Vulkanised 2024

The 6th Vulkan Developer Conference
Sunnyvale, California | February 5-7, 2024

Vulkan Video Encode APIs

Tony Zlatinski, NVIDIA

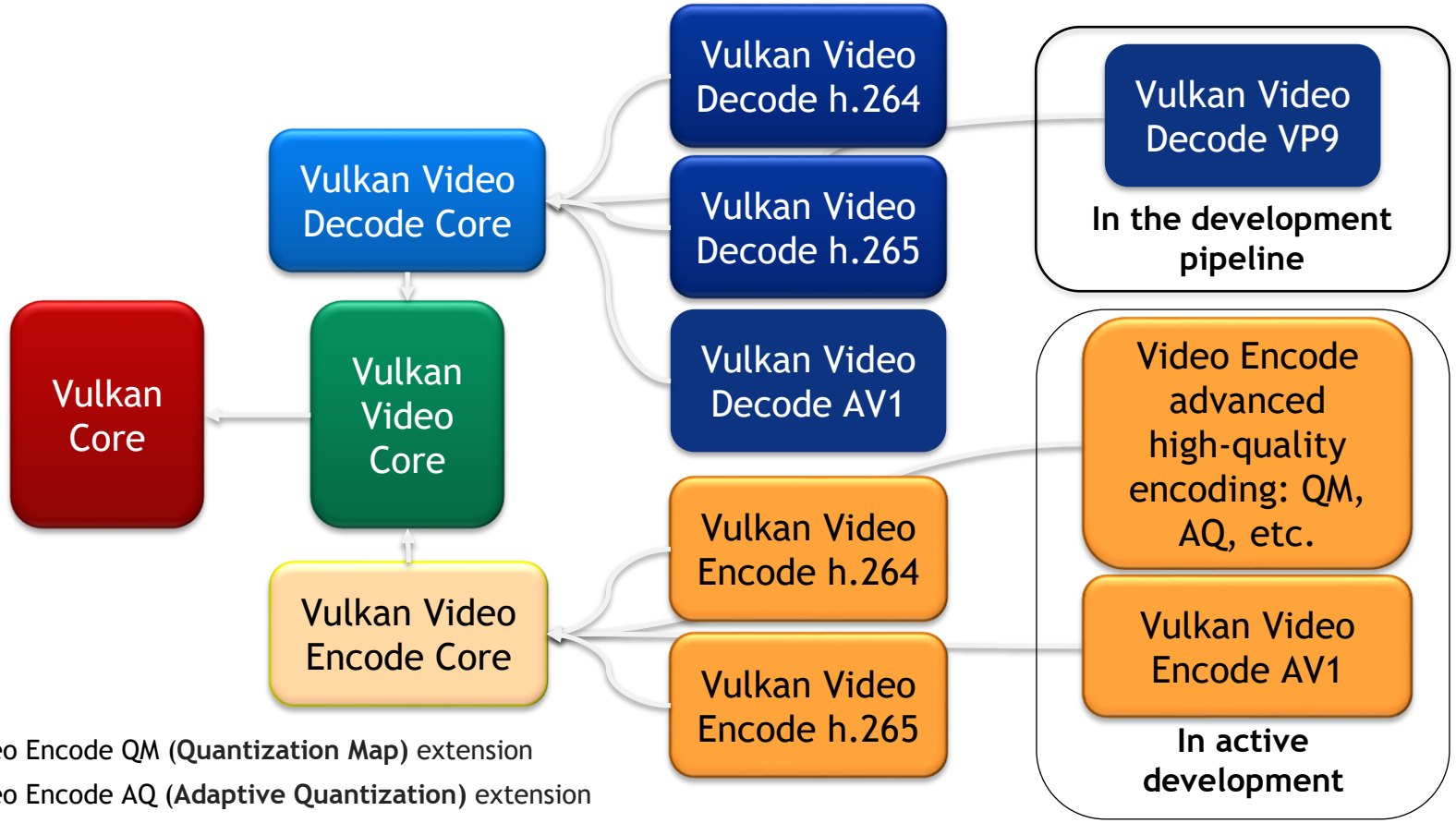
Presentation Outline

- Vulkan video roadmap
- General Vulkan video encode concepts and architecture
- Vulkan video encode quality and rate control
- Vulkan video tools

Vulkan Video API Design Goals

- **Cross-platform and vendor-neutral low-level HW state(-less) video codecs API**
- **Closer integration with graphics, compute and display**
- **Improved low-level synchronization reducing the processing delays**
- **Support for scalable applications, from high-performance servers to energy-efficient devices with limited resources**
  - An optimized solution for processing multiple video streams in parallel, efficiently distributing tasks across various CPU and hardware codec cores
  - Reduced latency and execution overhead during processing
  - Minimized overhead of CPU, GPU, ASIC, and memory resources

Vulkan Video Core and Codec Extensions



Video Encode QM (**Quantization Map**) extension

Video Encode AQ (**Adaptive Quantization**) extension

AVC (H.264) and HEVC (H.265) and AV1 driver availability

Intel will support **H.264/5 video encode** and **AV1 video decode** through Vulkan Video extension for all Intel® Arc™ Graphics products. Please download the driver for H.264/5 video decoding support.

AMD supports Vulkan Video extensions on all RDNA™ architecture-based graphics with AMD Software: Adrenalin Edition™ drivers for Windows®. Please download the latest driver with production support for **H.264/5 decode** and BETA support for **H.264/5 encode** and **decode AV1**.

NVIDIA released Windows and Linux drivers available for **H.264/5 decode** and **encode** starting with Maxwell/Pascal and later (the latest one being ADA) GPUs and **AV1 decode** available for Ampere and later GPUs

The latest information about **Open-Source drivers** for Vulkan Video **for RADV/AMD** and **ANV/Intel** can be found on Dave Airlie's blog.

