AMD HIP + GSTREAMER

Gstreamer conference 2025

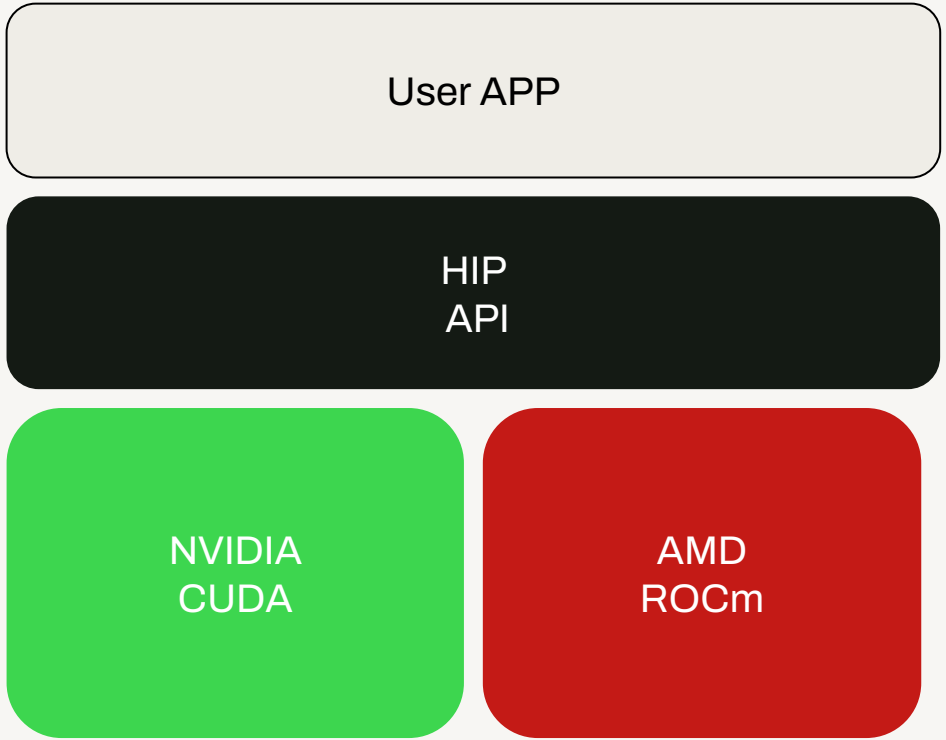
HIP

The Heterogeneous-computing Interface for Portability (HIP) API is a **C++ runtime API and kernel language** that lets developers create portable applications running in heterogeneous systems, **using CPUs and AMD GPUs or NVIDIA GPUs from a single source code.**

https://rocm.docs.amd.com/projects/HIP/en/docs-develop/what_is_hip.html

HIP

The Heterogeneous-computing Interface for Portability (HIP) API is a **C++ runtime API and kernel language** that lets developers create portable applications running in heterogeneous systems, **using CPUs and AMD GPUs or NVIDIA GPUs from a single source code.**



https://rocm.docs.amd.com/projects/HIP/en/docs-develop/what_is_hip.html

HIP vs CUDA API

```cuda
#include <cuda_runtime.h>
__global__ void hello_world_kernel()
{
    printf("Hello, World!\n");
}
int main()
{
    static constexpr unsigned int grid_size  = 1;
    static constexpr unsigned int block_size = 1;
    hello_world_kernel<<<grid_size, block_size>>>();

    cudaDeviceSynchronize();
}
```

```
$ nvcc example.cu -o example
```

```hip
#include <hip/hip_runtime.h>
__global__ void hello_world_kernel()
{
    printf("Hello, World!\n");
}
int main()
{
    static constexpr unsigned int grid_size  = 1;
    static constexpr unsigned int block_size = 1;
    hello_world_kernel<<<grid_size, block_size>>>();

    hipDeviceSynchronize();
}
```

```
$ hipcc example.cpp -o example
```

