Introducing Timeline Semaphores

James Jones
Principle Software Engineer, NVIDIA

Current Vulkan Synchronization Model

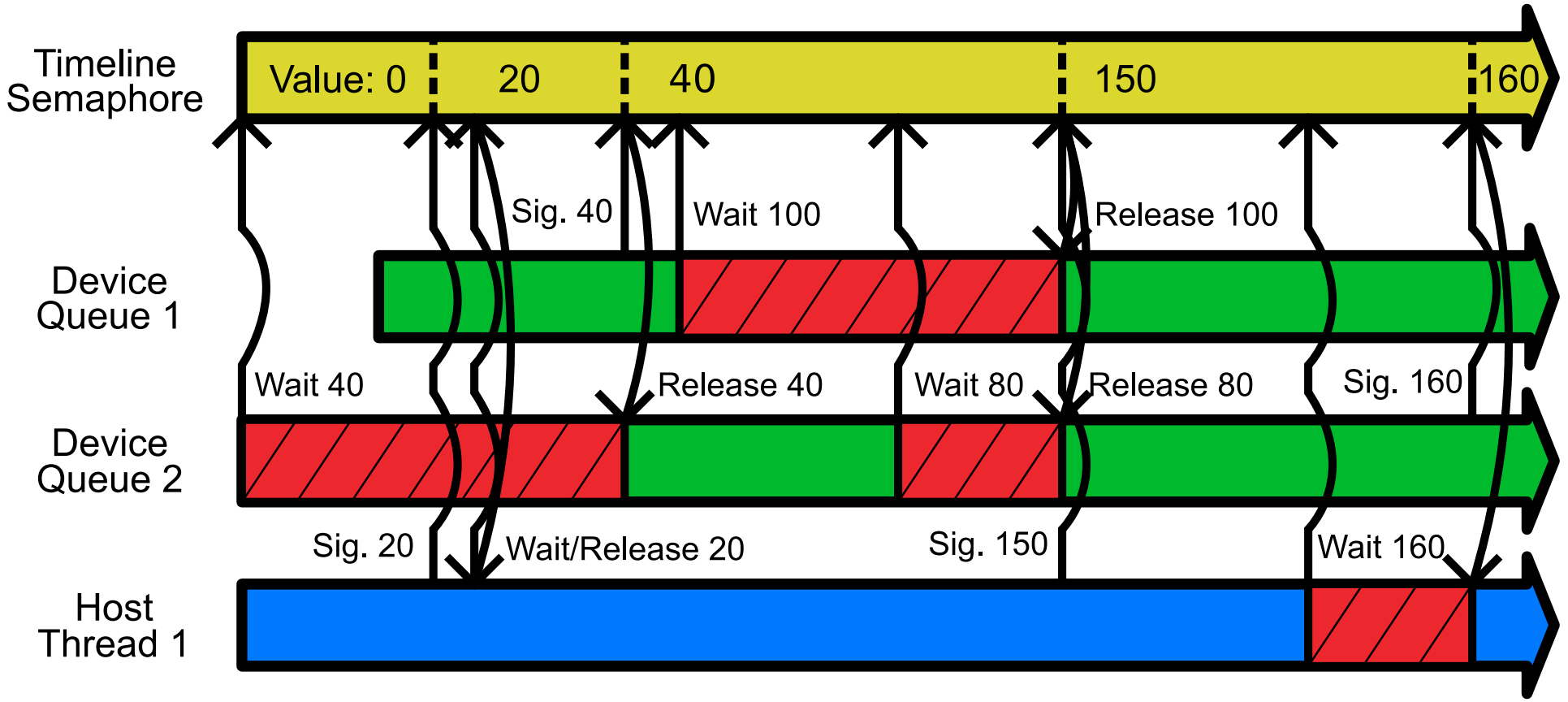
- **Two Coarse-Grained Primitives: VkSemaphore and VkFence**

- **VkSemaphore: Device->Device Synchronization**
  - Binary State
  - Auto-Reset - 1:1 signal:wait relationship
  - Queue operations wait on and signal an arbitrary number of semaphores
  - Reusable, but only in the unsignaled state
  - Signal must be queued before wait is queued

- **VkFence: Device->Host Synchronization**
  - Binary State
  - Manual Reset – 1:<N> signal:wait relationship
  - Queue operations signal at most one fence
  - Reusable, but only in the unsignaled state