Vulkan Video Support

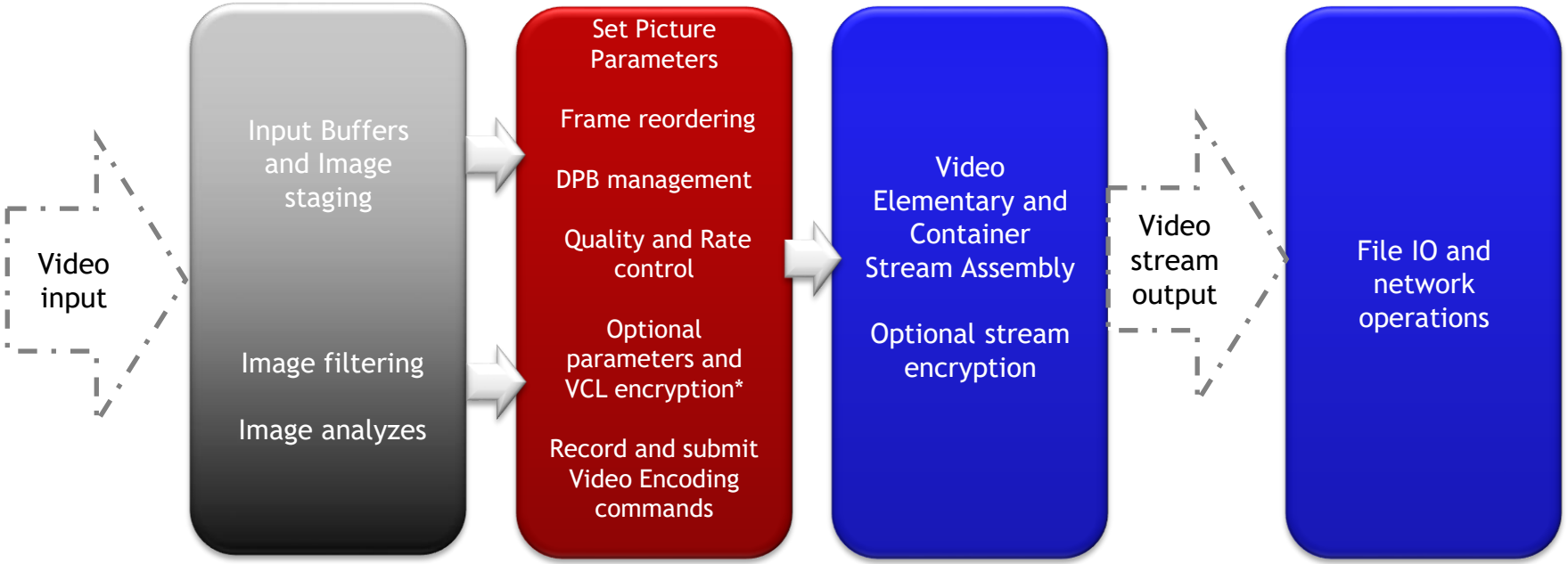
NVIDIA open-source vk_video_decode_and_encode sample

Early adoption by leading multimedia frameworks:

- GStreamer (including support by Igalia and COLLABORA)
- FFmpeg (including development by Lynne)

Vulkan Video Encode Execution Model

Video Encode Application Processing Pipeline



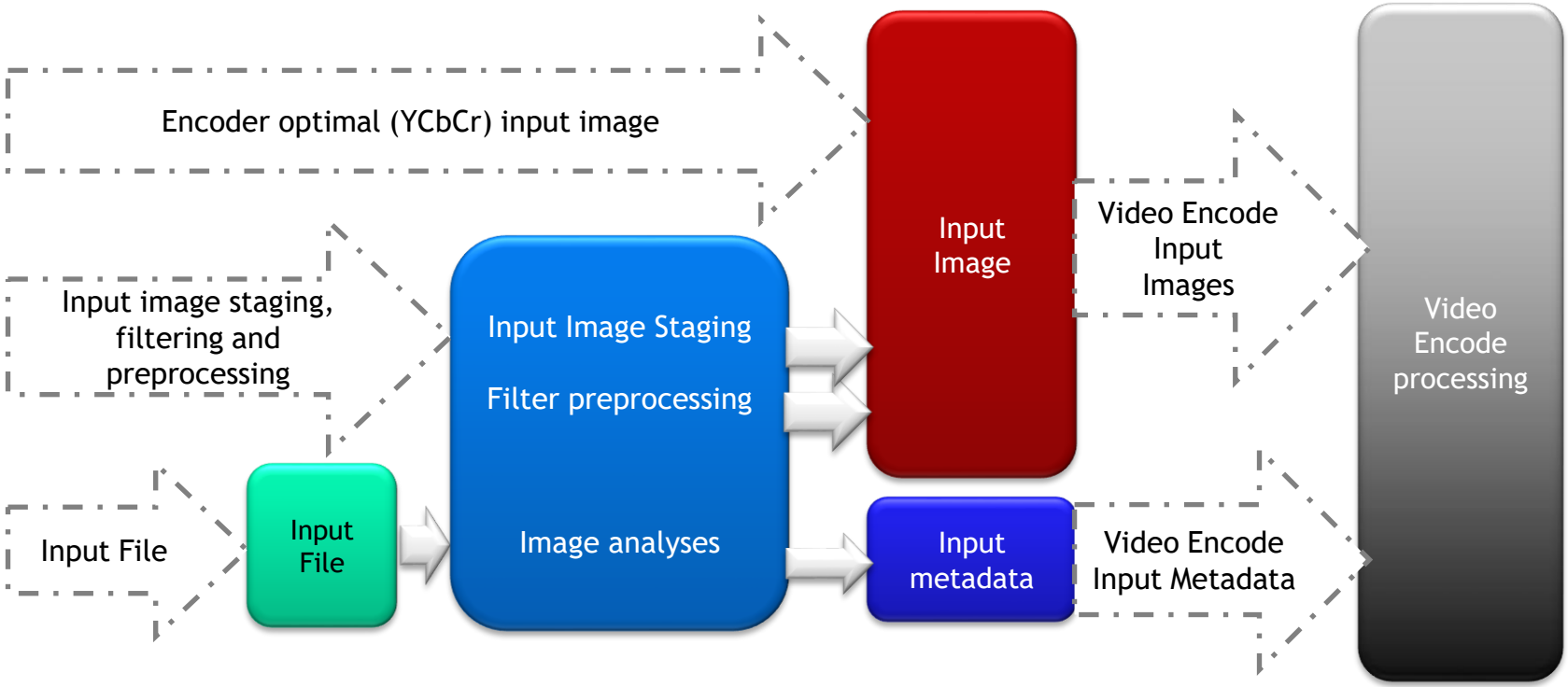
Vulkan HW accelerators and host (CPU) processing

