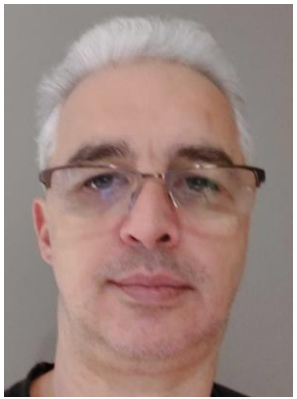
Vulkan Video Core APIs

Presented at the Khronos Vulkanised 2023 Conference

Tony Zlatinski, NVIDIA
tzlatinski@nvidia.com



This work is licensed under a Creative Commons Attribution 4.0 International License

Presentation outline

- General Vulkan Video concept and architecture
- Vulkan Video from HW perspective
- Vulkan Core and Video add-on objects
- Vulkan Video execution model
- Vulkan queue interactions and synchronization
- Application-specific usecases of Vulkan Video

KHRONOS GROUP

This work is licensed under a Creative Commons Attribution 4.0 International License

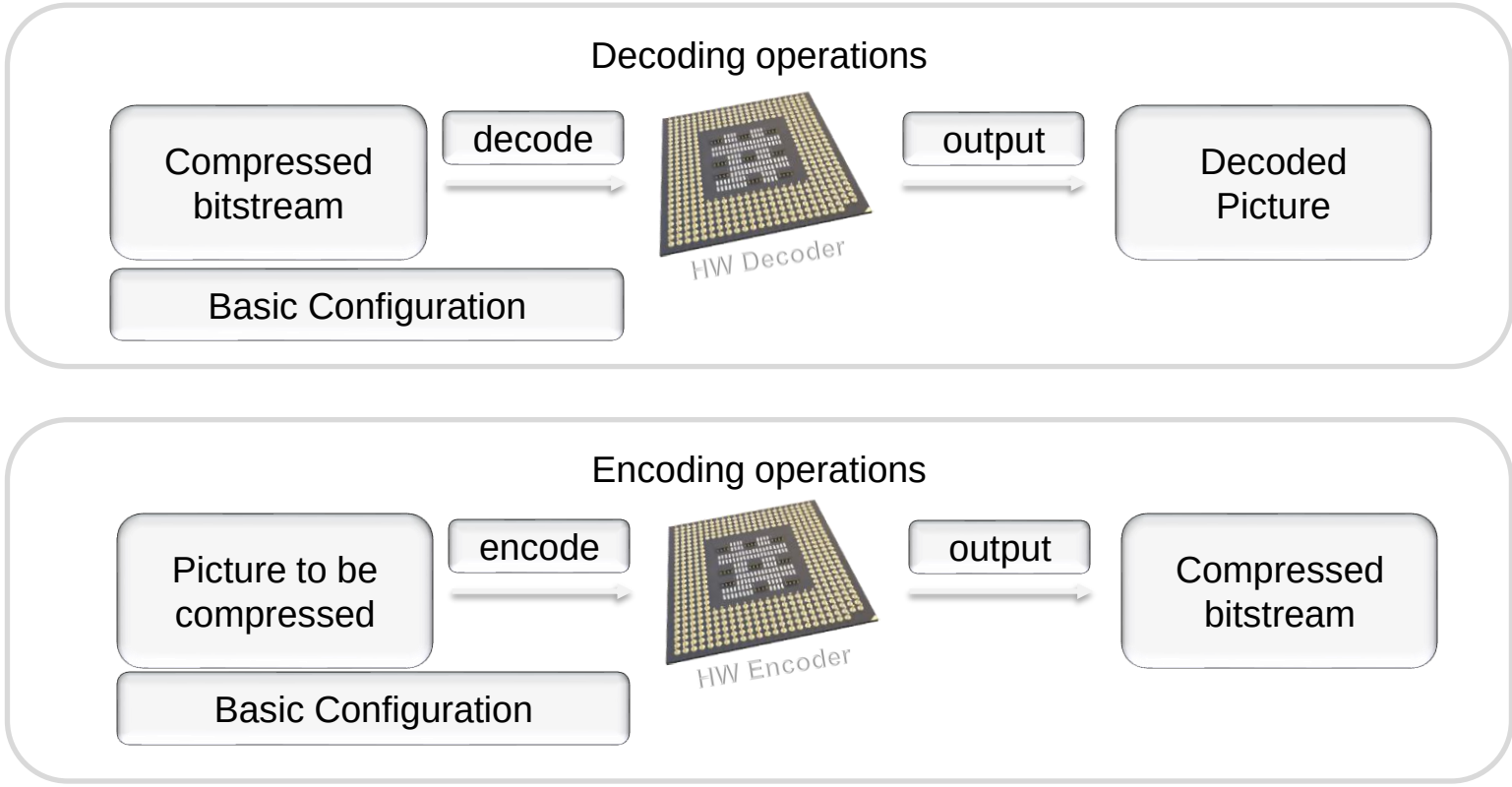
Vulkan Video API design goals

- Cross-platform and vendor-neutral low-level HW state(-less) video codecs API
- Closer integration with the Graphics and Displays
- Lower-level synchronization allowing for lower processing latency
- Enables scalable applications targeting high-throughput servers to low-power, limited-resource embedded devices.
  - A better solution for parallel video stream processing with ability to distribute work among multiple CPU and HW codec cores
  - Lower processing execution overhead
  - Lower CPU/GPU/ASIC and memory resource utilization

KHRONOS GROUP

This work is licensed under a Creative Commons Attribution 4.0 International License

Simplified Video Decoding / Encoding diagrams



This work is licensed under a Creative Commons Attribution 4.0 International License

Vulkan Video Support

NVIDIA, Intel & AMD are the first IHVs to implement support for video extensions:

- NVIDIA: Windows and Linux BETA drivers
- Intel: Coming soon
- AMD: Windows BETA driver

The official example application for Vulkan Video Decode extensions is the open-source vk_video_decode sample from NVIDIA.

The very popular gstreamer (developed by Igalia) and ffmpeg (developed by Lynne) video processing frameworks and RADV and ANV open-source drivers are early adopters of Vulkan Video.

This work is licensed under a Creative Commons Attribution 4.0 International License

IHV and open-source short-term AVC (h.264) and HEVC (h.265) implementation availability

Intel will support Windows OS for Intel® Arc™ A-Series Graphics and Intel® Iris® Xe Graphics.

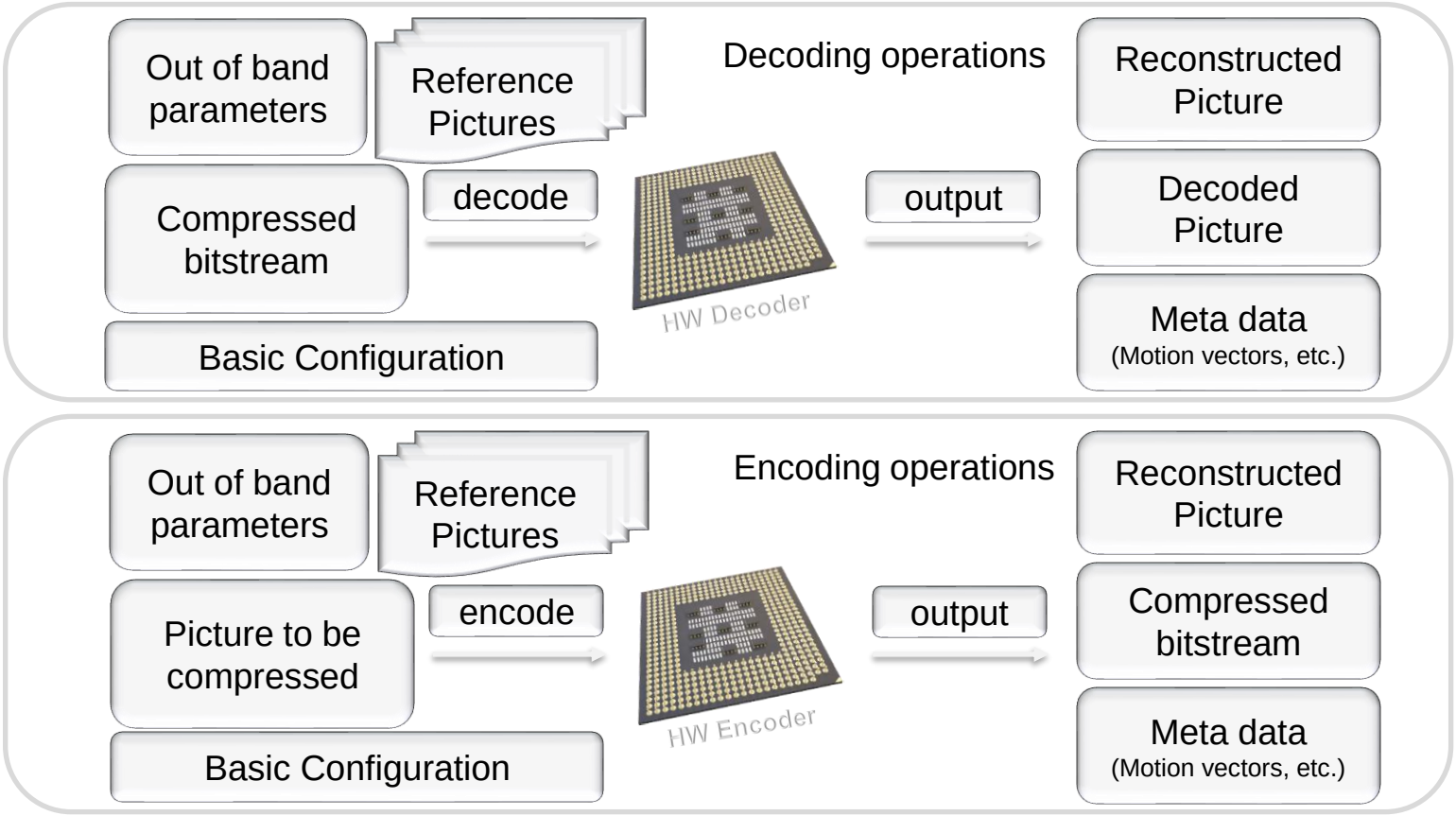
- **11th Gen Intel® Core™** processor family (Codename Tiger Lake, Rocket Lake)
- **12th Gen Intel® Core™** processor family (Codename Alder Lake)
- **13th Gen Intel® Core™** processor family (Codename Raptor Lake)
- **Intel® Arc® Graphics** family (Codename Alchemist)
- The **ANV** open-source driver, developed by **David Airlie**, will support h264 decode from Skylake to the Intel Gen13/Arc Graphics. Possibly, h.265 decode, as well.

AMD will support Vulkan Video decode on all RDNA™ architecture-based graphics in AMD Software: Adrenalin Edition™ drivers for Windows and Linux to be released later this year. Windows Beta driver is available now for **RX 5000 series and RX 6000 series GPUs**. The **RADV** open-source driver, developed by **David Airlie,** will support decode on AMD nav10 and above, and possibility of support back to Polaris.

NVIDIA will make Vulkan video decode available starting with **Maxwell and later** (the latest one being **ADA**) **GPUs** for Linux and Windows OS. Windows and Linux BETA drivers are currently available.

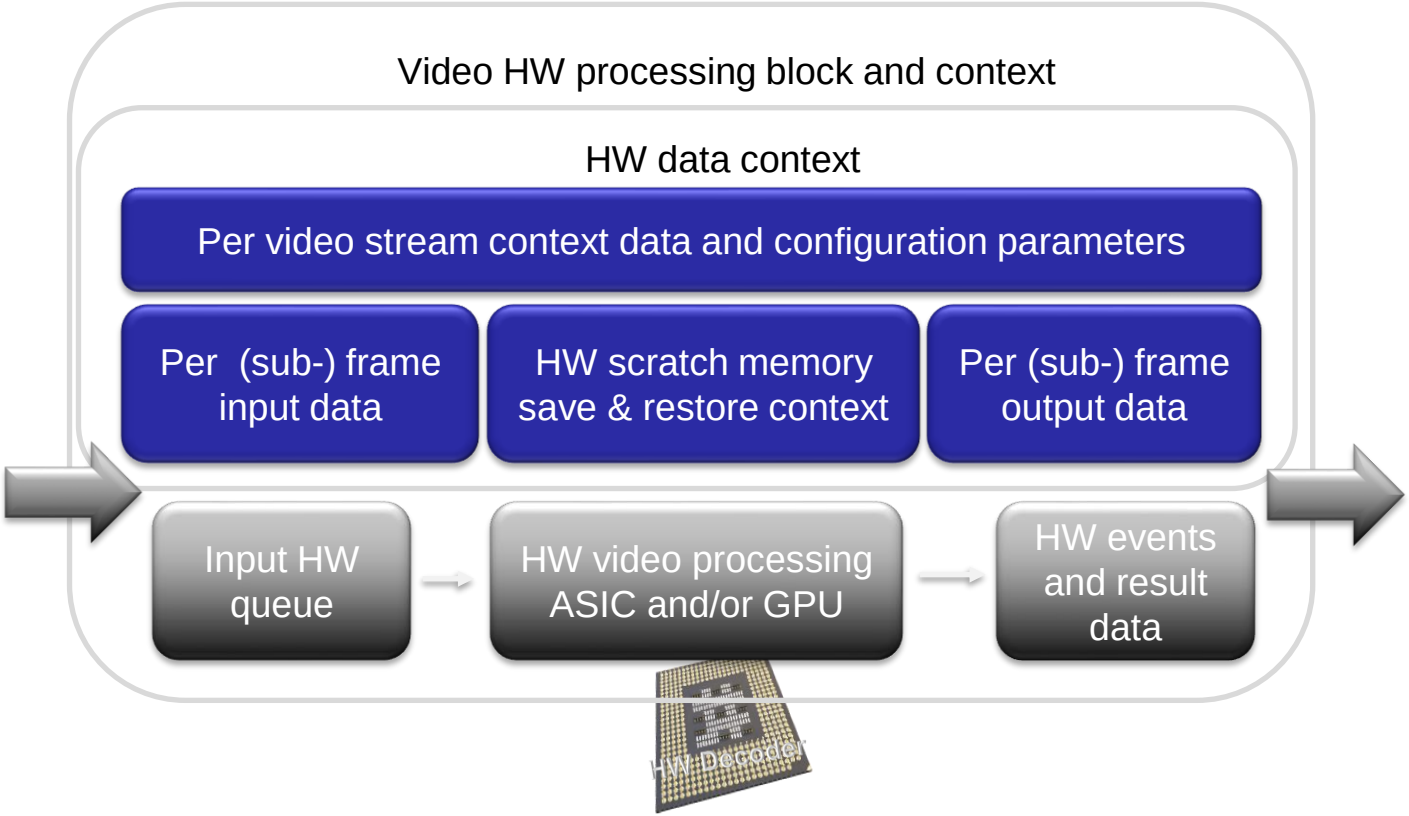
This work is licensed under a Creative Commons Attribution 4.0 International License