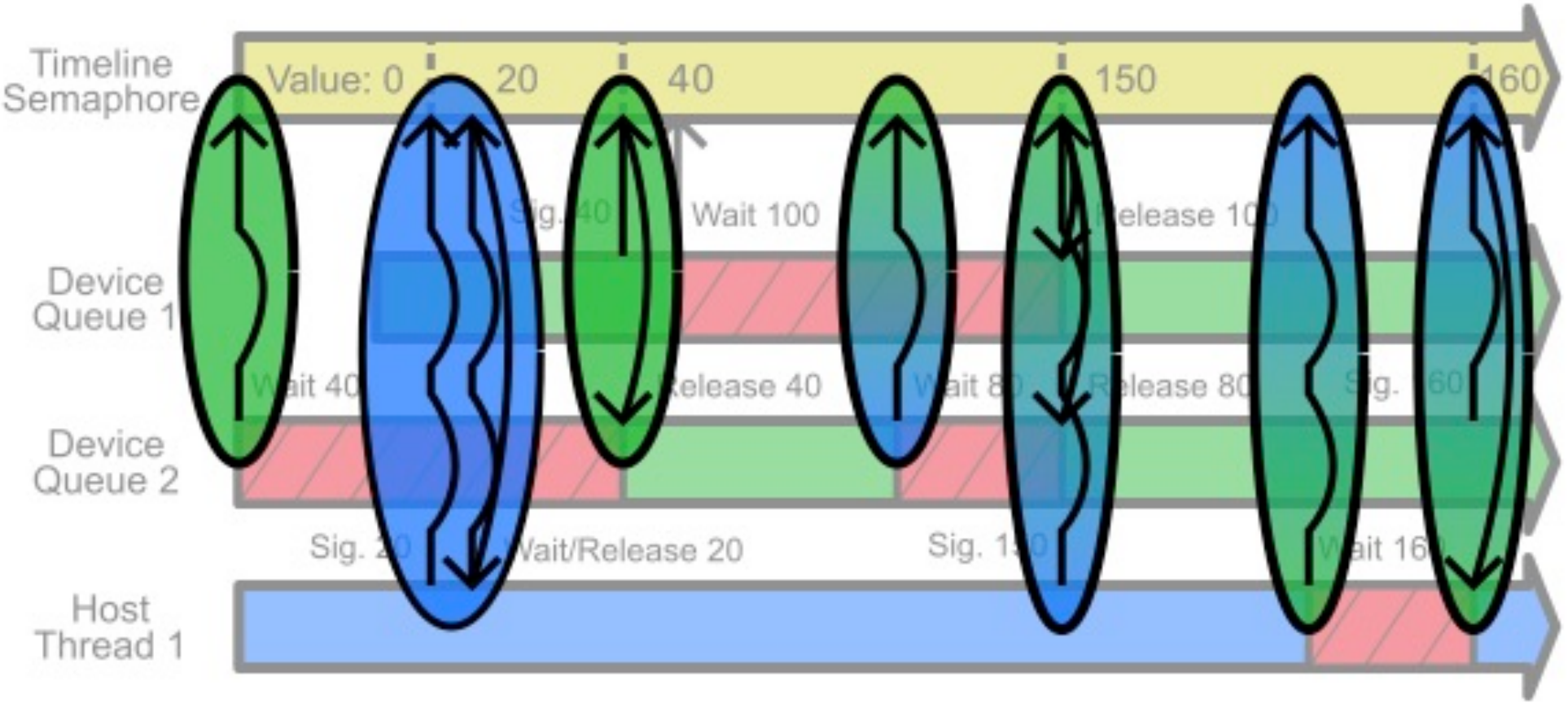
New Model: Timeline Semaphore Primitive

- **From ~10 sync primitives to 1**
  - Some of these operations not directly expressible with existing sync primitives