Parameters and VCL encryption features are coming soon*

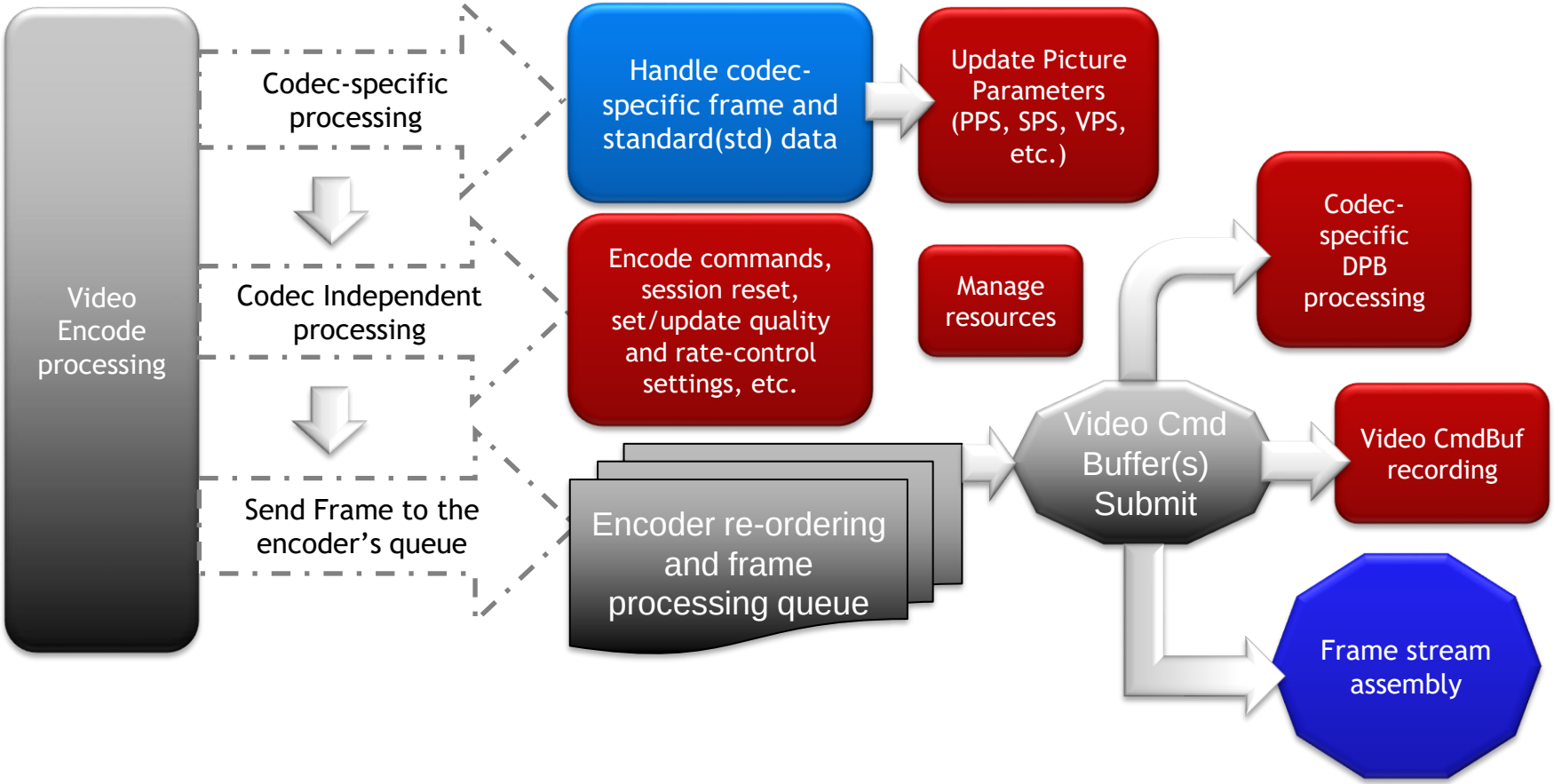
Vulkan HW accelerators processing

Host (CPU) application processing

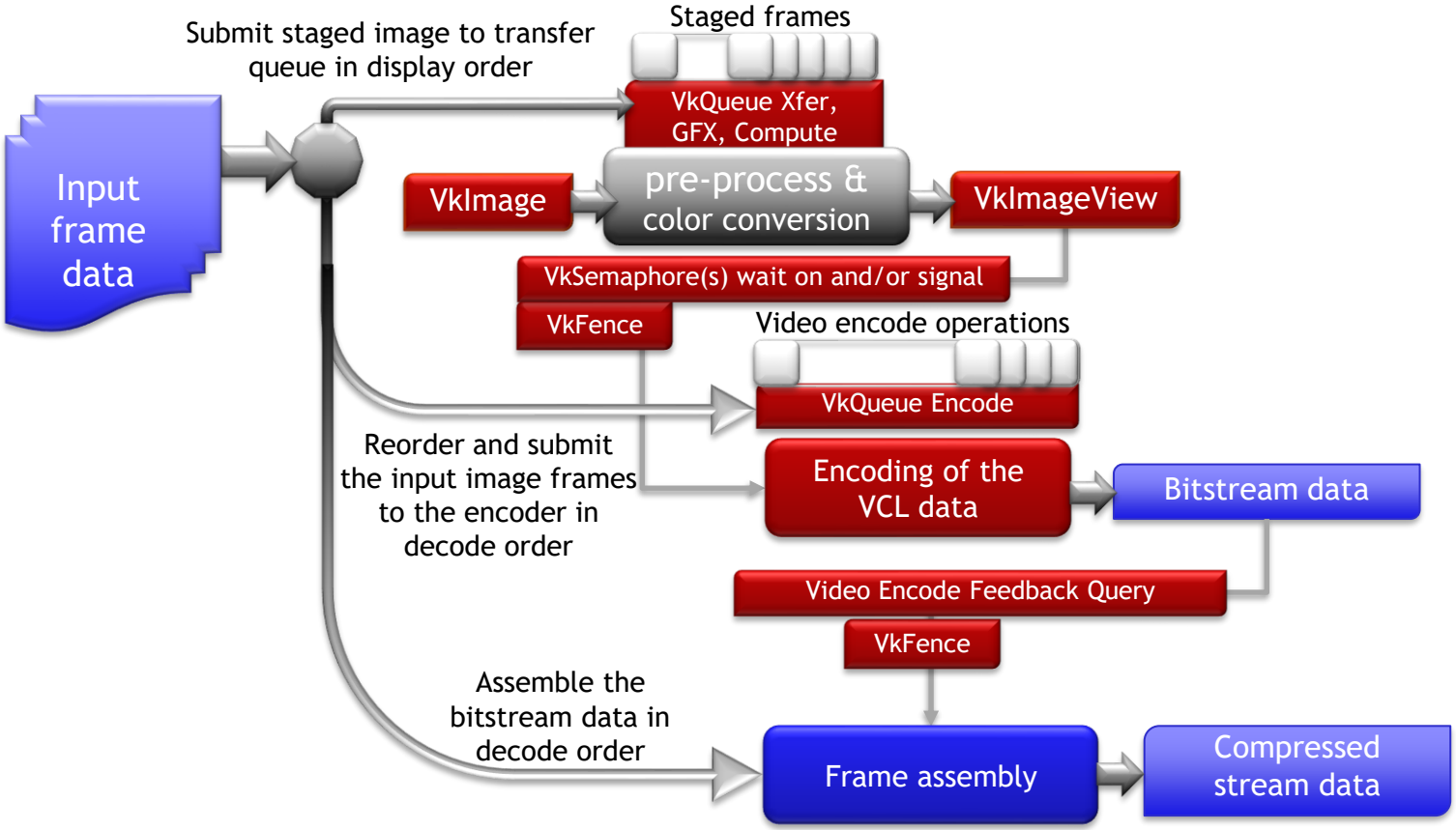
Input Application Processing Pipeline



Video Encode Application Processing Pipeline



Interaction Between Input and Encoding Stages



Before encoding the video sequence in Vulkan

- **Create Vulkan Device with Video Encode Support Queue(s)**
- **Obtain the Video Encode Queue for submitting the work**
- **Query the Vulkan Video Physical device properties to gather the implementation requirements**
- **Create:**
  - Input and DPB Vulkan image resources
  - Vulkan buffer resources for the compressed output bitstream
  - Video profile compatible with the targeted video stream format
  - Video session for the compressed stream
  - Video session parameters
  - Command buffers for recording the video commands
  - Command buffers for recording the staging and pre-processing of the video input
  - Synchronization primitives: semaphores, fences and queries
- **Reset and configure the video session**

Vulkan Video Core Architecture Presentation

Vulkanised 2023 presentation on Vulkan Video Core

A Deep Dive into Vulkan Video

Video on YouTube and Slides

Video Encode Properties and Capabilities

Get the supported video encode capabilities by calling vkGetPhysicalDeviceVideoCapabilitiesKHR with a VkVideoEncodeCapabilitiesKHR structure chained with the VkVideoCapabilitiesKHR.

For H.264, chain additionally VkVideoEncodeH264CapabilitiesKHR.
For H.265, chain additionally VkVideoEncodeH265CapabilitiesKHR.

```
VkVideoEncodeCapabilitiesKHR structure:
    VkVideoEncodeCapabilityFlagBitsKHR          flags;
    VkVideoEncodeRateControlModeFlagBitsKHR     rateControlModes;