Contemporary Video Decoding / Encoding



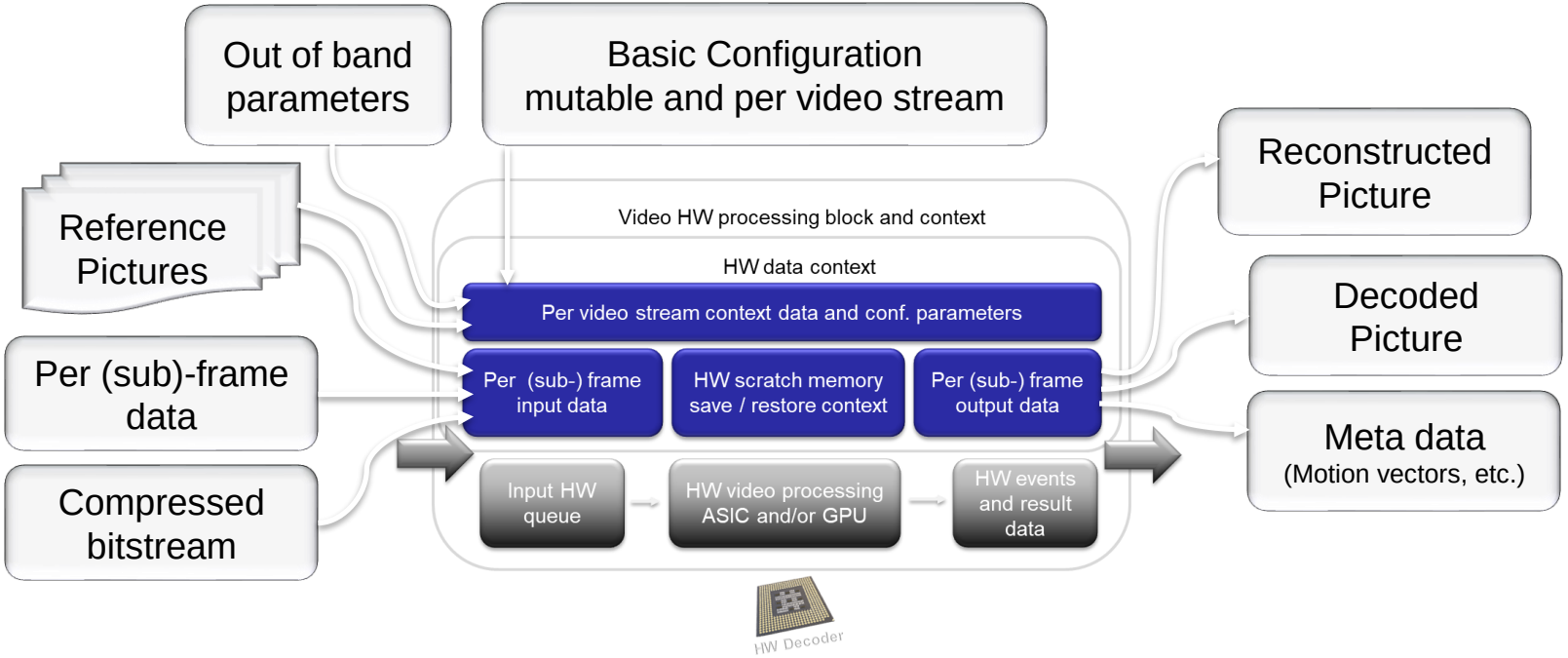
This work is licensed under a Creative Commons Attribution 4.0 International License

Video decoder (or encoder) hardware viewpoint



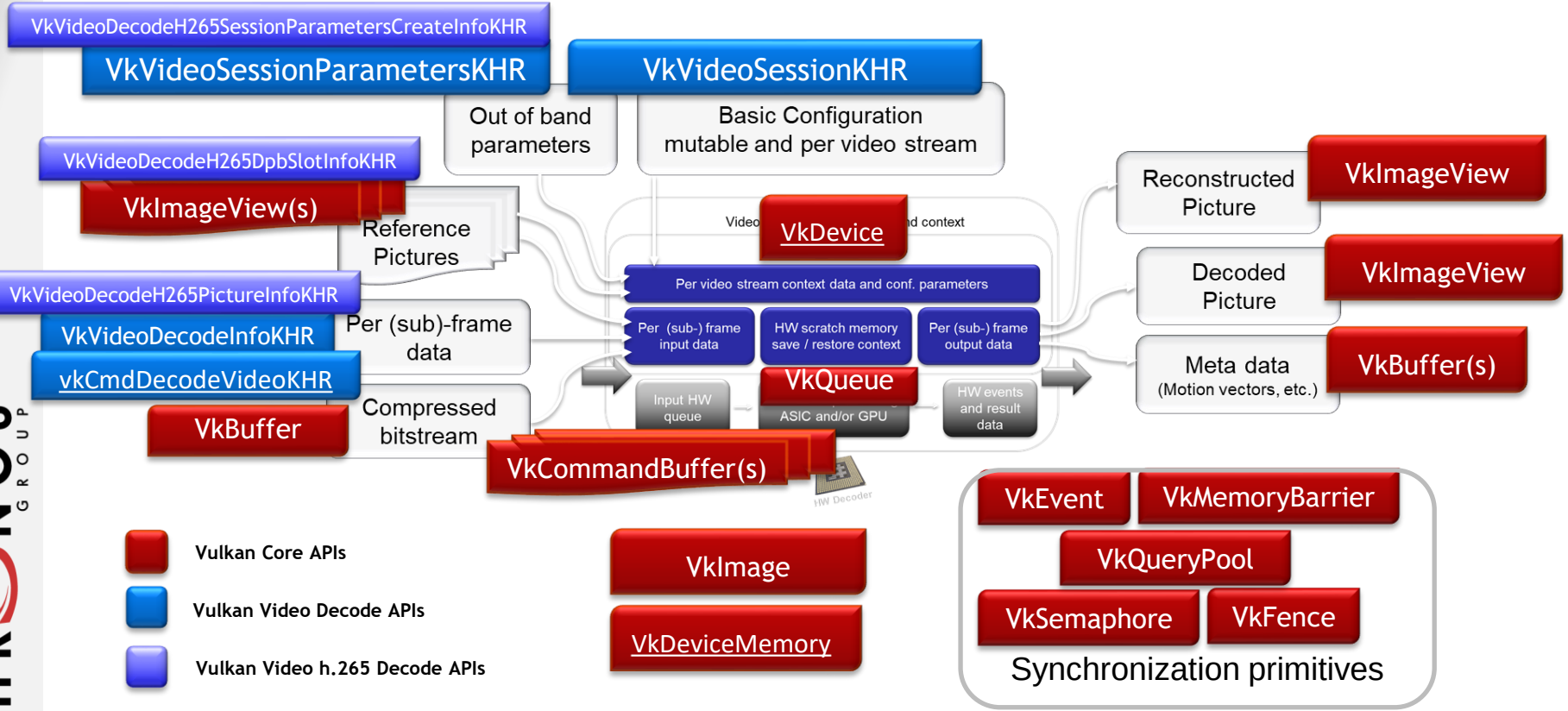
This work is licensed under a Creative Commons Attribution 4.0 International License

Video decoder with hardware viewpoint



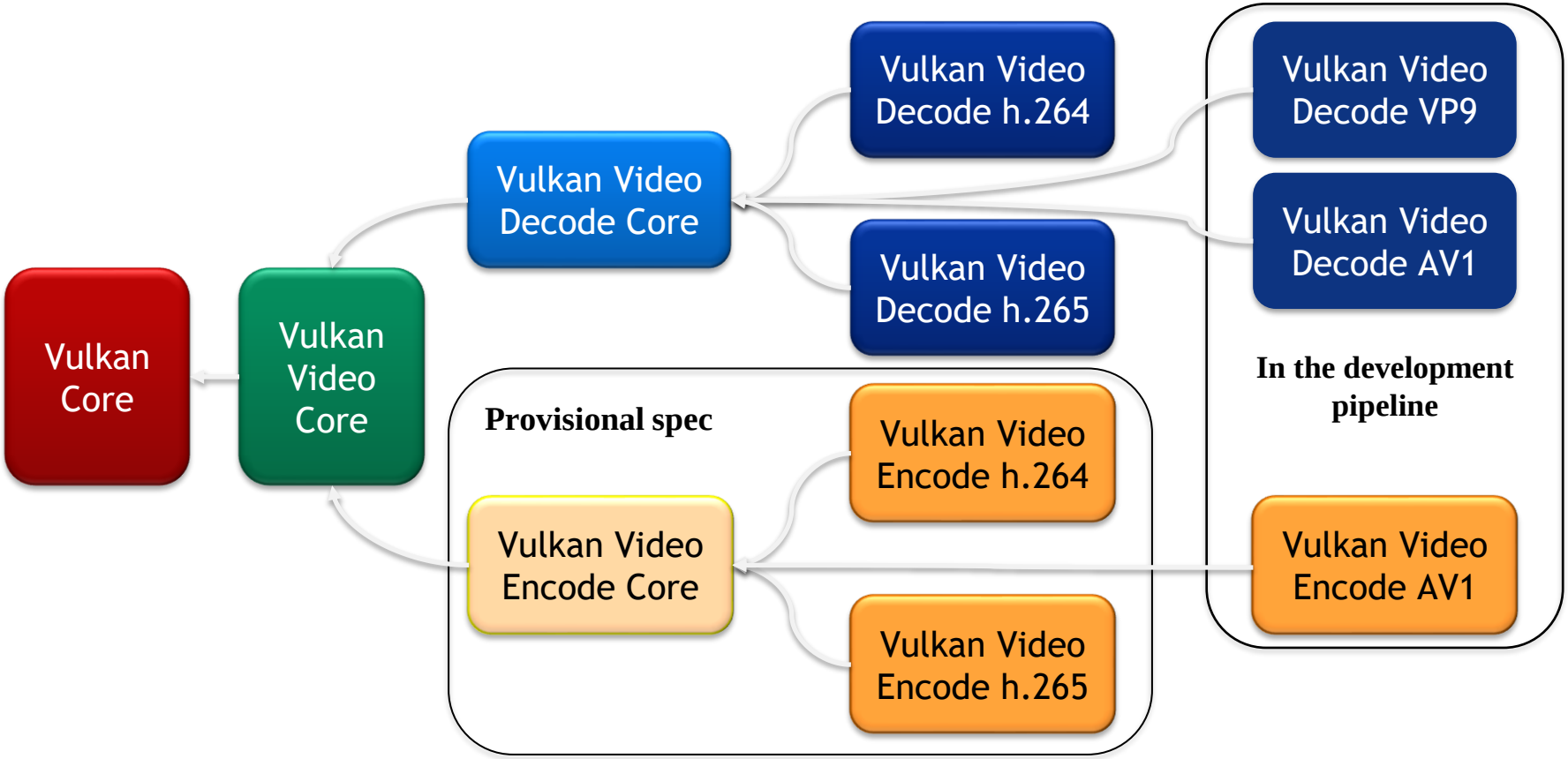
This work is licensed under a Creative Commons Attribution 4.0 International License

Vulkan Video decoder with hardware viewpoint



This work is licensed under a Creative Commons Attribution 4.0 International License

Vulkan Video Core and Codec Extensions



This work is licensed under a Creative Commons Attribution 4.0 International License

Vulkan queue family types and device queues

VkQueueFamilyProperties::queueFlags

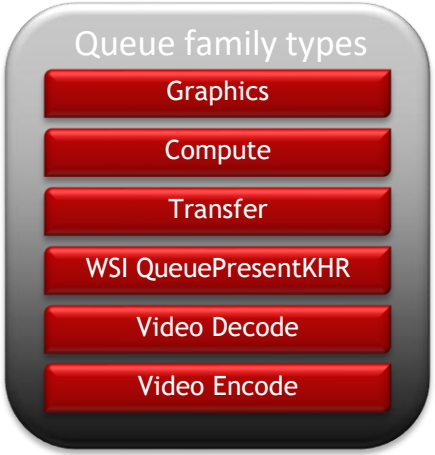
VK_QUEUE_GRAPHICS_BIT

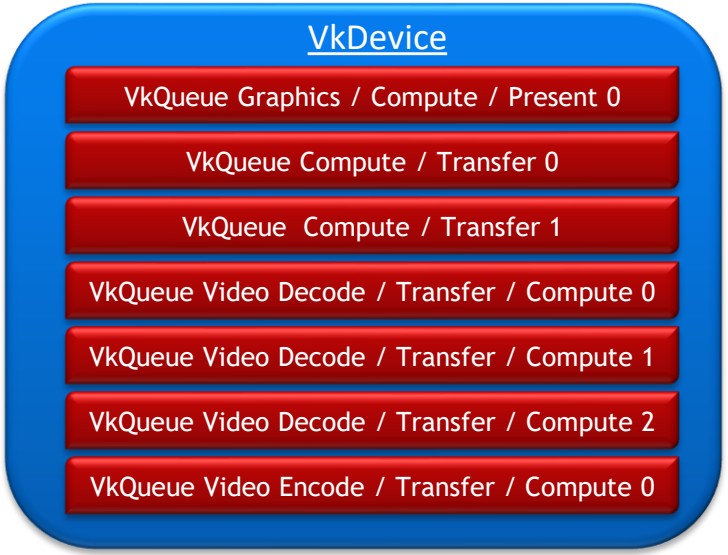
VK_QUEUE_COMPUTE_BIT

VK_QUEUE_TRANSFER_BIT

VK_QUEUE_VIDEO_DECODE_BIT_KHR

VK_QUEUE_VIDEO_ENCODE_BIT_KHR





This work is licensed under a Creative Commons Attribution 4.0 International License

How many instances of video HW blocks are available?

Use the vkGetPhysicalDeviceQueueFamilyProperties2() to query how many video HW instances does the implementation support.

Which queue family supports video?

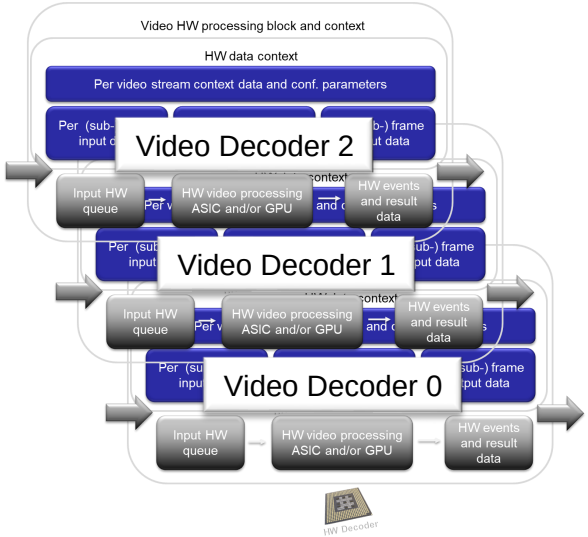
VkQueueFamilyProperties2::**queueFamilyProperties.queueFlags contains** ***VK_QUEUE_VIDEO_DECODE_BIT_KHR for decode or VK_QUEUE_VIDEO_ENCODE_BIT_KHR for encode.***

How many HW block instances are supported?