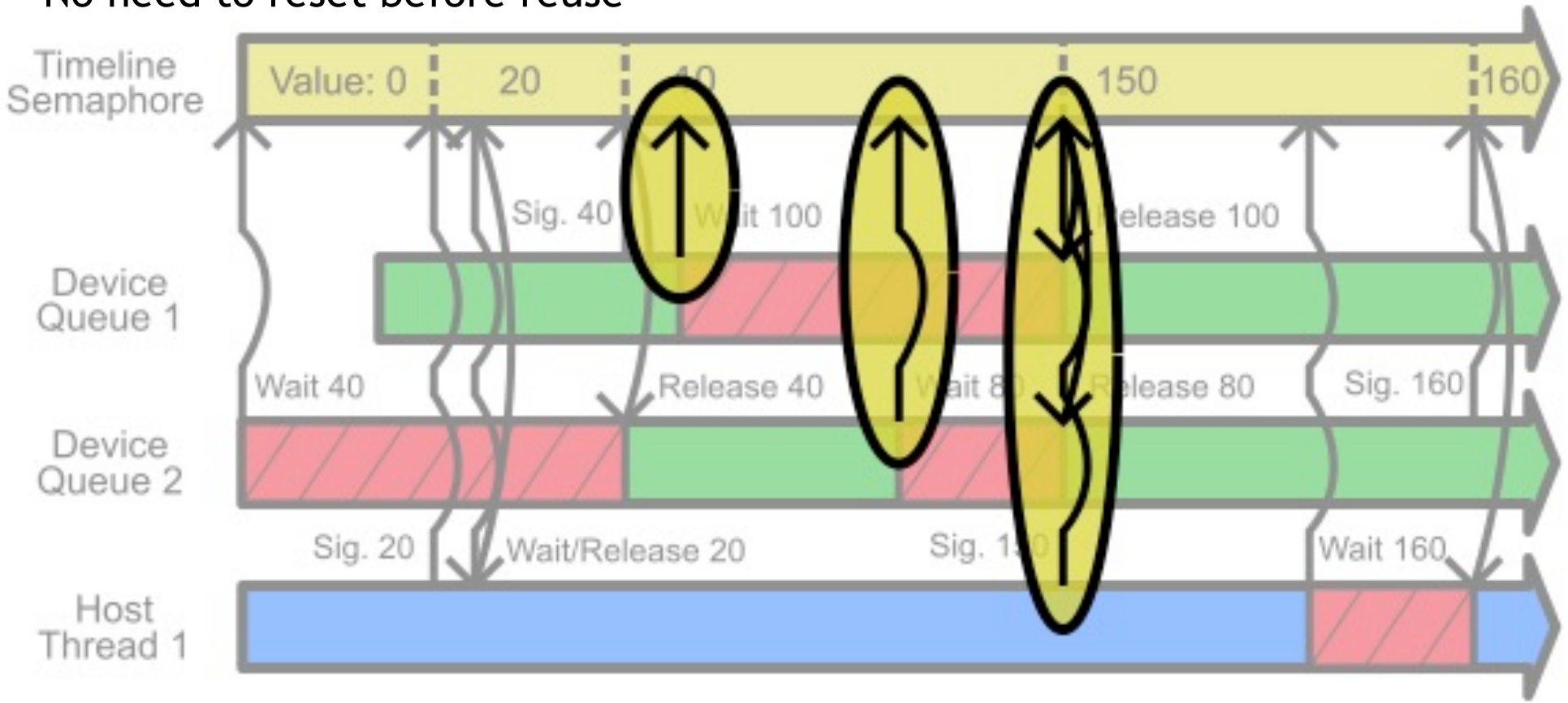
New Model: Timeline Semaphore Primitive

- **Combines Device->Device, Device->Host, Host->Device and Host->Host Sync**
  - Efficient signaling and waiting in any direction