    uint32_t                                    maxRateControlLayers;
    uint64_t                                    maxBitrate;
    uint32_t                                    maxQualityLevels;
    VkExtent2D                                  encodeInputPictureGranularity;
    VkVideoEncodeFeedbackFlagBitsKHR            supportedEncodeFeedbackFlags;
```

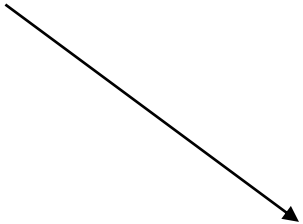
Video Encode Parameter Overrides

vkGetEncodedVideoSessionParametersKHR and optimizing overrides:

VkVideoSessionCreateInfoKHR::VK_VIDEO_SESSION_CREATE_ALLOW_ENCODE_PARAMETER_OPTIMIZATIONS_BIT_KHR

vkGetEncodedVideoSessionParametersKHR

```
VkDevice                                          device,
const VkVideoEncodeSessionParametersGetInfoKHR*   pVideoSessionParametersInfo,
VkVideoEncodeSessionParametersFeedbackInfoKHR*    pFeedbackInfo,
size_t*                                           pDataSize,
void*                                             pData
```

VkVideoEncodeSessionParametersFeedbackInfoKHR

```
VkBool32    hasOverrides;
```

VkVideoEncodeSessionParametersGetInfoKHR

```
VkVideoSessionParametersKHR    videoSessionParameters
```

Video Encode Feedback Queries

VkQueryPoolVideoEncodeFeedbackCreateInfoKHR

VkVideoEncodeFeedbackFlagBitsKHR:

- **VK_VIDEO_ENCODE_FEEDBACK_BITSTREAM_BUFFER_OFFSET_BIT_KHR**
- **VK_VIDEO_ENCODE_FEEDBACK_BITSTREAM_BYTES_WRITTEN_BIT_KHR**
- **VK_VIDEO_ENCODE_FEEDBACK_BITSTREAM_HAS_OVERRIDES_BIT_KHR**

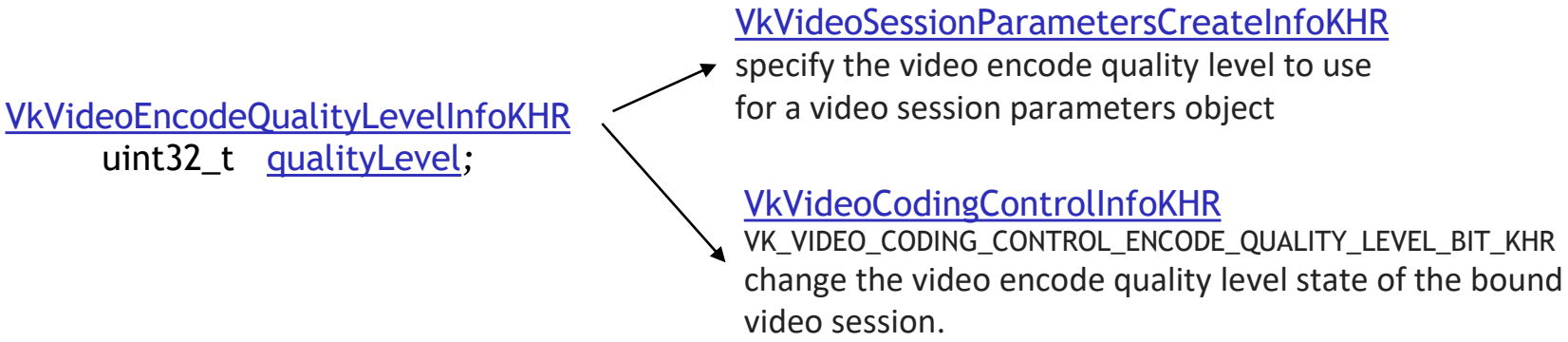
Inline Queries

- VK_KHR_video_maintenance1 extension adds support for Inline Queries

Vulkan video encode quality and rate control

Encoder’s Quality Level

Choose a qualityLevel *value that must be less than* VkVideoEncodeCapabilitiesKHR*::**maxQualityLevels** returned by the implementation via* vkGetPhysicalDeviceVideoCapabilitiesKHR