VkQueueFamilyProperties2::**queueFamilyProperties.queueCount**

When creating the device, specify the number of HW blocks required:

VkDeviceQueueCreateInfo::queueCount = numOfHwBlocks**Required**



This work is licensed under a Creative Commons Attribution 4.0 International License

Video Session Object

- **The VkVideoSessionKHR object stores (read-only) video configuration settings and the context associated with for a single video stream**
  - One session object per video stream
- **Required for any video decode or encode operations**
  - Specifies the video profile and maximum parameters for the video stream
- **A video session instance supports a single compression standard only**
  - H.264, HEVC, VP9, AV1, etc.
- **Video Session object maintains the device memory heaps**
  - The application allocates and binds VkDeviceMemory objects to the Video Session object which uses it for its memory heaps

KHRONOS GROUP

This work is licensed under a Creative Commons Attribution 4.0 International License

Vulkan Video Profiles

- The VkVideoProfileInfoKHR and VkVideoProfileListInfoKHR structures are containers of formats describing the compressed bitstream. Specifically, those structures describe:
  - videoCodecOperation – codec operations such as h264 encode, h265 encode, etc.
  - chromaSubsampling – YCBCr 4:4:4, 4:2:2, 4:2:0, 4:0:0 color subsampling mode
  - lumaBitDepth and chromaBitDepth describe luminance & chroma channel bit depth – 8,10,12-bit
  - videoUsageHints – transcoding, offline, streaming, etc.
- **Video Profile structures must be included** when obtaining device properties or creating Vulkan objects that will be used with Vulkan Video. A single profile instance of VkVideoProfileInfoKHR is also required as part of VkVideoSessionCreateInfoKHR when creating a video session with vkCreateVideoSessionKHR().
- The VkVideoProfileListInfoKHR structure contains multiple profiles and is usually used for matching the decoder's output with the encoder's inputs when requiring video transcoding.

Vulkan API	Vulkan Structure to Extend
vkGetPhysicalDeviceFormatProperties2	VkFormatProperties2
vkCreateImage	VkImageCreateInfo
vkCreateImageView	VkImageViewCreateInfo
vkCreateBuffer	VkBufferCreateInfo
vkCreateQueryPool	VkQueryPoolCreateInfo

This work is licensed under a Creative Commons Attribution 4.0 International License

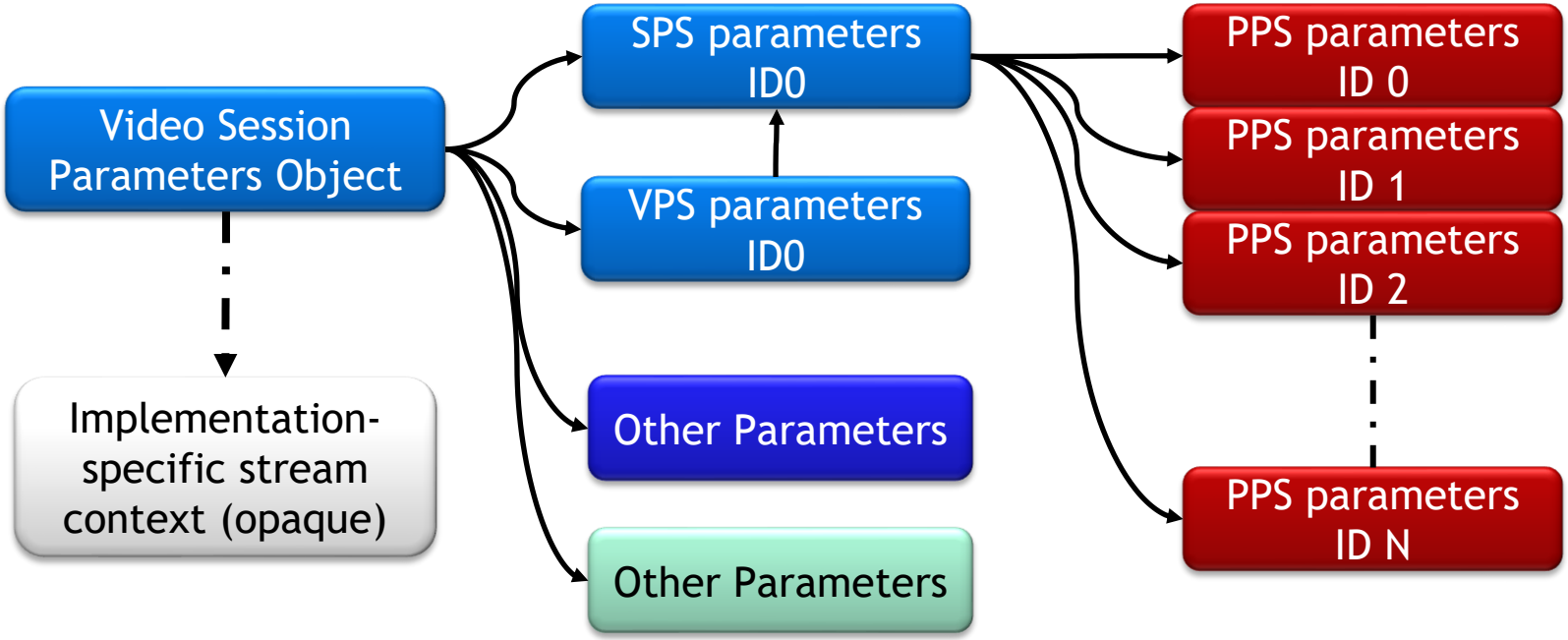
Video Session Parameters Object

- The VkVideoSessionParametersKHR **object contains processing parameters**
  - It must be created against and belongs to a Video Session object.
- **Multiple video session parameters objects can be used to process a video stream**
  - A video session parameters object can apply to the entire video stream, or only a part of it.
  - Session Parameter object is provided with the vkCmdBeginVideoCodingKHR command and remains in effect until the next vkCmdEndVideoCodingKHR command
- **The API can add parameters to a video session parameters object**
  - Previously added parameters with a specific ID cannot be modified
  - The API can clone all video parameters into a new Session Parameter object

KHRONOS GROUP

This work is licensed under a Creative Commons Attribution 4.0 International License

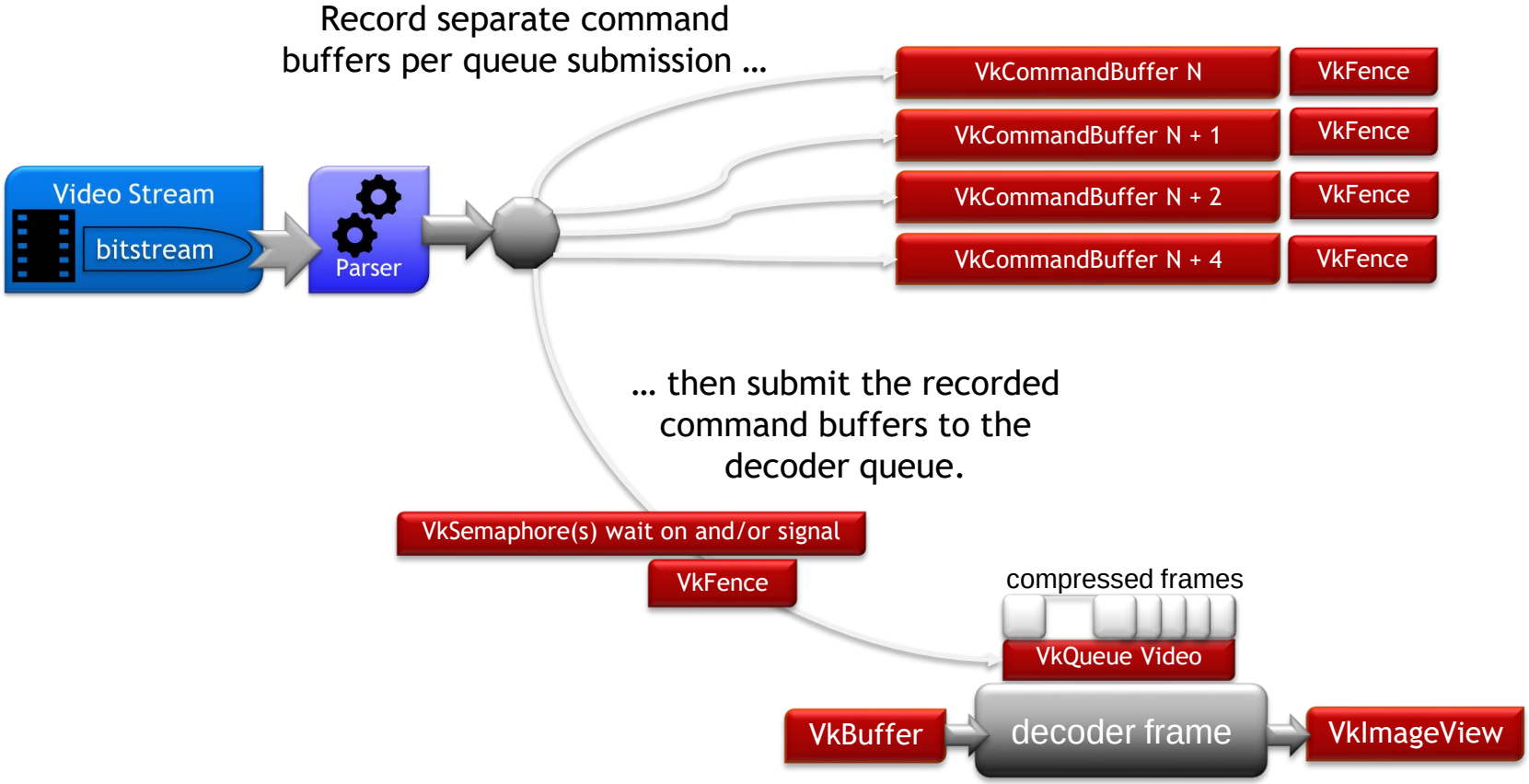
An example of a Set of HEVC Codec Parameters



[PPS/SPS/VPS/VUI parameters in h.265]

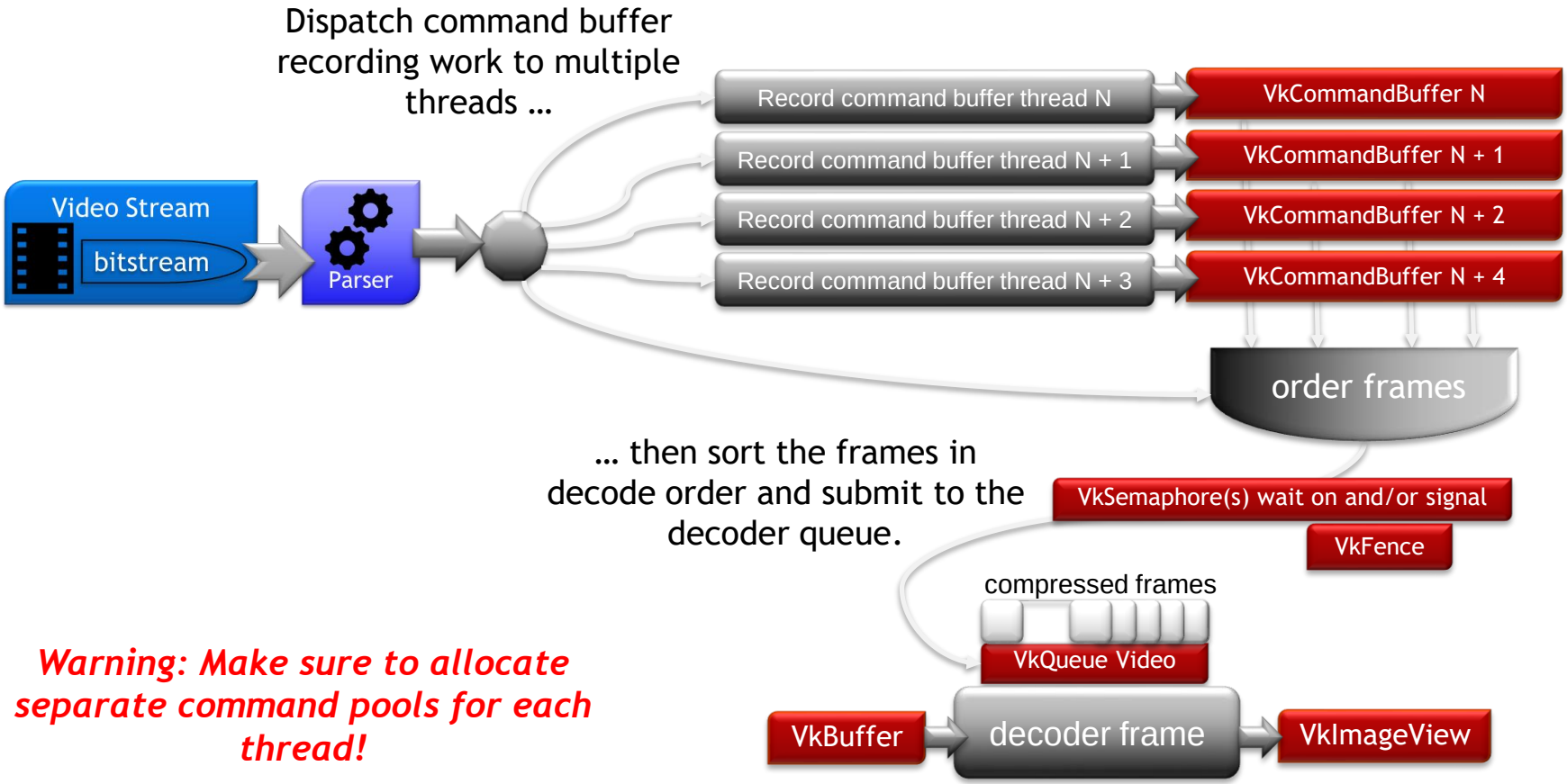
This work is licensed under a Creative Commons Attribution 4.0 International License

Recording and submitting the command buffers



This work is licensed under a Creative Commons Attribution 4.0 International License

Utilizing the host CPU multi-processing power



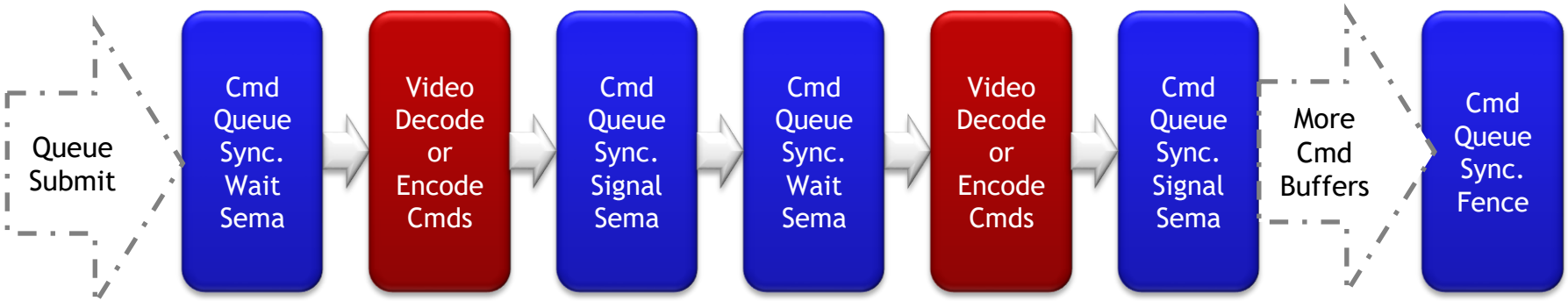
Warning: Make sure to allocate separate command pools for each thread!