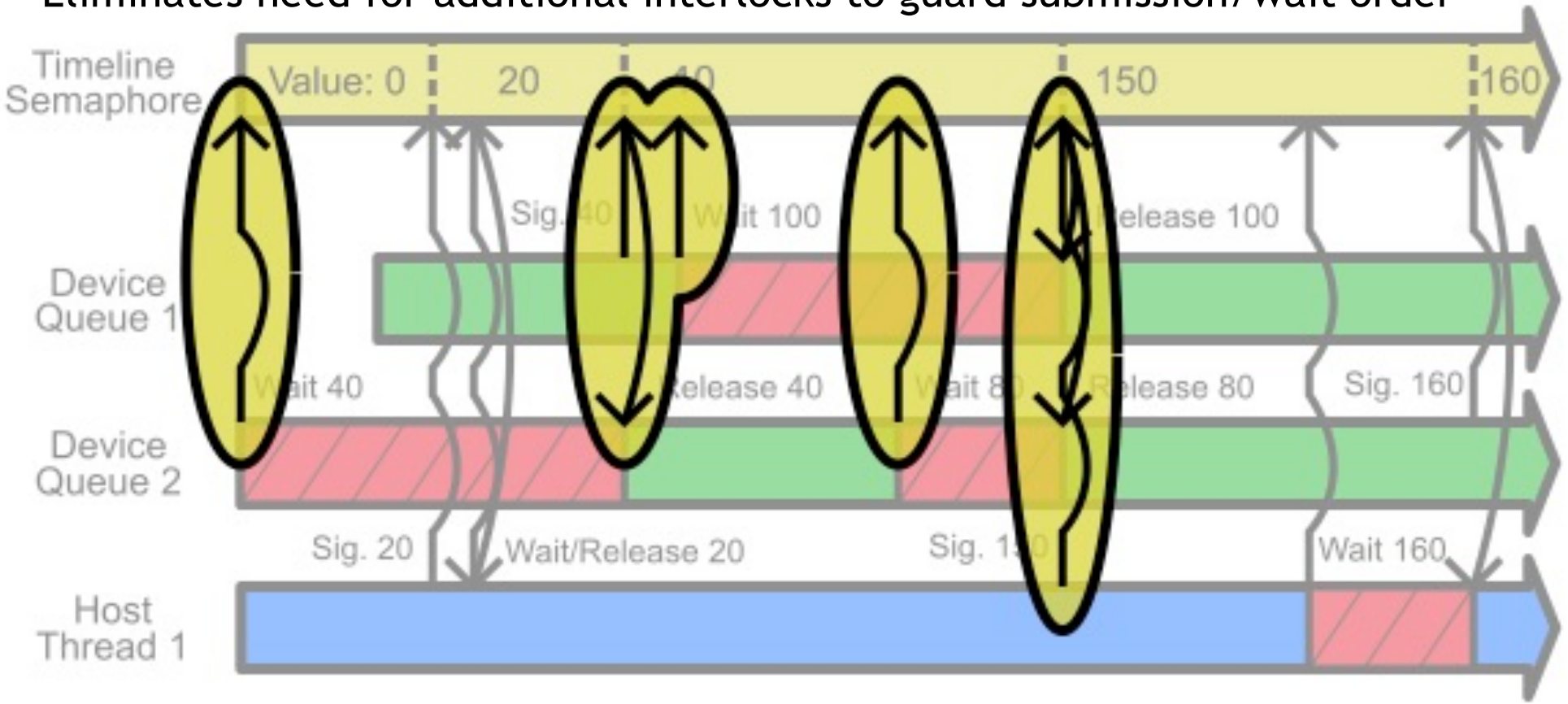
New Model: Timeline Semaphore Primitive

- **1:<N> Signal:Wait Relationship**
  - Consume each signal operation in as many waiters as needed, including zero
  - No need to reset before reuse