Quality usage flags from VkVideoEncodeUsageInfoKHR

- VkVideoEncodeUsageFlagBitsKHR
  - VK_VIDEO_ENCODE_USAGE_TRANSCODING_BIT_KHR
  - VK_VIDEO_ENCODE_USAGE_STREAMING_BIT_KHR
  - VK_VIDEO_ENCODE_USAGE_RECORDING_BIT_KHR
  - VK_VIDEO_ENCODE_USAGE_CONFERENCING_BIT_KHR
- VkVideoEncodeContentFlagBitsKHR
  - VK_VIDEO_ENCODE_CONTENT_CAMERA_BIT_KHR
  - VK_VIDEO_ENCODE_CONTENT_DESKTOP_BIT_KHR
  - VK_VIDEO_ENCODE_CONTENT_RENDERED_BIT_KHR
- VkVideoEncodeTuningModeKHR
  - VK_VIDEO_ENCODE_TUNING_MODE_DEFAULT_KHR
  - VK_VIDEO_ENCODE_TUNING_MODE_HIGH_QUALITY_KHR
  - VK_VIDEO_ENCODE_TUNING_MODE_LOW_LATENCY_KHR
  - VK_VIDEO_ENCODE_TUNING_MODE_ULTRA_LOW_LATENCY_KHR
  - VK_VIDEO_ENCODE_TUNING_MODE_LOSSLESS_KHR

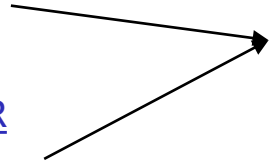
VkVideoEncodeUsageInfoKHR is chained to VkVideoProfileInfoKHR

Video Encode Rate Control

- Rate Control modes
  - Explicit, by disabling the rate control and allowing the application to specify per-operation rate control parameters controlling the encoding quality via the QP value specified in constantQp of the corresponding codec slice/segment/tile
  - Constant bitrate (CBR) rate control
  - Variable bitrate (VBR) rate control
  - Implementation-specific rate control
- Rate Control State

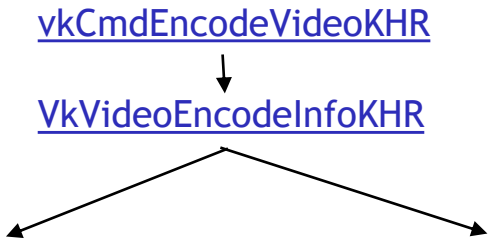
VkVideoBeginCodingInfoKHR
for the video coding scope

VkVideoCodingControlInfoKHR
command against the bound video session



VkVideoEncodeRateControlInfoKHR

Control Layout with Rate Control Disabled



H.265:

VkVideoEncodeH265PictureInfoKHR

```
uint32_t                                naluSliceSegmentEntryCount;
const VkVideoEncodeH265NaluSliceSegmentInfoKHR*
                                        pNaluSliceSegmentEntries;
```

H.265:

VkVideoEncodeH265NaluSliceSegmentInfoKHR

```
int32_t                                 constantQp;
const StdVideoEncodeH265SliceSegmentHeader*
                                        pStdSliceSegmentHeader;
```

H.264:

VkVideoEncodeH264PictureInfoKHR

```
uint32_t                                naluSliceEntryCount;
const VkVideoEncodeH264NaluSliceInfoKHR*
                                        pNaluSliceEntries;
```

H.264:

VkVideoEncodeH264NaluSliceInfoKHR

```
int32_t                                 constantQp;
const StdVideoEncodeH264SliceHeader*
                                        pStdSliceHeader;
```

Control layout case with rate control enabled

vkCmdControlVideoCodingKHR VK_VIDEO_CODING_CONTROL_ENCODE_RATE_CONTROL_BIT_KHR

VkVideoEncodeRateControlInfoKHR:

VkVideoEncodeRateControlFlagsKHR	flags;
VkVideoEncodeRateControlModeFlagBitsKHR	rateControlMode;
uint32_t	layerCount;
const VkVideoEncodeRateControlLayerInfoKHR *	pLayers;
uint32_t	virtualBufferSizeInMs;
uint32_t	initialVirtualBufferSizeInMs;

VkVideoEncodeRateControlLayerInfoKHR

uint64_t	averageBitrate;
uint64_t	maxBitrate;
uint32_t	frameRateNumerator;
uint32_t	frameRateDenominator;

VkVideoEncodeH26XRateControlLayerInfoKHR

VkVideoEncodeH26XQpKHR	minQp;
VkBool32	useMaxQp;
VkVideoEncodeH26XQpKHR	maxQp;
VkBool32	useMaxFrameSize;
VkVideoEncodeH26XFrameSizeKHR	maxFrameSize;