This work is licensed under a Creative Commons Attribution 4.0 International License

Video VkCommandBuffers Queue Submission

Regular Vulkan Queue Submit Sequence:

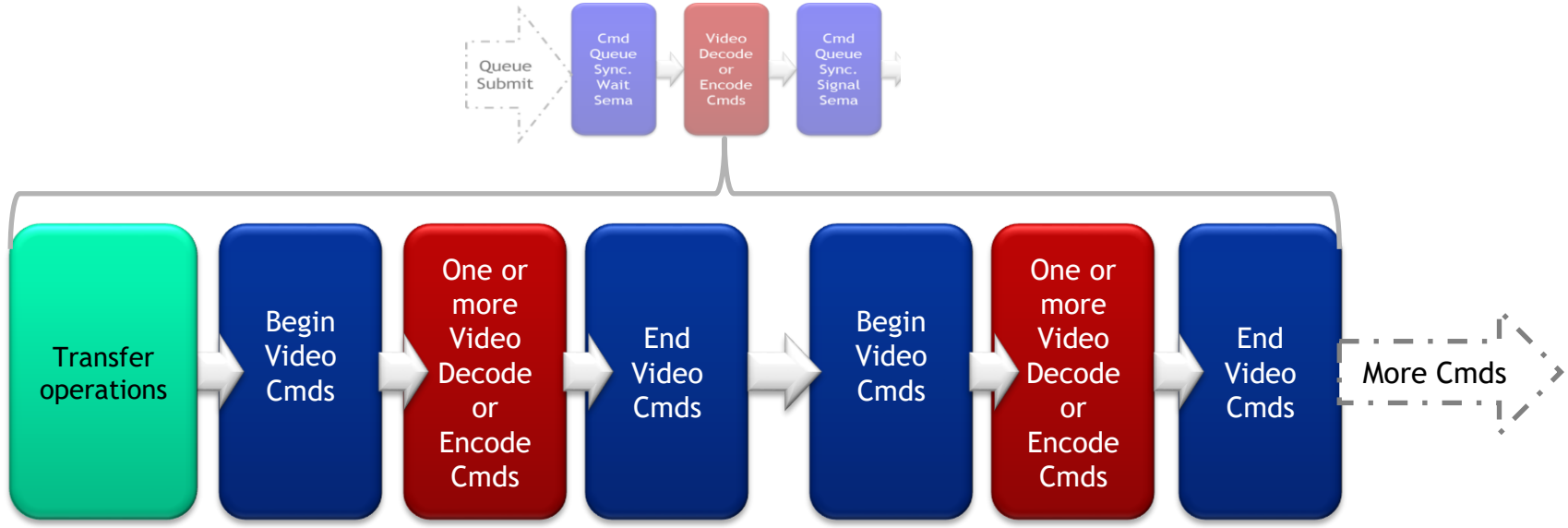
- One or more Recorded Vulkan Video Command Buffers can be submitted
- The command buffer sequences can be synchronized by binary or timeline semaphores
- The command buffer sequences can be synchronized with the host CPU or between the Vulkan queues via semaphores or a fence



This work is licensed under a Creative Commons Attribution 4.0 International License

VkCommandBuffers Video Recording Sequence

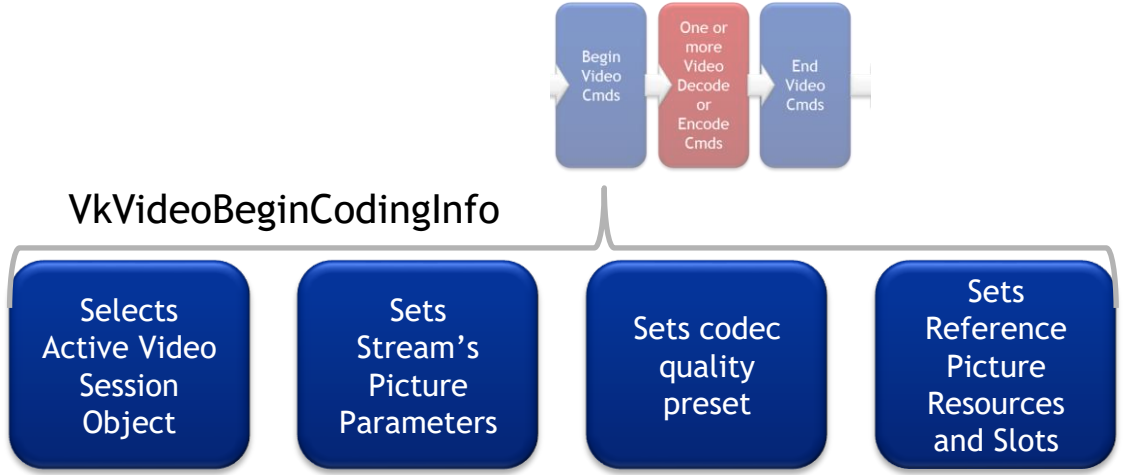
- All Vulkan Video command sequences start with vkCmdBeginVideoCodingKHR and end with vkCmdEndVideoCodingKHR
- Multiple Video Start/End command sequences are supported
- Implicit ordering guarantees also apply to the video and other commands that belong to the same command buffer



This work is licensed under a Creative Commons Attribution 4.0 International License

VkCommandBuffers Recording Context Setup

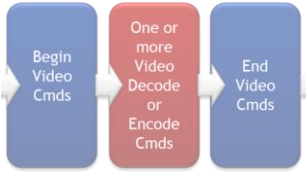
- vkCmdBeginVideoCodingKHR via the VkVideoBeginCodingInfoKHR parameters establishes a context for the subsequent video decode and/or encode commands
- vkCmdEndVideoCodingKHR terminates the context established by the last vkCmdBeginVideoCodingKHR



This work is licensed under a Creative Commons Attribution 4.0 International License

Recording VkCommandBuffer Commands

Only commands for decode/encode, barriers/events/query and transfer operation are allowed between the vkCmdBeginVideoCodingKHR and vkCmdEndVideoCodingKHR commands.



Handle Global Memory Barriers
VkMemoryBarrier

Handle Image Layouts Transition

Handle buffer barriers for stream buffers with VkBufferMemoryBarrier

Handle image barriers for input, output and DPB images and buffers with VkImageMemoryBarrier

vkCmdPipelineBarrier

Set frame events
vkCmdWaitEvents()
vkCmdSetEvent()
vkCmdResetEvent()

Set per frame or slice queries*
vkCmdResetQueryPool()
vkCmdBeginQuery()
vkCmdEndQuery()

One or more Video Decode operations
vkCmdDecodeVideoKHR

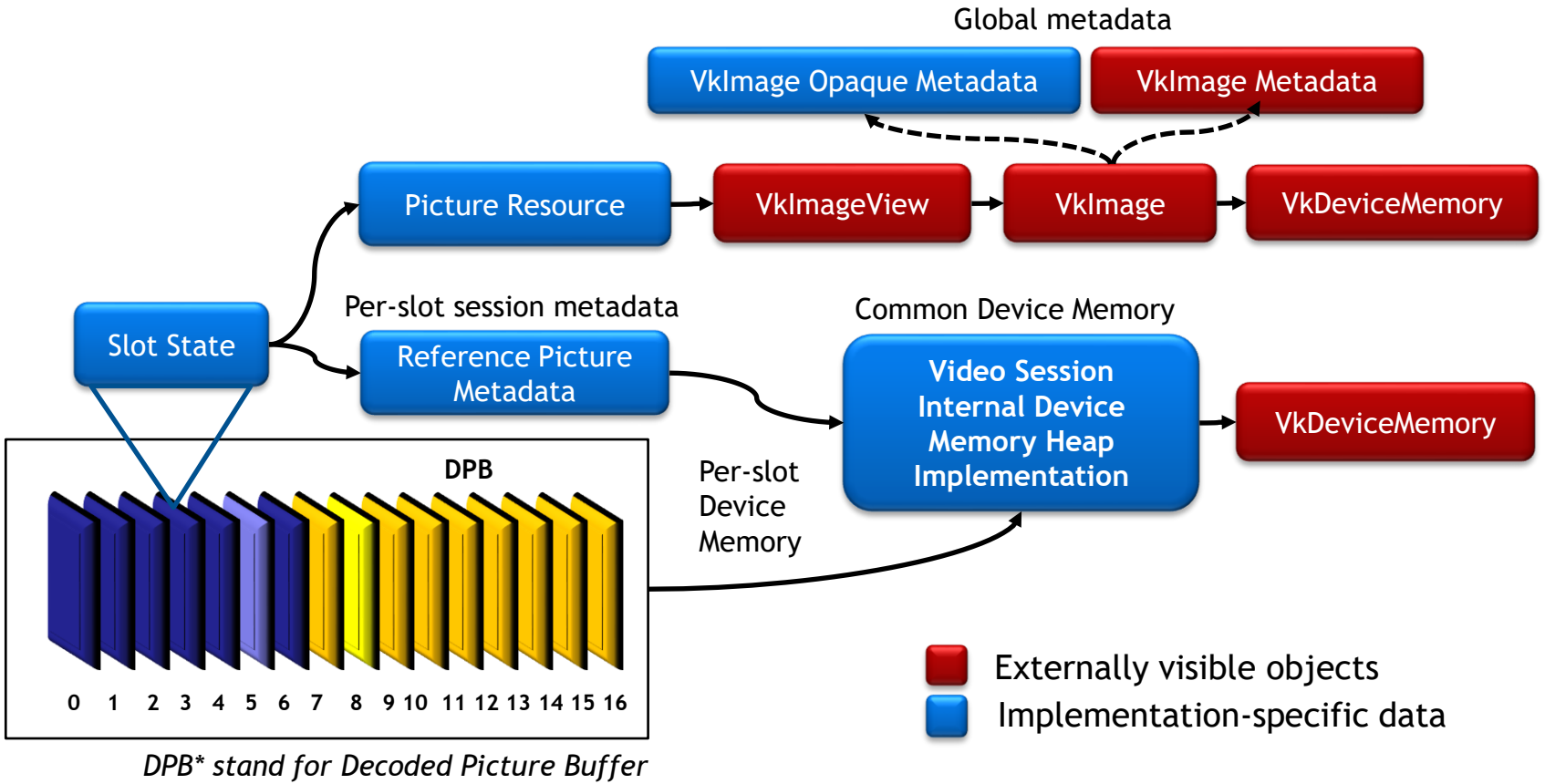
One or more Video Encode operations
vkCmdEncodeVideoKHR

Transfer Operations**:
vkCmdCopyBuffer
vkCmdCopyImage
vkCmdCopyBufferToImage
vkCmdCopyImageToBuffer

** Some implementations may NOT support video queries.*
*** Transfer operation are only allowed if the video decode and/or encode queue advertises transfer operations via the VK_QUEUE_TRANSFER_BIT*

This work is licensed under a Creative Commons Attribution 4.0 International License

Video Decode/Encode DPB Picture Resources



DPB stand for Decoded Picture Buffer*

This work is licensed under a Creative Commons Attribution 4.0 International License

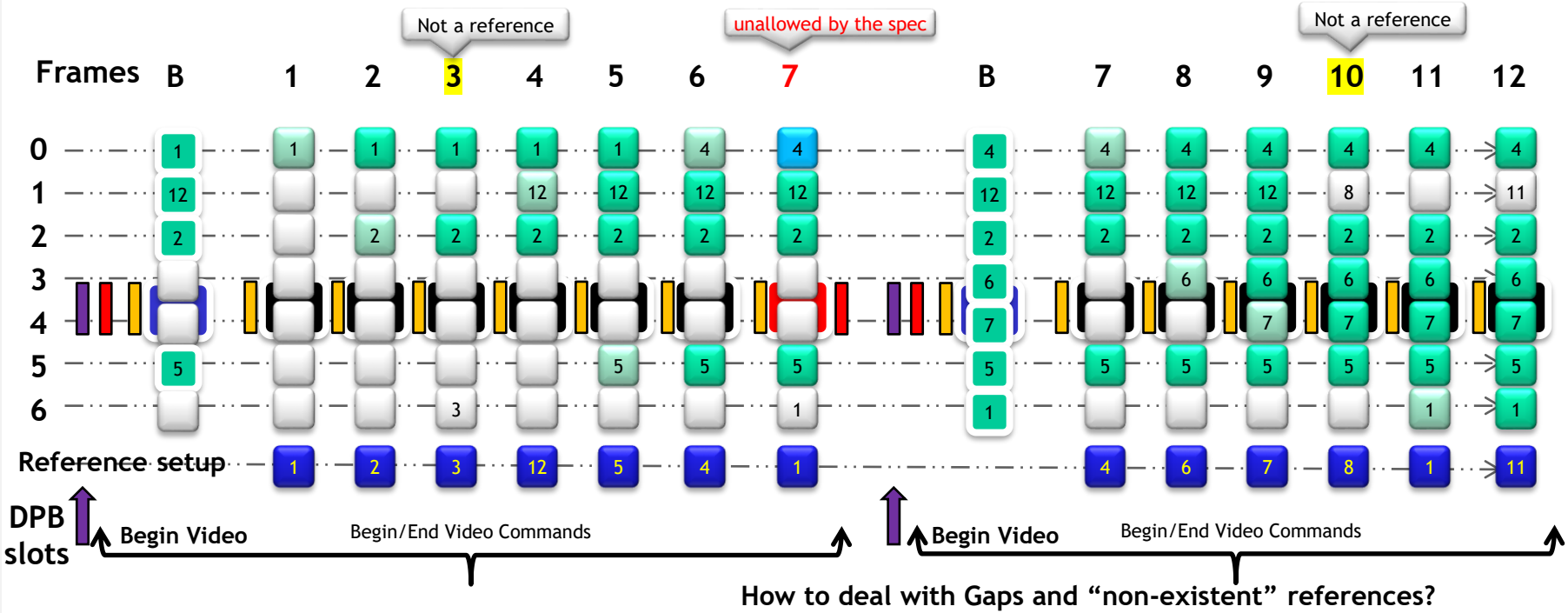
DPB Slot Management

- **Allocating/Associating DPB reference slots with *slotId***
  - Add entry with slotId and associated VkImageView resource in the array of VkVideoBeginCodingInfoKHR::pReferenceSlots within the vkCmdBeginVideoCodingKHR command
- **Making DPB reference slot with slotId valid**
  - Decode (vkCmdDecodeVideoKHR) or Encode (vkCmdEncodeVideoKHR) commands targeting slotId within the VkVideoDecodeInfoKHR::pSetupReferenceSlot
- **Invalidating DPB slot with slotId**
  - Replace association of the reference slot with slotId with a different VkImageView resource
  - Decode or Encode commands targeting the reference slot with slotId (with VkVideoDecodeInfoKHR:: pSetupReferenceSlot)
  - Reset the decoder/encoder session via vkCmdControlVideoCodingKHR with `VK_VIDEO_CODING_CONTROL_RESET_BIT_KHR flag set.`
  - Replace the content of the associated VkImageView resource or unbind the backing memory
  - Change the layout of the associated VkImageView resource to an incompatible to DPB layout

This work is licensed under a Creative Commons Attribution 4.0 International License

