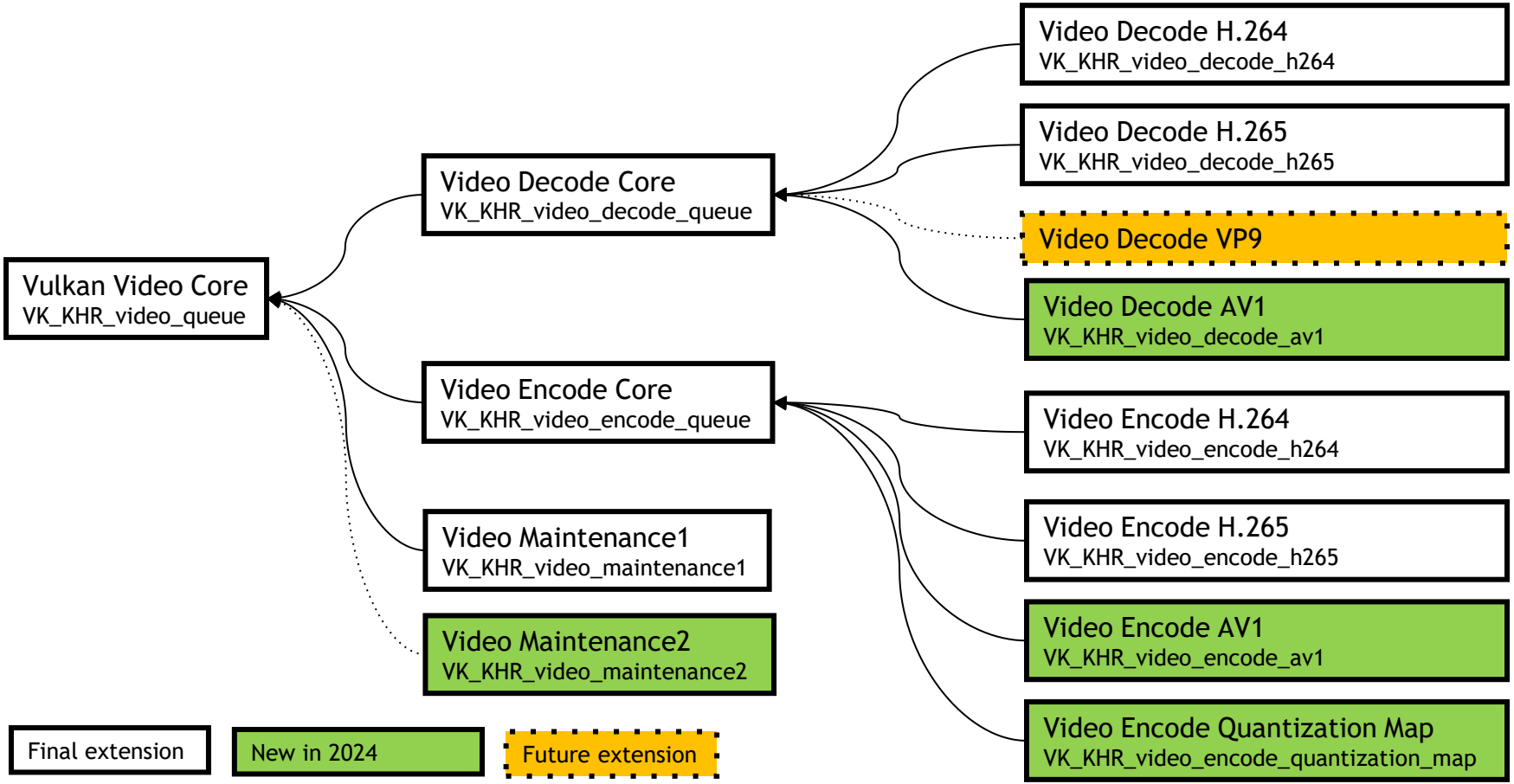
New Extensions

- **32 new extensions since Vulkanised 2024**
  - 10 KHR, 8 multi-vendor EXT, 14 vendor extensions

VK_KHR_compute_shader_derivatives
VK_KHR_depth_clamp_zero_one
VK_KHR_maintenance7
VK_KHR_maintenance8
VK_KHR_pipeline_binary
VK_KHR_shader_relaxed_extended_instruction
VK_KHR_video_decode_av1
VK_KHR_video_encode_av1
VK_KHR_video_encode_quantization_map
VK_KHR_video_maintenance2

VK_EXT_depth_clamp_control
VK_EXT_device_generated_commands
VK_EXT_external_memory_metal
VK_EXT_legacy_vertex_attributes
VK_EXT_map_memory_placed
VK_EXT_present_mode_fifo_latest_ready
VK_EXT_shader_replicated_composites
VK_EXT_vertex_attribute_robustness

Vulkan Video



Shader Ecosystem Changes

- **Slang**
  - Khronos have adopted NVIDIA's slang compiler as a Khronos-hosted open-source project
  - Now under multi-company governance
  - https://github.com/shader-slang
- **HLSL**
  - Migrating to clang/LLVM
  - Direct3D will adopt SPIR-V as the default interchange format
    - https://devblogs.microsoft.com/directx/directx-adopting-spir-v/
  - HLSL now has a defined proposal process
    - https://github.com/microsoft/hlsl-specs/blob/main/docs/Process.md

What we're working on

Evolving the API

Disclaimer: Predicting the future is hard

We have a Roadmap, but it's an evolving document
Some items on here are close to final
Some require hardware changes, and may be years out

Our Roadmap is driven by your feedback

We prioritize based on developer feedback

Roadmap Topics

Robustness and Constraining Undefined Behaviour

WebGPU is likely to become a major source of content

Better support for safety-conscious languages like Rust

Shader range checking, poison/freeze, data race handling

Unifying Compute

Support HPC use cases. Feature parity with OpenCL

Untyped pointers, 64-bit addressing,

Unify OpenCl and Vulkan SPIR-V dialects, replace buffers with device addresses

Crash Debugging Enhancements

Cross-vendor progress markers, enhanced device fault reporting,
shader abort

State management and Synchronization

Pipelines, state and synchronization are major pain points

These two topics are our largest usability challenges

State Management

More widely supportable Shader Object extension

Cleaner, simpler API for descriptor management

Synchronization

Usability improvements for image layouts in-flight

On-going discussion about whether anything else can be simplified

Machine Learning

Machine Learning for graphics will be everywhere

Lots of compelling use-cases (upscaling, denoising, neural texturing etc.)

Vulkan not aiming to be a general ML standard

Instead provide the foundations for existing engines and APIs

Foundational Tools

Cooperative matrix, ML-friendly data types (bfloat16 etc.),
more shader support for DLA hardware

More Information

- **Vulkan: https://www.vulkan.org/**
  - Vulkan 1.4 Press Release: https://khr.io/vulkan14
  - Specification: https://docs.vulkan.org/spec/latest
  - Spec GitHub Repo: https://github.com/KhronosGroup/Vulkan-Docs/
  - Discord Link for community discussion: https://discord.com/invite/vulkan

Thank You!

Vulkan Working Group, Khronos F2F Meeting, Seattle, September 2024