VkVideoEncodeH26XQpKHR

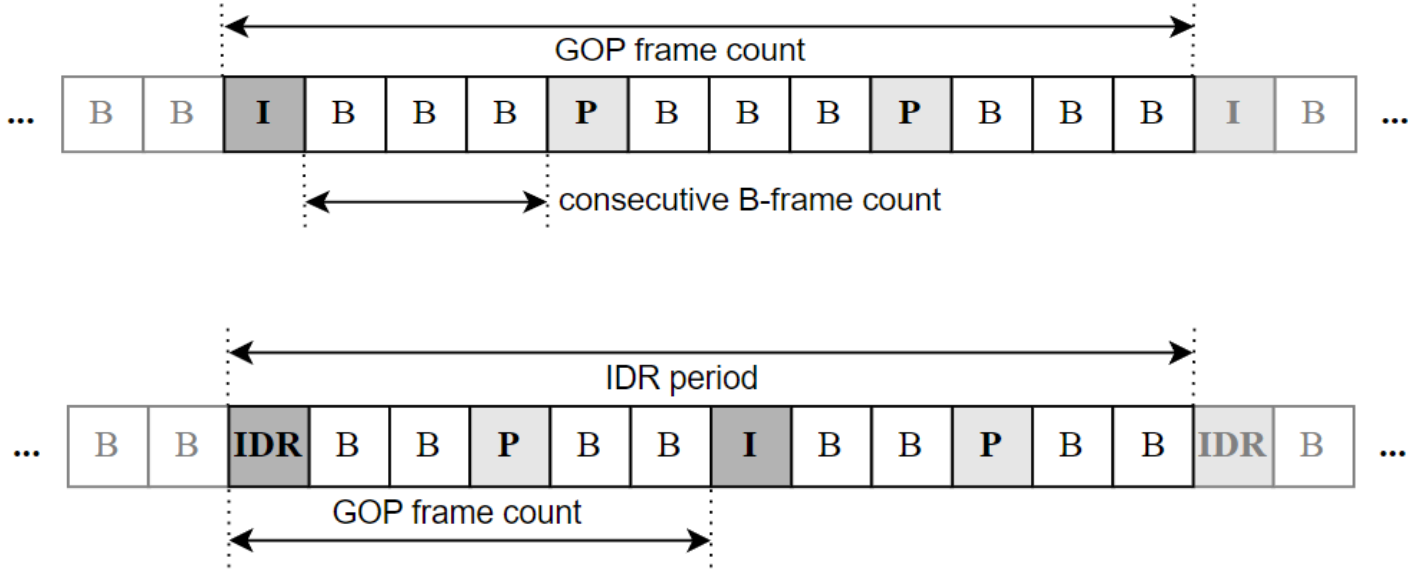
int32_t	qpI;
int32_t	qpP;
int32_t	qpB;

VkVideoEncodeH26XFrameSizeKHR

uint32_t	frameISize;
uint32_t	framePSize;
uint32_t	frameBSize;

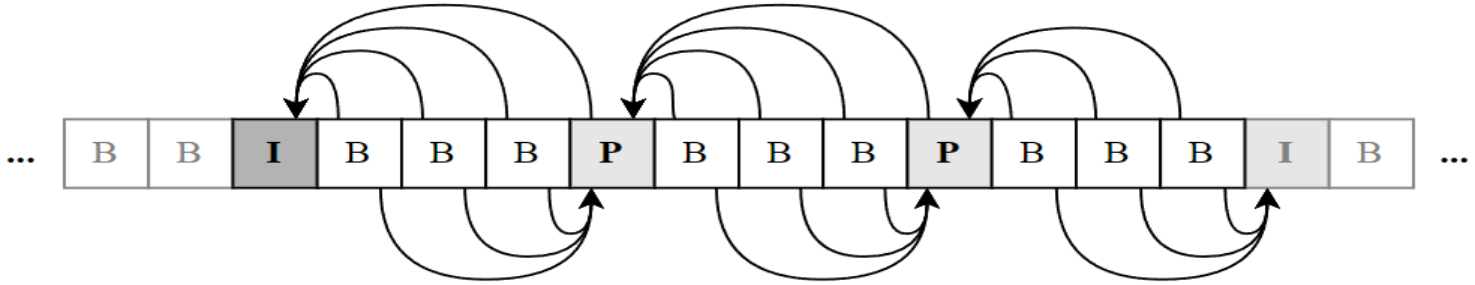
Specifying the GOP structure

`gopFrameCount`
`idrPeriod`
consecutiveBFrameCount
temporalLayerCount



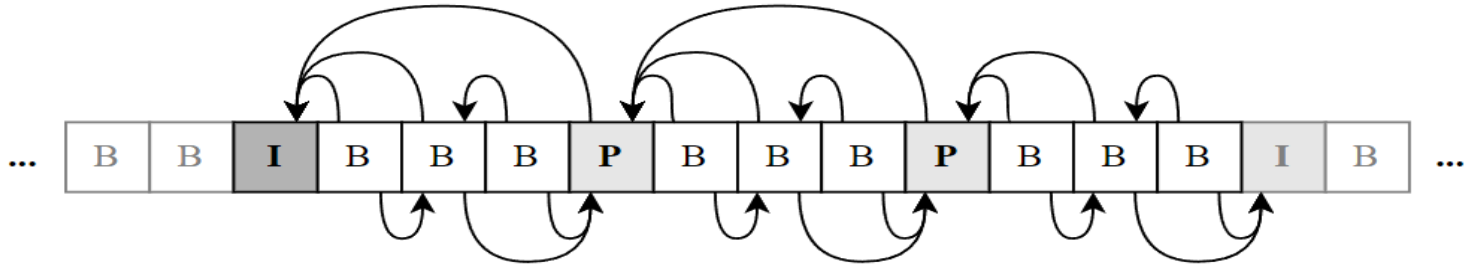
VkVideoEncodeH26XRateControlFlagBitsKHR:VK_VIDEO_ENCODE_H26X_RATE_CONTROL_REGULAR_GOP_BIT_KHR

Vulkan Video API Supported GOP Patterns



Flat Reference Pattern

VkVideoEncodeH26XRateControlFlagBitsKHR:VK_VIDEO_ENCODE_H26X_RATE_CONTROL_REFERENCE_PATTERN_FLAT_BIT_KHR