DPB Slot Management Example

Allocating slots with associated picture resources



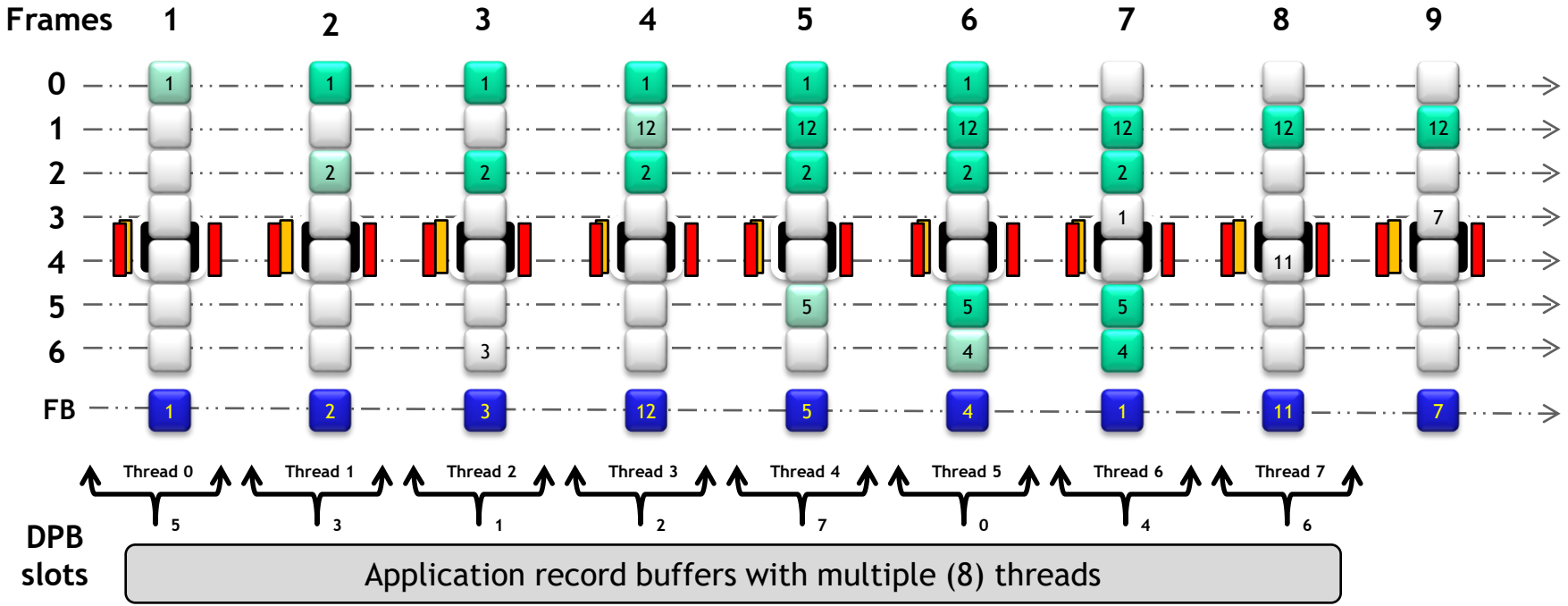
Note: Sparse may require a different queue family to submit, if the video queue families do not advertise sparse capabilities

Available DPB slot | Allocated Target | Reference | Current FB | Image and buffer Memory Barriers | New Begin/end sequence | Manage Image/buffer Sparse Bindings

This work is licensed under a Creative Commons Attribution 4.0 International License

DPB Slots

Multi-threaded cmdBuffer Recording



Make sure to allocate separate command pools for each thread!

Available DPB slot | Allocated Target | Reference | Current FB | Image and buffer Memory Barriers | New Begin/end sequence | Manage Image/buffer Sparse Bindings

This work is licensed under a Creative Commons Attribution 4.0 International License

Special Image Layout Transitions

- **DPB image special handling**
  - DPB images implicitly transition to VK_IMAGE_LAYOUT_UNDEFINED when:
    - Image is used for the first time with a video session
    - Content size or other parameters change within a video session
    - VkVideoSession object is reset
    - DPB slot is assigned for the first time with the image view representing the image
  - DPB images underlying content should not be affected when:
    - Transitioning reference images from VK_IMAGE_LAYOUT_VIDEO_DECODE_DPB or VK_IMAGE_LAYOUT_VIDEO_DECODE_DST to Gfx/compute friendly layouts
- **Video Input images transition**
  - When the content size or parameters change encode input images implicitly transition from VK_IMAGE_LAYOUT_VIDEO_ENCODE_SRC to VK_IMAGE_LAYOUT_UNDEFINED or VK_IMAGE_LAYOUT_PREINITIALIZED.

Please don't forget to transition the images to the correct video layout when the content size change!

This work is licensed under a Creative Commons Attribution 4.0 International License

Video Queries

- **Result Status Query (optional)**
  - Used to check whether a set of operations has been completed successfully
  - Type is VK_QUERY_RESULT_WITH_STATUS_BIT_KHR
  - Can be used with other than video queue families
- **Encode Bitstream Range Query**
  - Describes range of bytes written in the bitstream buffer by video encode commands
  - Type is VK_QUERY_TYPE_VIDEO_ENCODE_BITSTREAM_BUFFER_RANGE_KHR
- **Queries supported with Video**
  - Host side vkGetQueryPoolResults()
- **Queries not supported with Video**
  - Device side: vkCmdCopyQueryPoolResults()

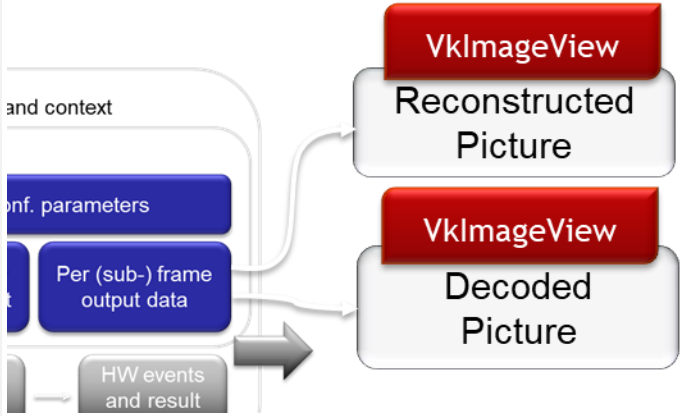
This work is licensed under a Creative Commons Attribution 4.0 International License

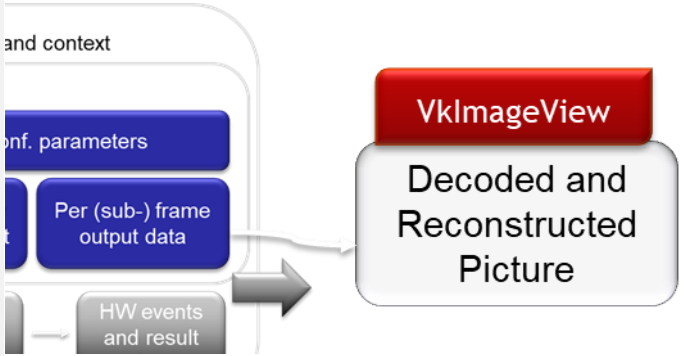
Video Properties and Capabilities

- **Supported codecs for a particular Vulkan video queue**
  - Queried through VkVideoQueueFamilyProperties2KHR, chained to vkGetPhysicalDeviceQueueFamilyProperties() function
- **Supported video decode and encode capabilities**
  - Queried through vkGetPhysicalDeviceVideoCapabilitiesKHR() function
- **Supported video output, input and DPB image formats**
  - Enumerated through vkGetPhysicalDeviceVideoFormatPropertiesKHR() function

This work is licensed under a Creative Commons Attribution 4.0 International License

Output reconstructed pictures implementation variations





VkVideoDecodeCapabilitiesKHR::flags reported by the implementation indicates if the output and reconstructed picture coincide (use one image view) or separate image view resources must be used.

VK_VIDEO_DECODE_CAPABILITY_DPB_AND_OUTPUT_DISTINCT_BIT_KHR - separate images must be allocated used for output and reconstructed picture. However, when linear output is required or output in RGB color space is required then the same output image can be used.

VK_VIDEO_DECODE_CAPABILITY_DPB_AND_OUTPUT_COINCIDE_BIT_KHR – a single image must be used for output and reconstructed image. However, for images requiring linear output a separate output image may still be required, along with an inlined post-processing transfer command.

This work is licensed under a Creative Commons Attribution 4.0 International License

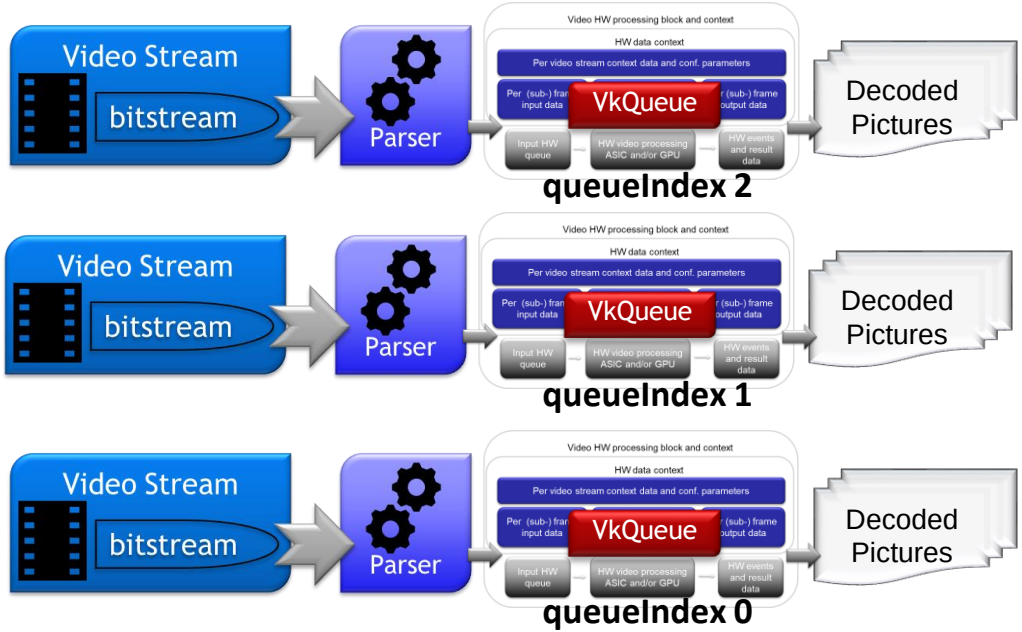
HW load balancing - HW block instance per stream or multiple streams

vkGetDeviceQueue(device, queueFamilyIndex, **queueIndex**, VkQueue);

Use multiple HW instances for decoding (or encoding) to fully utilize the HW.

This exact setup works against a single or multiple instances of the HW instances.

When using a single instance, the HW will time-slice between the submitted work.



This setup is compatible with both, multi-processes multi-thread configuration.

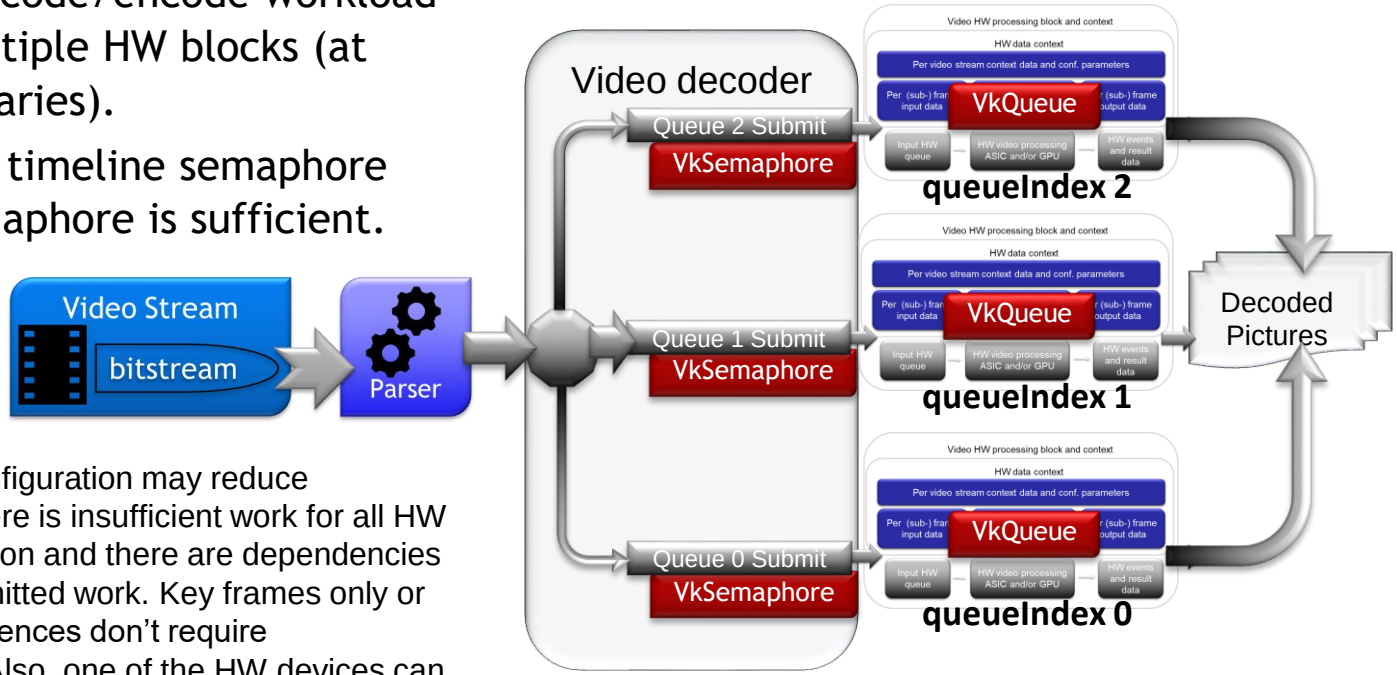
This work is licensed under a Creative Commons Attribution 4.0 International License

HW load balancing- multiple HW blocks per video stream

vkGetDeviceQueue(device, queueFamilyIndex, **queueIndex**, VkQueue);

Split the decode/encode workload between multiple HW blocks (at frame boundaries).

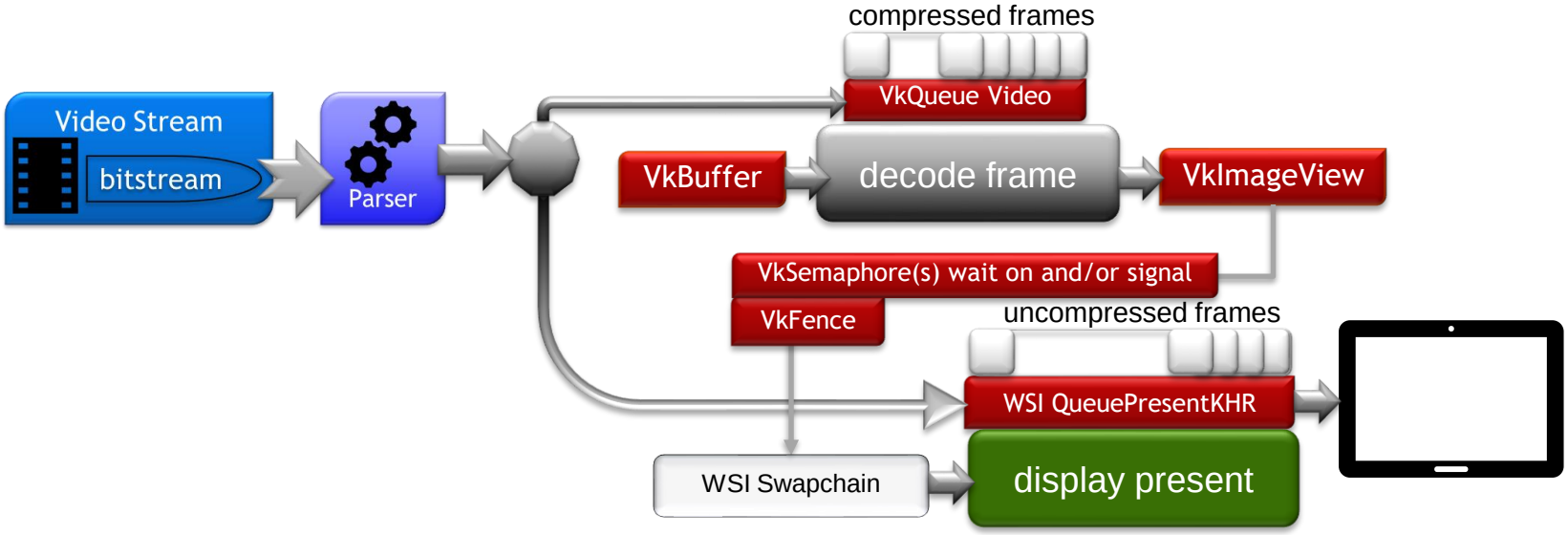
When using timeline semaphore only one semaphore is sufficient.