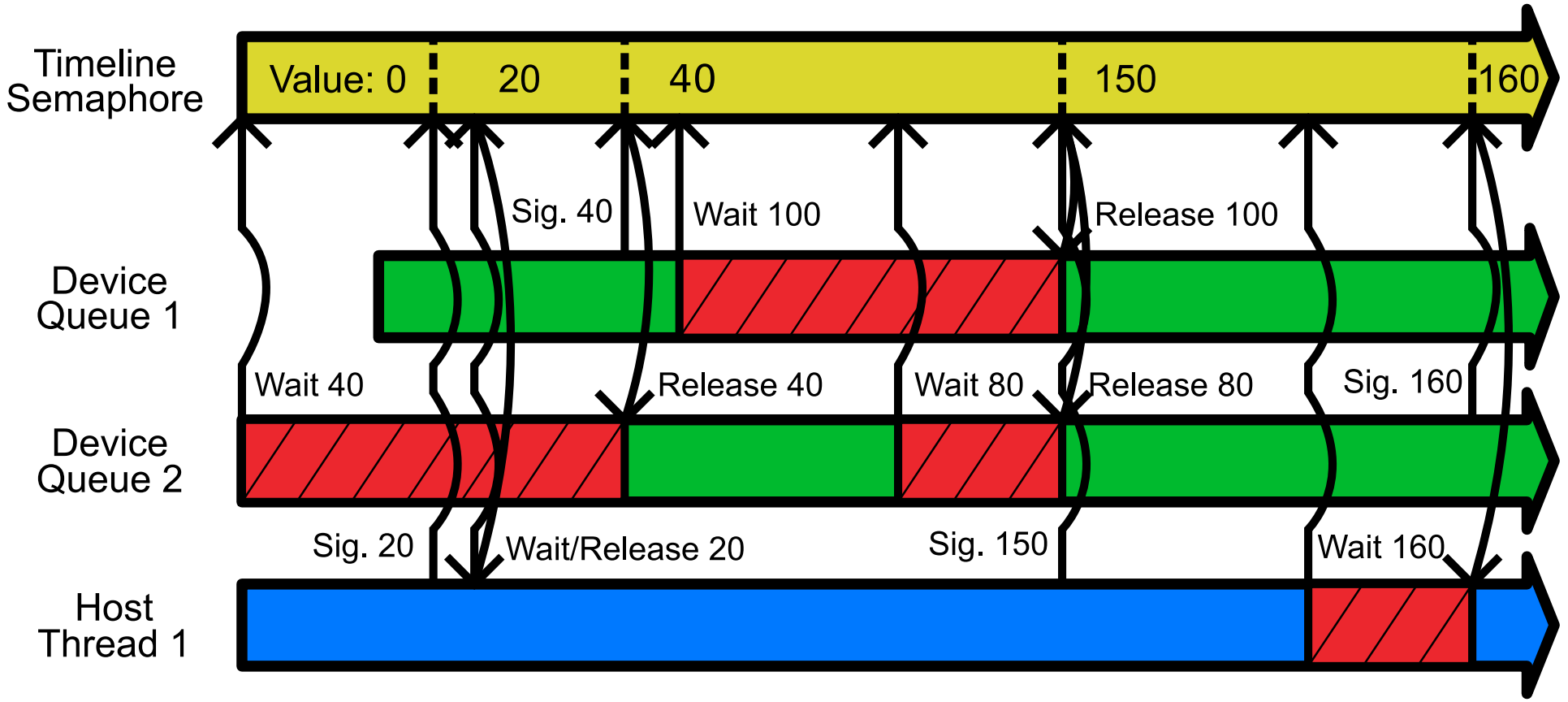
New Model: Timeline Semaphore Primitive

- **Wait-Before-Signal**
  - Begin a Host wait or submit a Device wait before queuing its signal
  - Eliminates need for additional interlocks to guard submission/wait order



New Model: Timeline Semaphore Primitive

- **Allows Multiple In-Flight Signals and Asynchronous Waits on One Semaphore**
  - Subsequent signals do not impact existing or future waits on prior signals



What Is A Vulkan Timeline Semaphore?

- **Extends Existing VkSemaphore API**
  - Supported by all core VkQueue operations that use VkSemaphore objects
  - Export and Import across processes or APIs using existing VkSemaphore APIs

- **Functional Superset of Both Binary-Type VkSemaphores and VkFence**

- **64-bit Monotonically Increasing Counter Replaces Binary VkSemaphore State**
  - Values can now be descriptive, e.g. a monotonic timestamp, frame count, etc.

- **New APIs to Signal and Wait From Host Threads**

- **Broad OS Support**
  - Initially Windows 7 through 10, Linux, Android

Remaining Limitations and Compromises

- **No Window-System Integration API Support**
  - vkQueuePresentKHR() and vkAcquireNextImageKHR() not supported
  - The OS/Window System infrastructure is not ready everywhere yet
  - Several API-semantics issues to work through as well (Input Welcome!)

- **Import/Export Not a Required Feature**
  - Relies on kernel-level support that is not available everywhere yet
  - Works on Windows 10+, and Linux/Android devices with newer kernels/drivers

- **64-bit Values, but Sometimes Only 31-bit Range Between Outstanding Operations**
  - Allows Implementations to hide wrapping when using 32-bit HW or OS primitives
  - Still allows use of full 64-bit range if gap between signals and waits is reasonable

When?

- **Aiming to Ship VK_KHR_timeline_semaphore Specification in August**
  - Windows and Linux implementations ready at launch
  - Native Android implementations coming with device updates
  - Available via a layer for devices without native driver support

Questions?

- Thanks to Jason Ekstrand and Lionel Landwerlin at Intel for Leading Development of the Spec, CTS, and Layered Implementations.