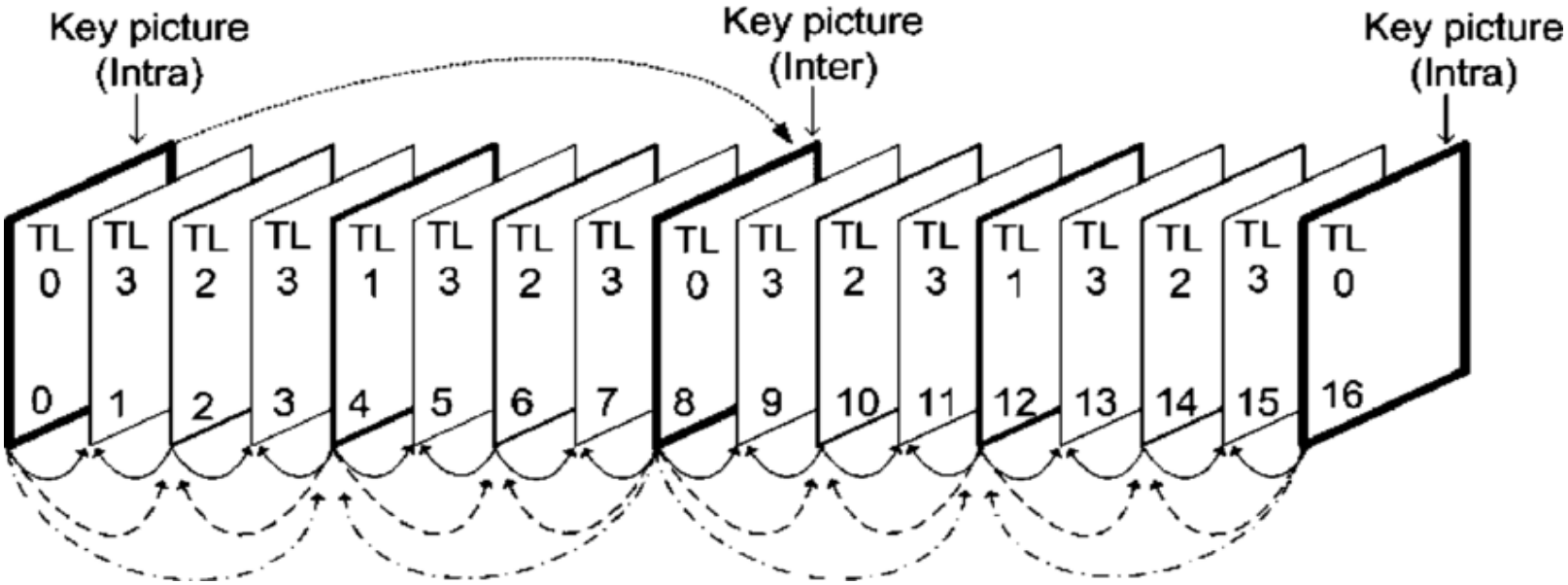


Dyadic Reference Pattern

VkVideoEncodeH26XRateControlFlagBitsKHR:VK_VIDEO_ENCODE_H26X_RATE_CONTROL_REFERENCE_PATTERN_DYADIC_BIT_KHR

VkVideoEncodeH264RateControlInfoKHR::temporalLayerCount = 1; VkVideoEncodeH265RateControlInfoKHR ::subLayerCount = 0

Dyadic Temporal Layer Pattern



VkVideoEncodeH26XRateControlFlagBitsKHR:VK_VIDEO_ENCODE_H26X_RATE_CONTROL_TEMPORAL_SUB_LAYER_PATTERN_DYADIC_BIT_KHR

VkVideoEncodeH264RateControlInfoKHR::temporalLayerCount = 4; VkVideoEncodeH265RateControlInfoKHR ::subLayerCount = 4

Picture Source:
https://www.researchgate.net/figure/Hierarchical-prediction-structure-for-SVC-for-a-GOP-size-of-8_fig1_23676483

Getting Codec-specific Quality Level Properties

vkGetPhysicalDeviceVideoEncodeQualityLevelPropertiesKHR()
VkPhysicalDeviceVideoEncodeQualityLevelInfoKHR * pQualityLevelInfo
VkVideoEncodeQualityLevelPropertiesKHR* pQualityLevelProperties

VkPhysicalDeviceVideoEncodeQualityLevelInfoKHR :
const VkVideoProfileInfoKHR * pVideoProfile;
uint32_t qualityLevel;

VkVideoEncodeQualityLevelPropertiesKHR:
VkVideoEncodeRateControlModeFlagBitsKHR preferredRateControlMode;
uint32_t preferredRateControlLayerCount;

VkVideoEncodeUsageInfoKHR:
VkVideoEncodeUsageFlagBitsKHR videoUsageHints;
VkVideoEncodeContentFlagBitsKHR videoContentHints;
VkVideoEncodeTuningModeKHR tuningMode;

H.264
VkVideoEncodeH264QualityLevelPropertiesKHR
VkVideoEncodeH264RateControlFlagBitsKHR preferredRateControlFlags;
uint32_t preferredGopFrameCount;
uint32_t preferredIdrPeriod;
uint32_t preferredConsecutiveBFrameCount;
uint32_t preferredTemporalLayerCount;
VkVideoEncodeH264QpKHR preferredConstantQp;
uint32_t preferredMaxL0ReferenceCount;
uint32_t preferredMaxL1ReferenceCount;
VkBool32 preferredStdEntropyCodingModeFlag;

H.265
VkVideoEncodeH265QualityLevelPropertiesKHR:
VkVideoEncodeH265RateControlFlagBitsKHR preferredRateControlFlags;
uint32_t preferredGopFrameCount;
uint32_t preferredIdrPeriod;
uint32_t preferredConsecutiveBFrameCount;
uint32_t preferredSubLayerCount;
VkVideoEncodeH265QpKHR preferredConstantQp;
uint32_t preferredMaxL0ReferenceCount;
uint32_t preferredMaxL1ReferenceCount;

Selecting Supported Optimal GOP Patterns

H.264

VkVideoEncodeH264CapabilitiesKHR

```
...
uint32_t     maxPPictureL0ReferenceCount;
uint32_t     maxBPictureL0ReferenceCount;
uint32_t     maxL1ReferenceCount;
uint32_t     maxSubLayerCount;
VkBool32     expectDyadicTemporalSubLayerPattern;
int32_t      minQp;
int32_t      maxQp;
VkBool32     prefersGopRemainingFrames;
VkBool32     requiresGopRemainingFrames;
...
```

VkVideoEncodeH264QualityLevelPropertiesKHR

VkVideoEncodeH264RateControlFlagBitsKHR preferredRateControlFlags;

```
uint32_t                    preferredGopFrameCount;
uint32_t                    preferredIdrPeriod;
uint32_t                    preferredConsecutiveBFrameCount;
uint32_t                    preferredTemporalLayerCount;
VkVideoEncodeH264QpKHR      preferredConstantQp;
uint32_t                    preferredMaxL0ReferenceCount;
uint32_t                    preferredMaxL1ReferenceCount;
....
```

H.265

VkVideoEncodeH265CapabilitiesKHR

```
...
uint32_t     maxPPictureL0ReferenceCount;
uint32_t     maxBPictureL0ReferenceCount;
uint32_t     maxL1ReferenceCount;
uint32_t     maxSubLayerCount;
VkBool32     expectDyadicTemporalSubLayerPattern;
int32_t      minQp;
int32_t      maxQp;
VkBool32     prefersGopRemainingFrames;
VkBool32     requiresGopRemainingFrames;
...
```

VkVideoEncodeH265QualityLevelPropertiesKHR:

VkVideoEncodeH265RateControlFlagBitsKHR preferredRateControlFlags;