Warning: This configuration may reduce performance if there is insufficient work for all HW instances to work on and there are dependencies between the submitted work. Key frames only or closed-GOP sequences don't require synchronization. Also, one of the HW devices can already be used by other APIs or devce instances.

This work is licensed under a Creative Commons Attribution 4.0 International License

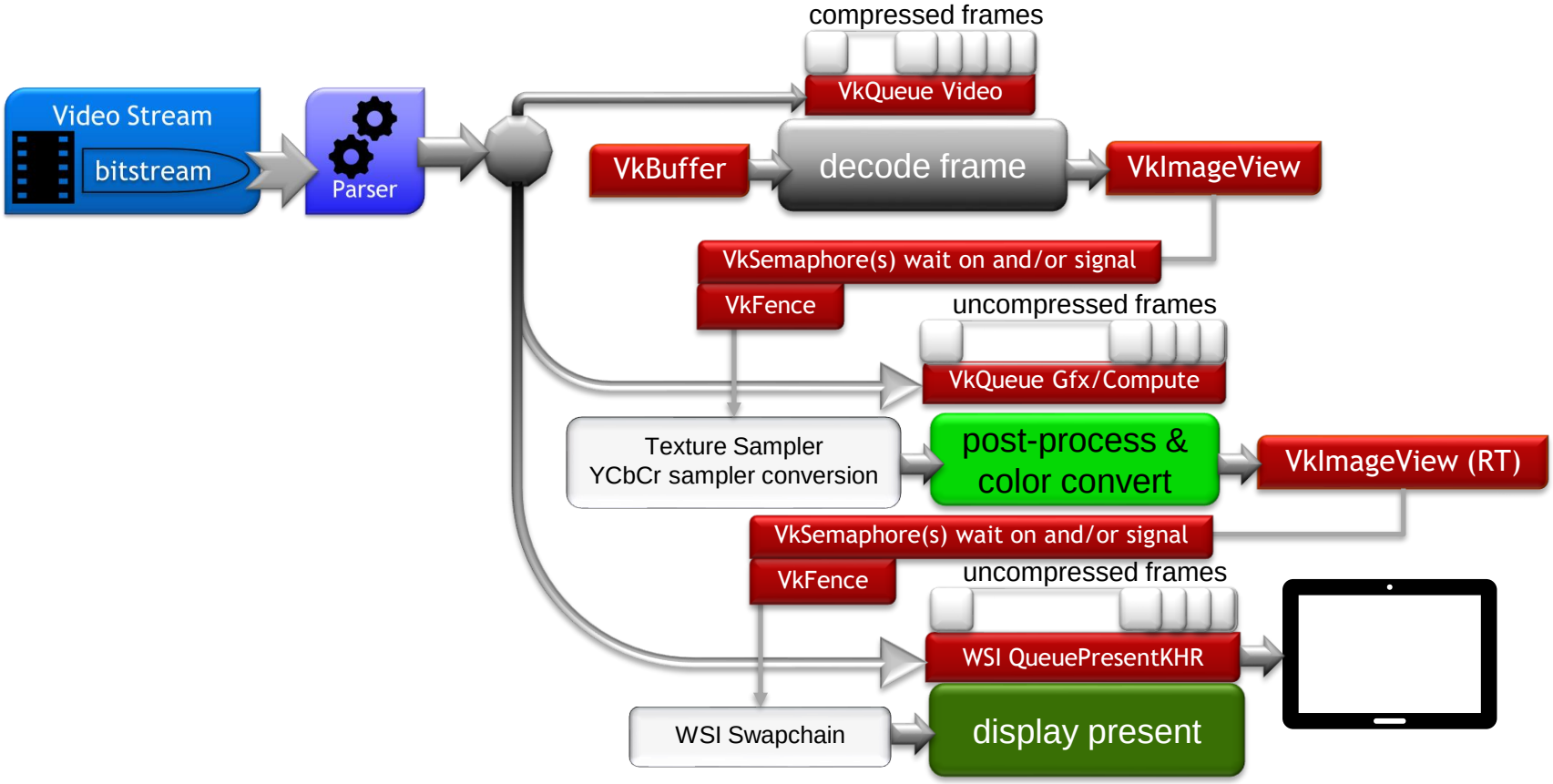
Interaction with the display (WSI)



The display must support YCbCr images for its swapchain and there must be a compatible format that is returned by both vkGetPhysicalDeviceSurfaceFormats2KHR() and vkGetPhysicalDeviceVideoFormatPropertiesKHR().

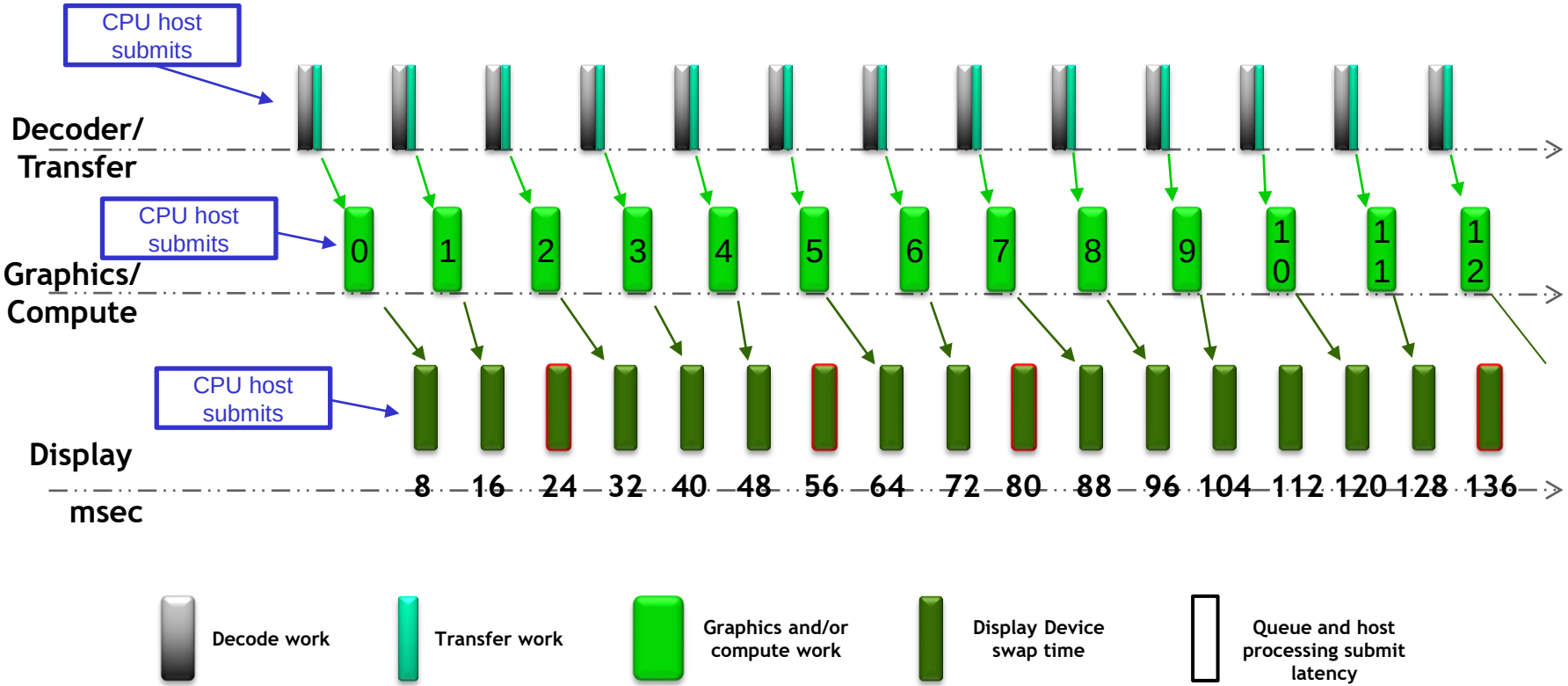
This work is licensed under a Creative Commons Attribution 4.0 International License

Interaction with the graphics and display (WSI)



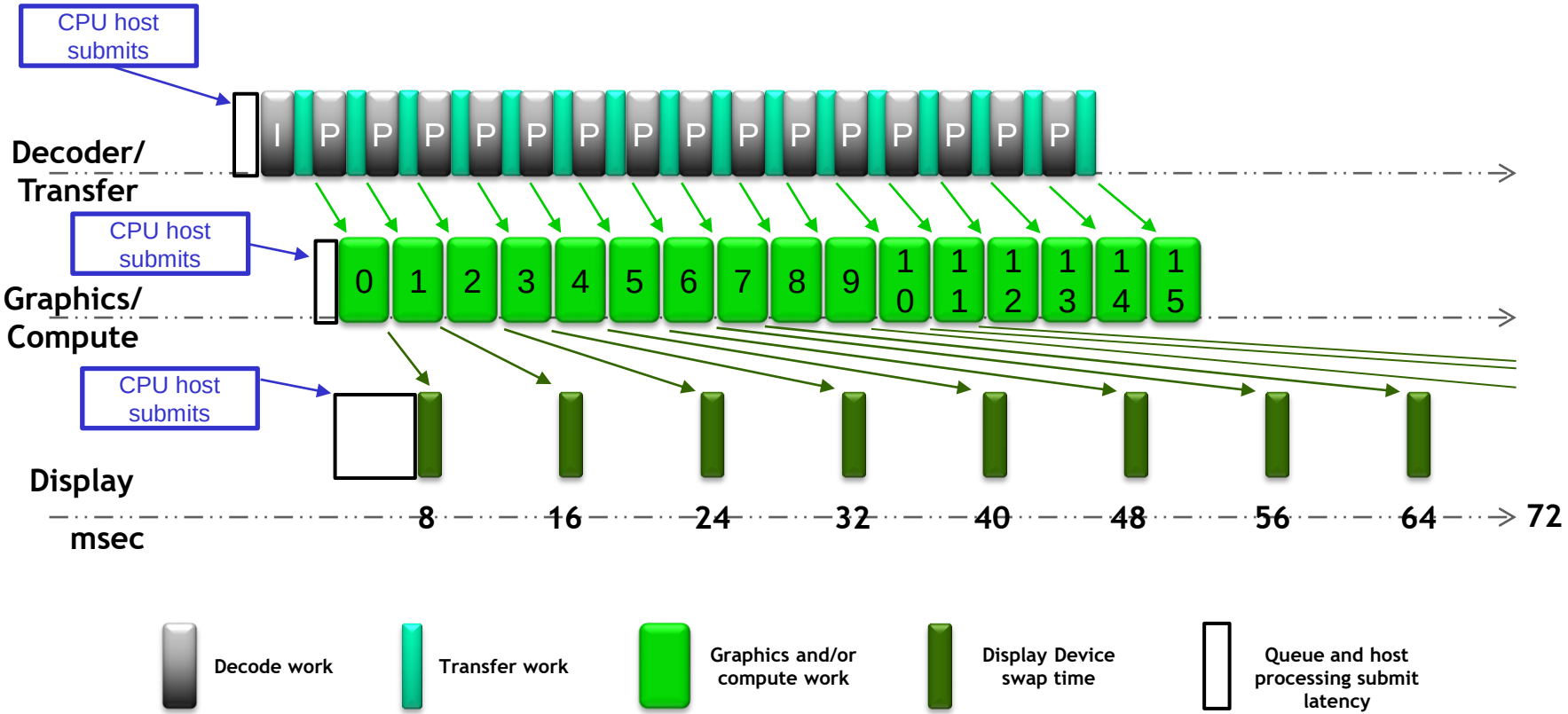
This work is licensed under a Creative Commons Attribution 4.0 International License

HW queues workload graph using Vulkan fences (CPU based synchronization, including semaphore host wait and query waits)



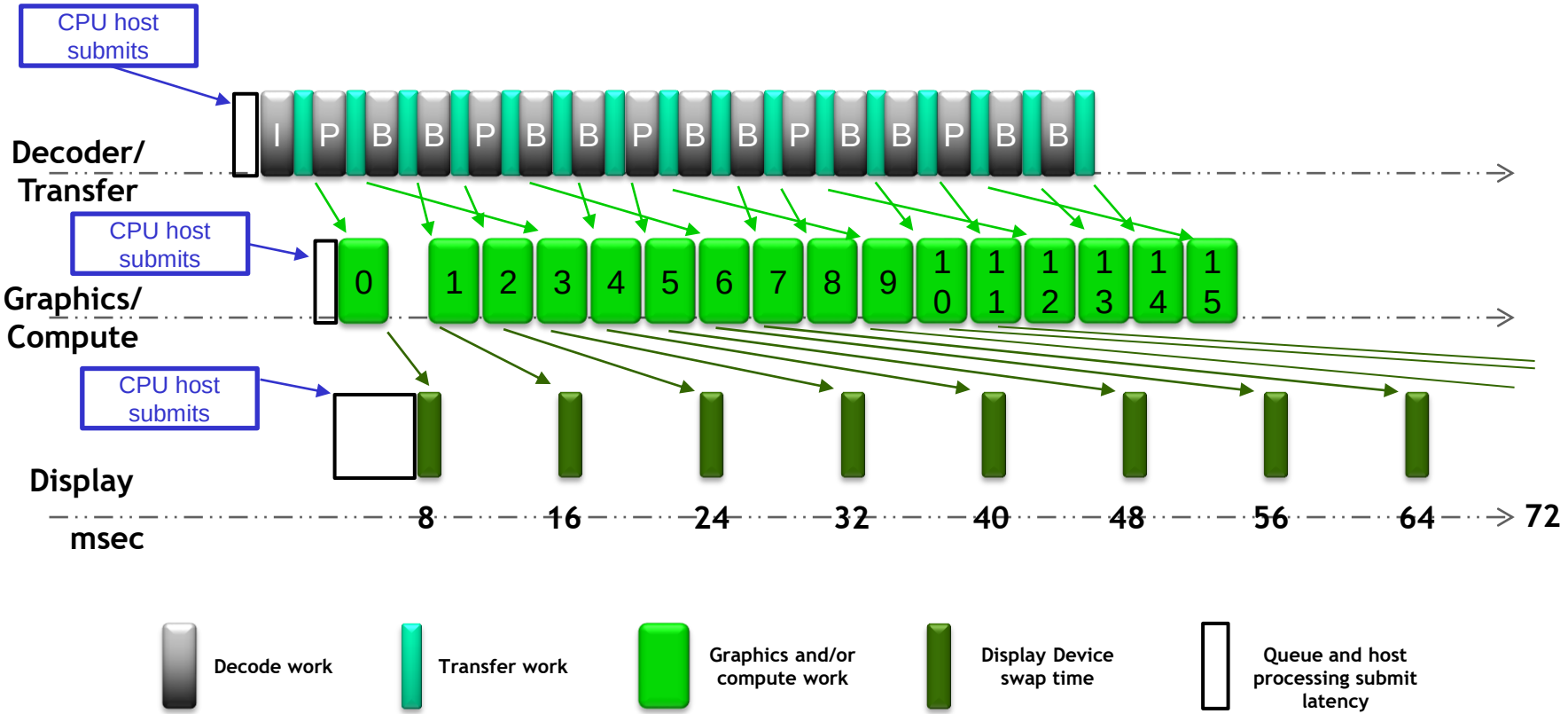
This work is licensed under a Creative Commons Attribution 4.0 International License