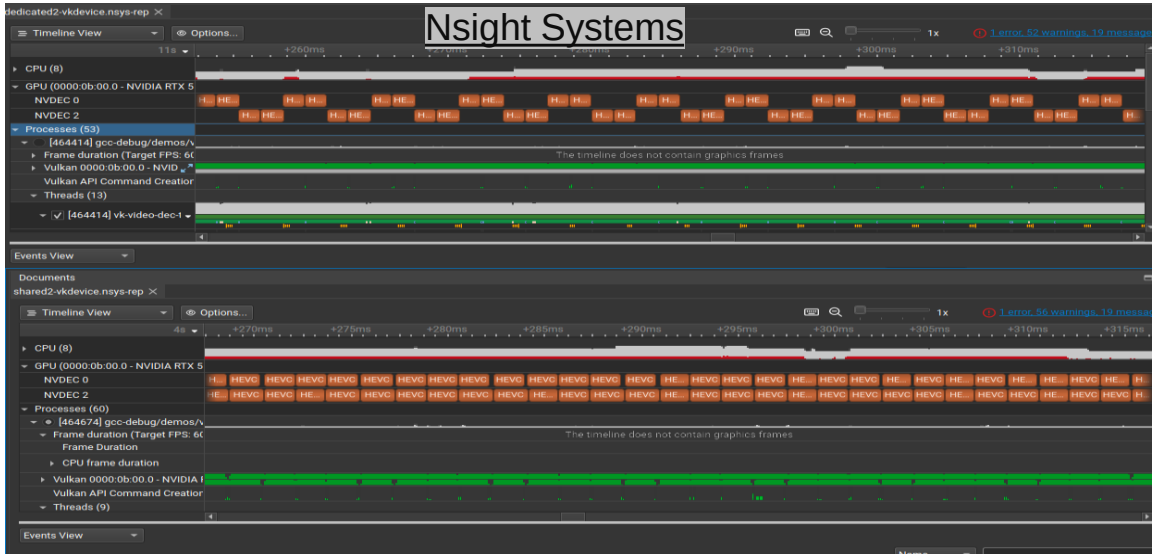
```
uint32_t                    preferredGopFrameCount;
uint32_t                    preferredIdrPeriod;
uint32_t                    preferredConsecutiveBFrameCount;
uint32_t                    preferredSubLayerCount;
VkVideoEncodeH265QpKHR      preferredConstantQp;
uint32_t                    preferredMaxL0ReferenceCount;
uint32_t                    preferredMaxL1ReferenceCount;
...
```

Development tools

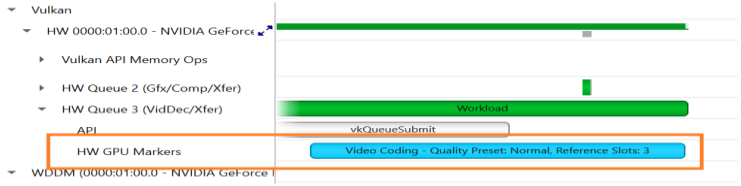
Vulkan Video Frame Timing Using NVIDIA Nsight Graphics and Systems profiler

Check out Optimizing Vulkan 1.3 Applications with Nsight Graphics and Nsight Systems and Nsight Systems - Vulkan Trace.

NVIDIA Nsight Developer Tools are a collection of debuggers, profilers, and optimizers that support performance tuning for applications using many graphics APIs, including Vulkan.



The NVIDIA Nsight Systems profiler will enable parsing of the specific workload of a Vulkan Video decoding queue, providing insights into processing bottlenecks within the context of the application.



Vulkan Video Conclusions

- A robust cross-platform API for video acceleration
- Extensive flexibility for creating scalable, high-performance, and highly customizable video applications
- Benefits support from open-source community and leading Independent Hardware Vendors (IHVs)

Reference Materials

- An Introduction to Vulkan Video
- A deep dive into Vulkan Video Presentation and Slides Vulkan Specification, including Vulkan video extensions
- Vulkan Headers
- Validation Layer PR
- Vulkan SDK including Vulkan video support and validation layers
- Vulkan video samples
- Optimizing Vulkan 1.3 Applications with Nsight Graphics and Nsight Systems and Nsight Systems - Vulkan Trace.
- NVIDIA beta Vulkan drivers

The Conference for the Era of AI and the Metaverse

NVIDIA GTC is where developers, researchers, students, creators, IT decision-makers, and business leaders gather to learn about the latest breakthroughs shaping our world.

We explore what's driving transformation in everything from collaborative virtual worlds and advanced graphics to data science, healthcare, and beyond.

Come experience GTC for groundbreaking content, expert-led sessions, and a must-see keynote to accelerate your life's work.

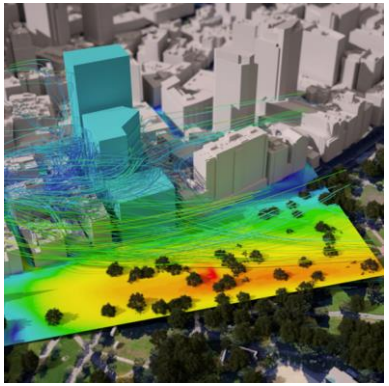
March 17-21 | http://www.nvidia.com/gtc/

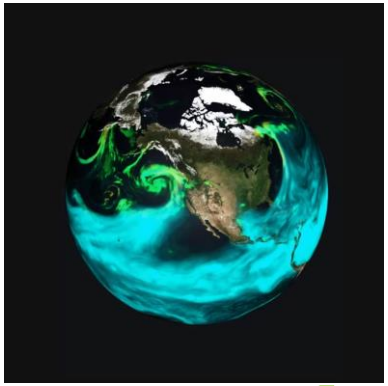
Recommended Sessions and Training

Watch Party: Accelerating Ray Tracing and AI in Unreal Engine [WP51851]

Connect with the Experts: Using NVIDIA Developer Tools to Optimize Ray Tracing [CWES52009]

Ray-Tracing Development using NVIDIA Nsight Graphics and NVIDIA Nsight Systems* [DLIT51580]





Join us at GTC

The Conference for the Era of AI and the Metaverse

NVIDIA Developer Program

The Community that Builds

Program Benefits:

Tools

- 550+ exclusive SDKs and models
- GPU-optimized software, model scripts, and containerized apps
- Early access programs

Training

- Research papers, technical documentation, webinars, blogs, and news
- Technical training and certification opportunities
- 1,000s of technical sessions from industry events On-Demand

Community

- NVIDIA developer forums
- Exclusive meetups, hackathons, and events

Join the Community

NVIDIA

Q & A

Thank you!

Questions?