HW queues workload graph using Vulkan semaphores



This work is licensed under a Creative Commons Attribution 4.0 International License

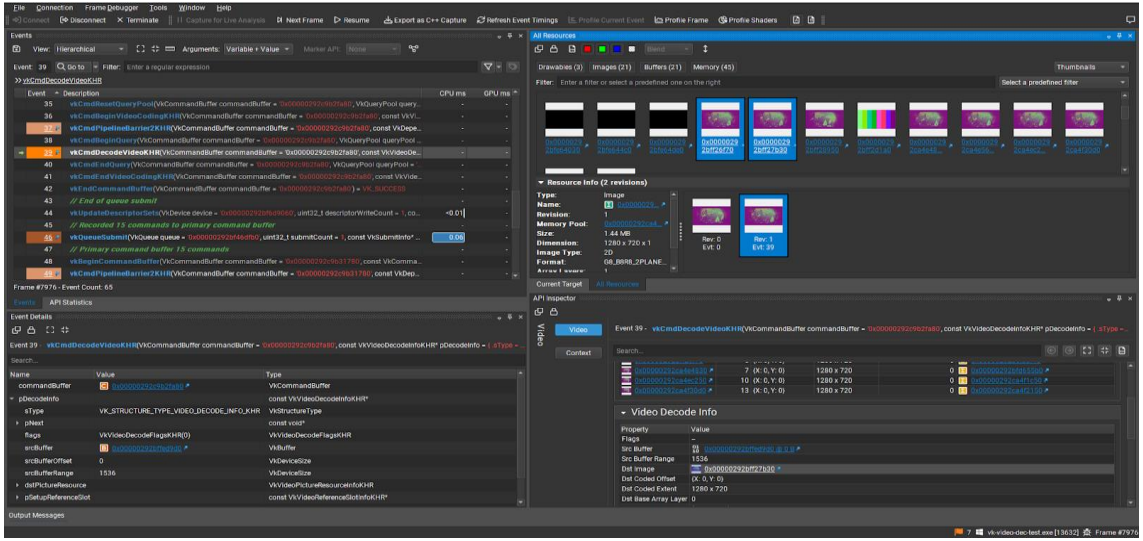
HW queues workload graph using Vulkan semaphores latency because of the video (IPB) GOP structure



This work is licensed under a Creative Commons Attribution 4.0 International License

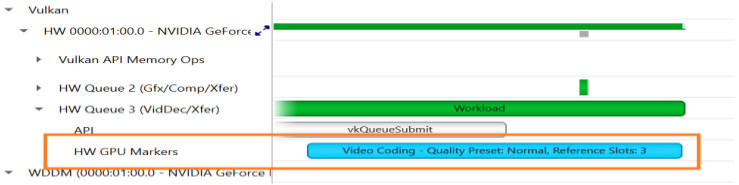
Vulkan video frame timing using NVIDIA Nsight Graphics and Systems profiler

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



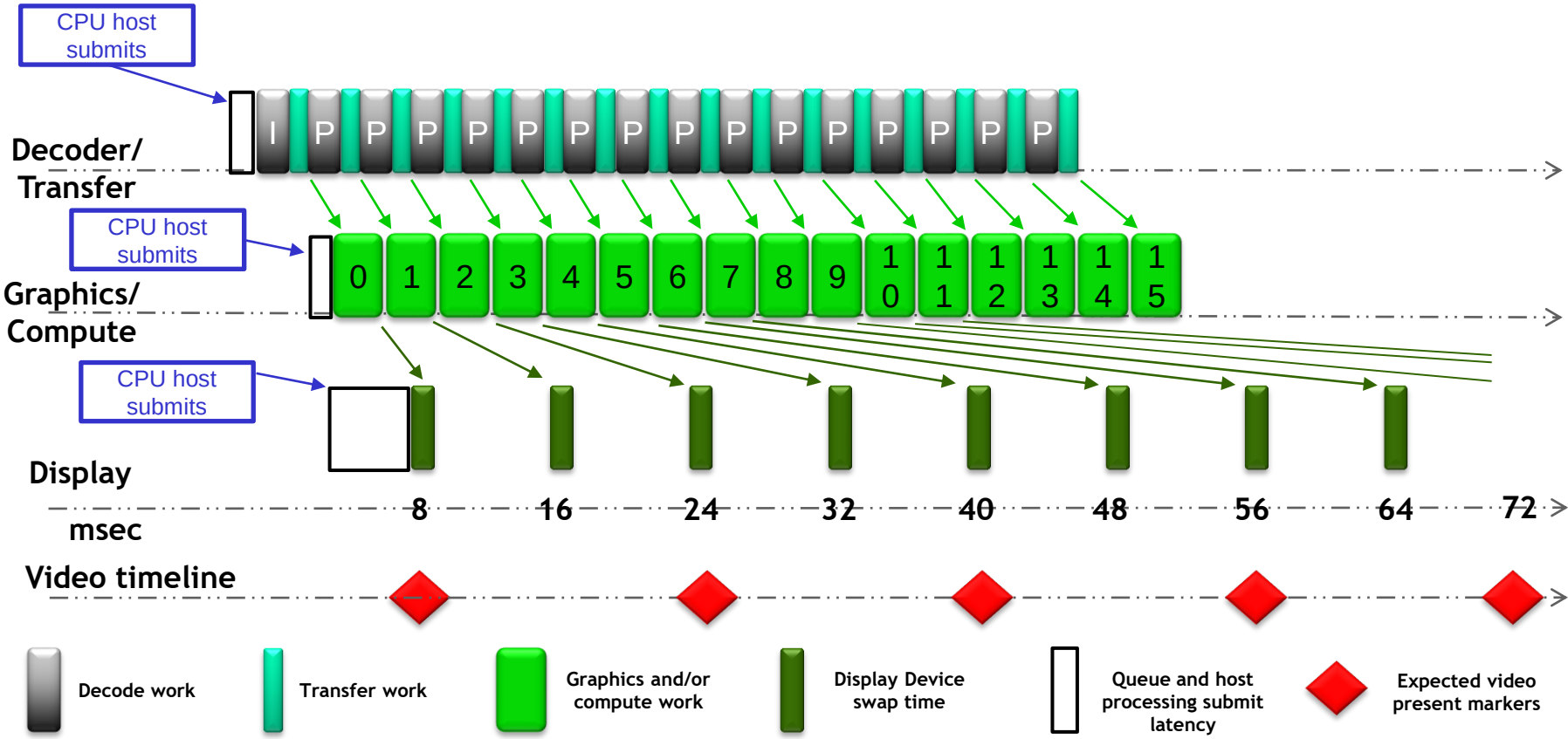
NVIDIA Nsight Developer Tools are a collection of debuggers, profilers, and optimizers that support performance tuning for applications using many graphics APIs, including Vulkan.

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.



This work is licensed under a Creative Commons Attribution 4.0 International License

Dealing with video present time



This work is licensed under a Creative Commons Attribution 4.0 International License

Present time extensions

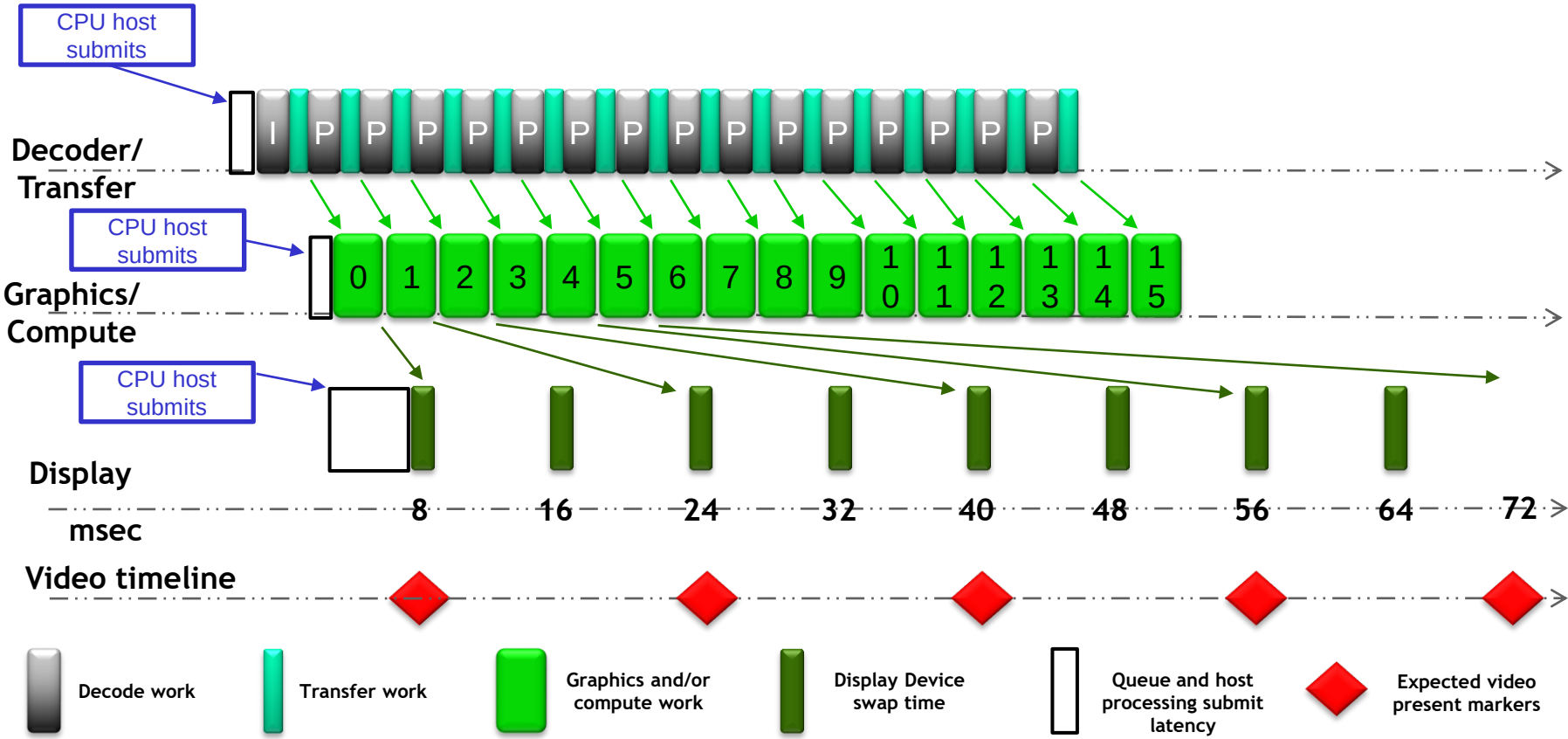
Both VK_GOOGLE_display_timing and VK_EXT_present_timing extensions provide the following features:

- Provide information on how long is the gap between display refreshes in nanoseconds resolution
- When presenting using vkQueuePresentKHR, indicate that image should be displayed at particular time and/or rate. This feature is also known as “Present at time”.
- Provides an accurate timestamp when the image was actually flipped on screen, as well as report on how early it could have been presented

Unfortunately, these extensions are not yet widely supported.

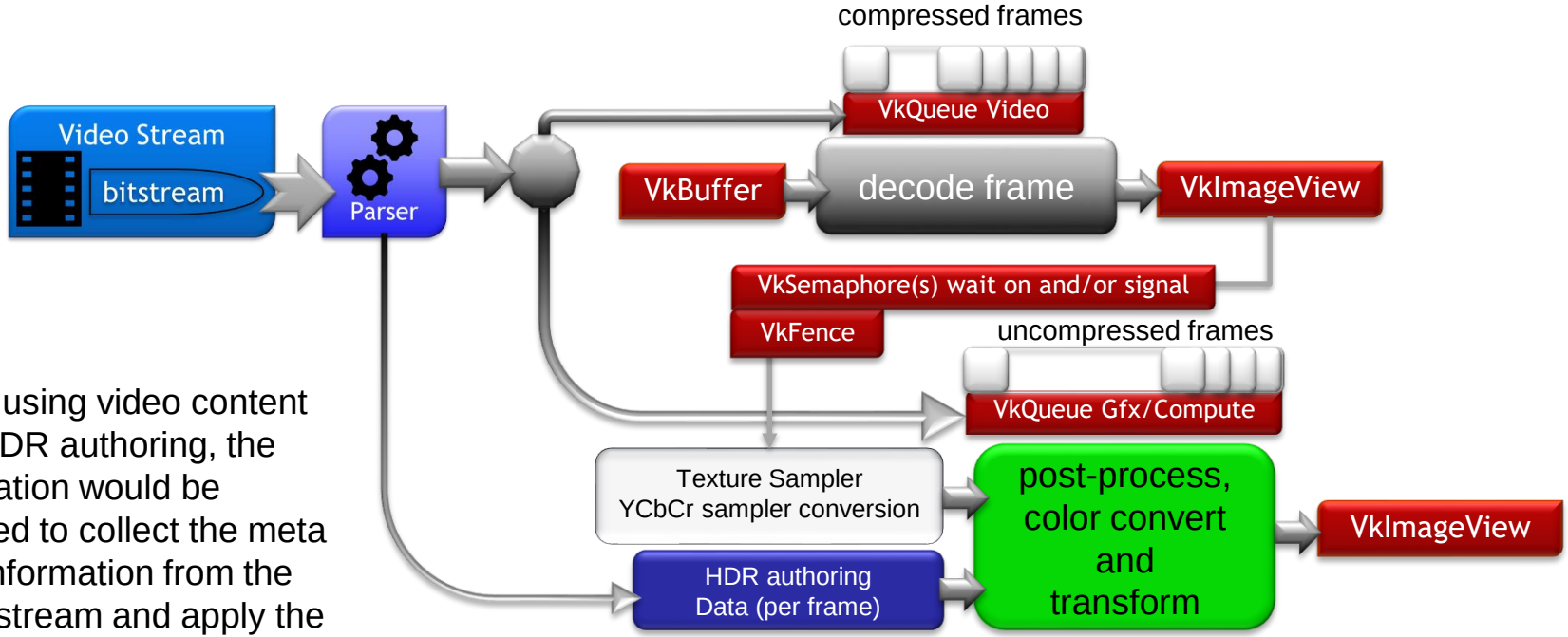
This work is licensed under a Creative Commons Attribution 4.0 International License

Managing video present time using VK_GOOGLE_display_timing extension



This work is licensed under a Creative Commons Attribution 4.0 International License

High dynamic range (HDR)

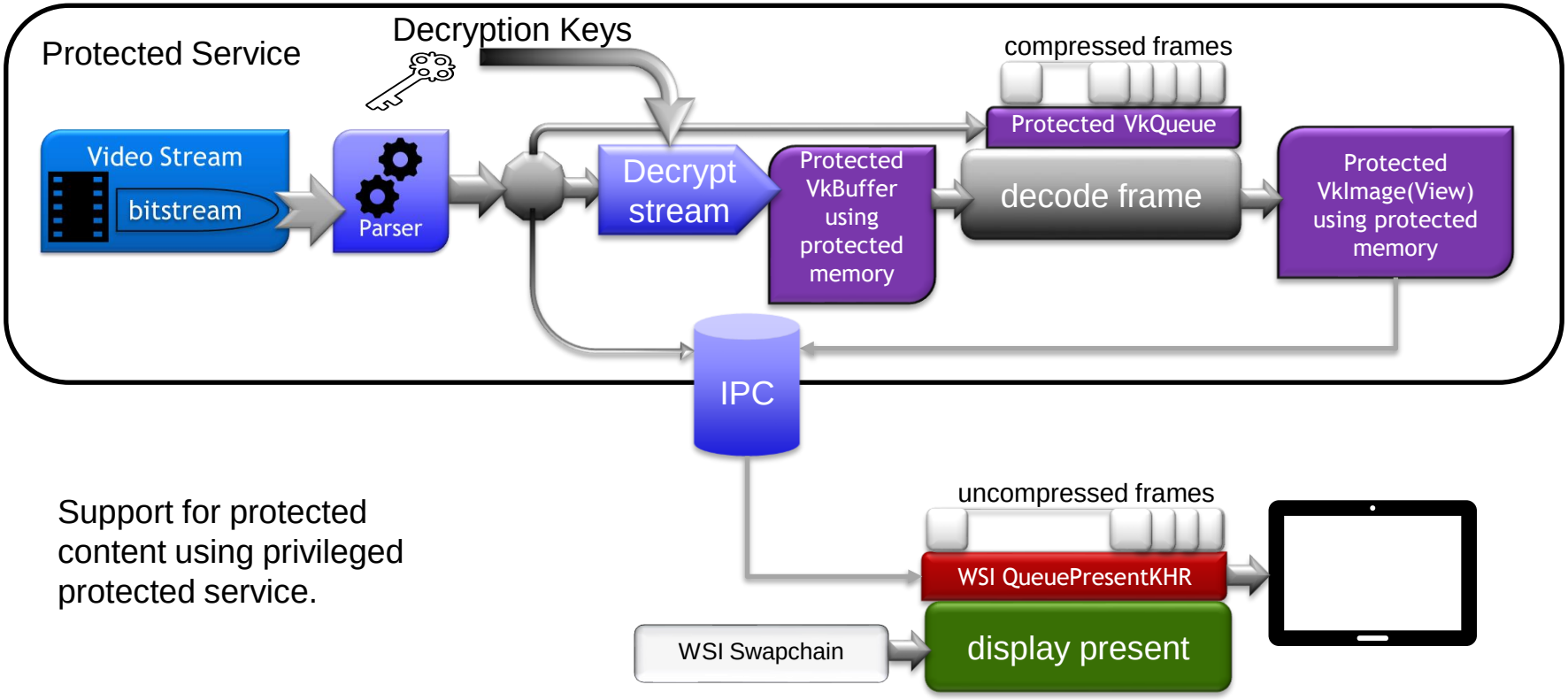


When using video content with HDR authoring, the application would be required to collect the meta data information from the video stream and apply the transformation data as a post-processing step utilizing compute or graphics-enabled queues.

Beyond the content authoring Vulkan Video supports 8, 10 and 12-bit HDR video formats.

This work is licensed under a Creative Commons Attribution 4.0 International License

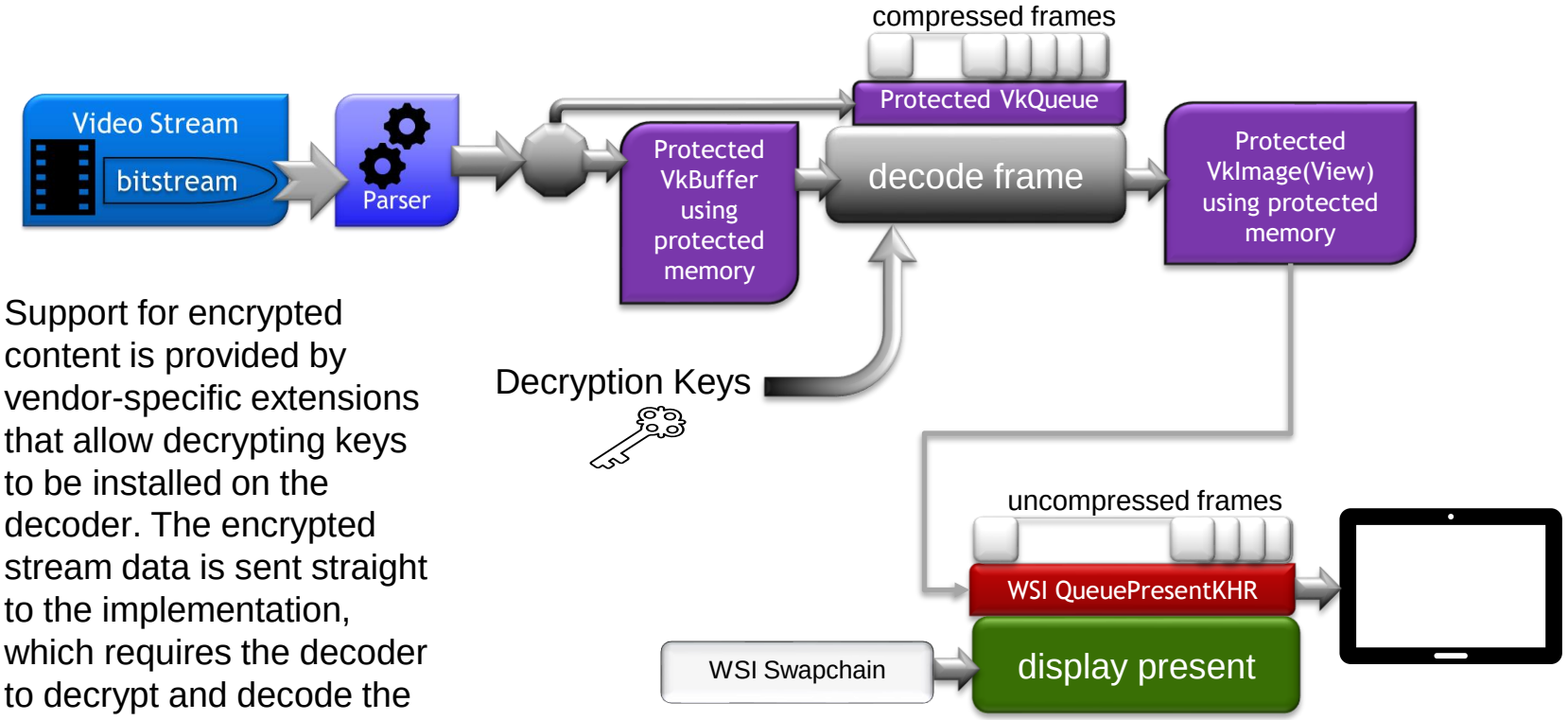
Protected and encrypted compressed content (1)



Support for protected content using privileged protected service.

This work is licensed under a Creative Commons Attribution 4.0 International License

Protected and encrypted compressed content (2)



Support for encrypted content is provided by vendor-specific extensions that allow decrypting keys to be installed on the decoder. The encrypted stream data is sent straight to the implementation, which requires the decoder to decrypt and decode the data.

This work is licensed under a Creative Commons Attribution 4.0 International License

Optimizing Memory Usage

- **Supporting dynamic content stream resize:**
  - Create Sessions with the maximum parameters required for video content: max resolution, max number of DPB, etc. or
  - Create a new session when the content size were to change;
- **Allocate image and buffer resources on demand**
  - When the content requires those resources
- **Free image or buffer resources that are not required**
- **Strip the resources of their physical memory backing**
  - Using sparse memory binding if supported
- **Use the output of the decoded images directly by Vulkan graphics/compute and display processing pipelines**
- **Use the output of the Vulkan graphics/compute to be consumed directly by the encoder's input**

This work is licensed under a Creative Commons Attribution 4.0 International License

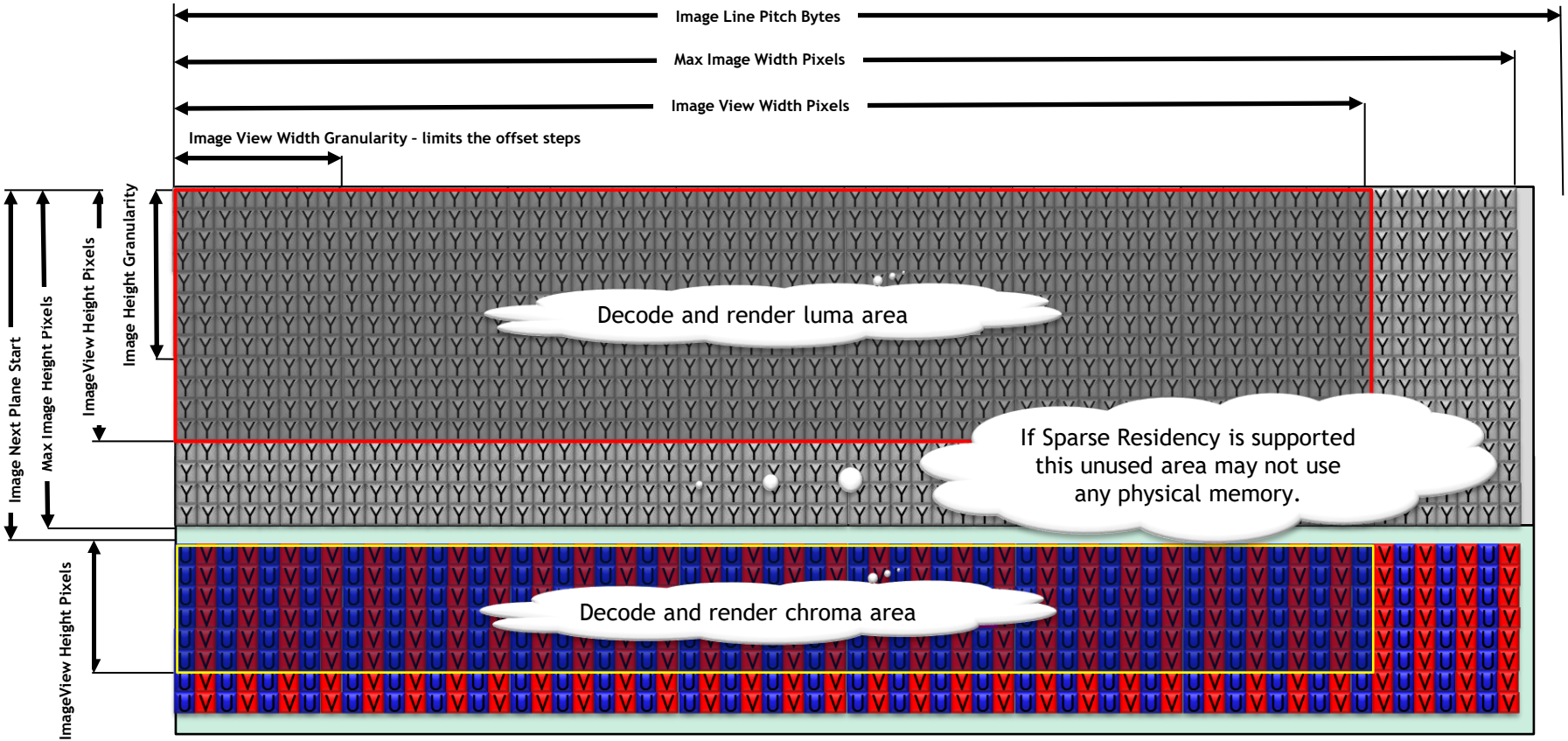
Vulkan Sparse Resources with Video

- **4k/8k and 10/12-bit video content requires significant memory resources for stream buffers and picture images**
  - One frame can be bigger than 2 MB. Video session may require 3-8 input and/or output images and 4 to 16 references that requires hundreds of megabytes of memory
- **IHVs are advised in supporting Vulkan Sparse binding for buffers and images for memory efficient resource management**
  - Sparse Partially-Resident Buffers
  - Support for both Sparse Buffers with VK_BUFFER_CREATE_SPARSE_RESIDENCY_BIT enables for portions of the Vulkan buffers used for the input or output stream to be unmapped
- **Sparse Partially-Resident Images**
  - VK_IMAGE_CREATE_SPARSE_BINDING_BIT and VK_IMAGE_CREATE_SPARSE_RESIDENCY_BIT to support an efficient memory use during content resolution change
  - Use vkQueueBindSparse() before or after the queuing video commands

KHRONOS GROUP

This work is licensed under a Creative Commons Attribution 4.0 International License

Reusing images without reallocation on decode size change



This work is licensed under a Creative Commons Attribution 4.0 International License

Conclusions

- Vulkan Video extension is powerful cross-platform API
- It provides a ton of flexibility in building scalable performing and highly configurable video applications.
- The API is backed-up with implementation by major IHVs and open source.

This work is licensed under a Creative Commons Attribution 4.0 International License

References

- An Introduction to Vulkan Video
- 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.
- NVIDA beta Vulkan drivers

KHRONOS GROUP

This work is licensed under a Creative Commons Attribution 4.0 International License

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.

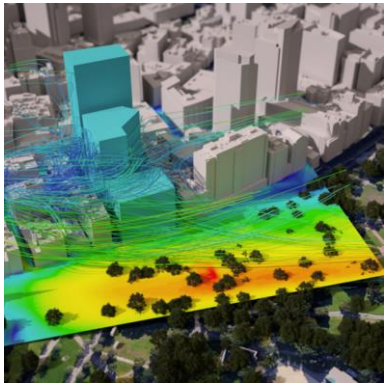
March 20-23 | http://www.nvidia.com/gtc/

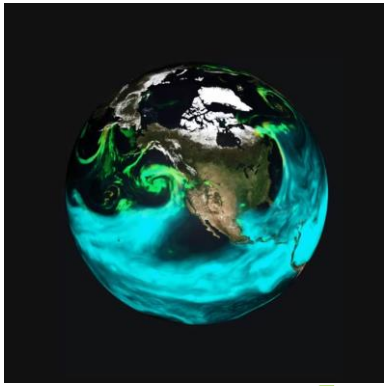
Recommended Sessions and Training

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

This work is licensed under a Creative Commons Attribution 4.0 International License

Thank you!

Questions?

This work is licensed under a Creative Commons Attribution 4.0 International License