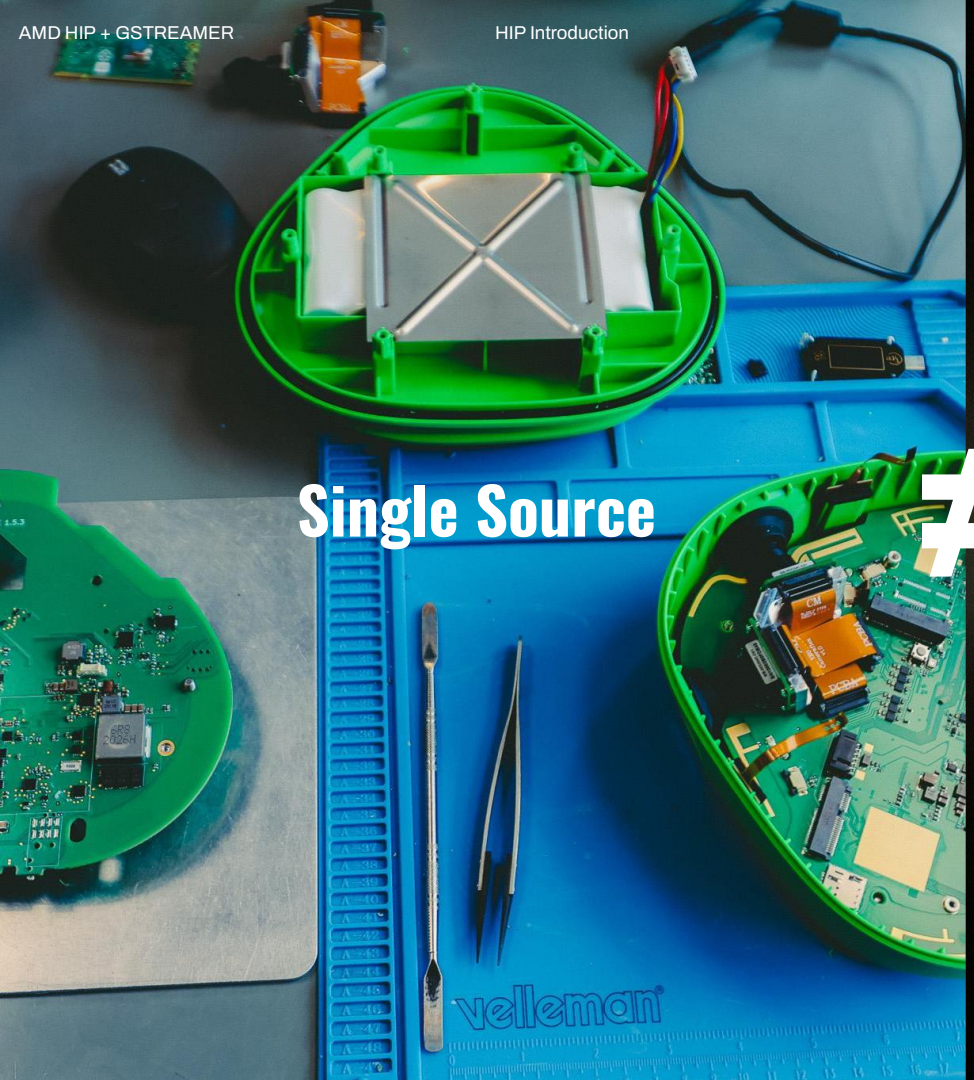
HIP vs CUDA API

```cpp
#include <cuda_runtime.h>
__global__ void hello_world_kernel()
{
    printf("Hello, World!\n");
}
int main()
{
    static constexpr unsigned int grid_size  = 1;
    static constexpr unsigned int block_size = 1;
    hello_world_kernel<<<grid_size, block_size>>>();

    cudaDeviceSynchronize();
}
```

$ **nvcc** example.cu -o example

```cpp
#include <hip/hip_runtime.h>
__global__ void hello_world_kernel()
{
    printf("Hello, World!\n");
}
int main()
{
    static constexpr unsigned int grid_size  = 1;
    static constexpr unsigned int block_size = 1;
    hello_world_kernel<<<grid_size, block_size>>>();

    hipDeviceSynchronize();
}
```

$ **hipcc** example.cpp -o example



Single Source ≠ Single Binary

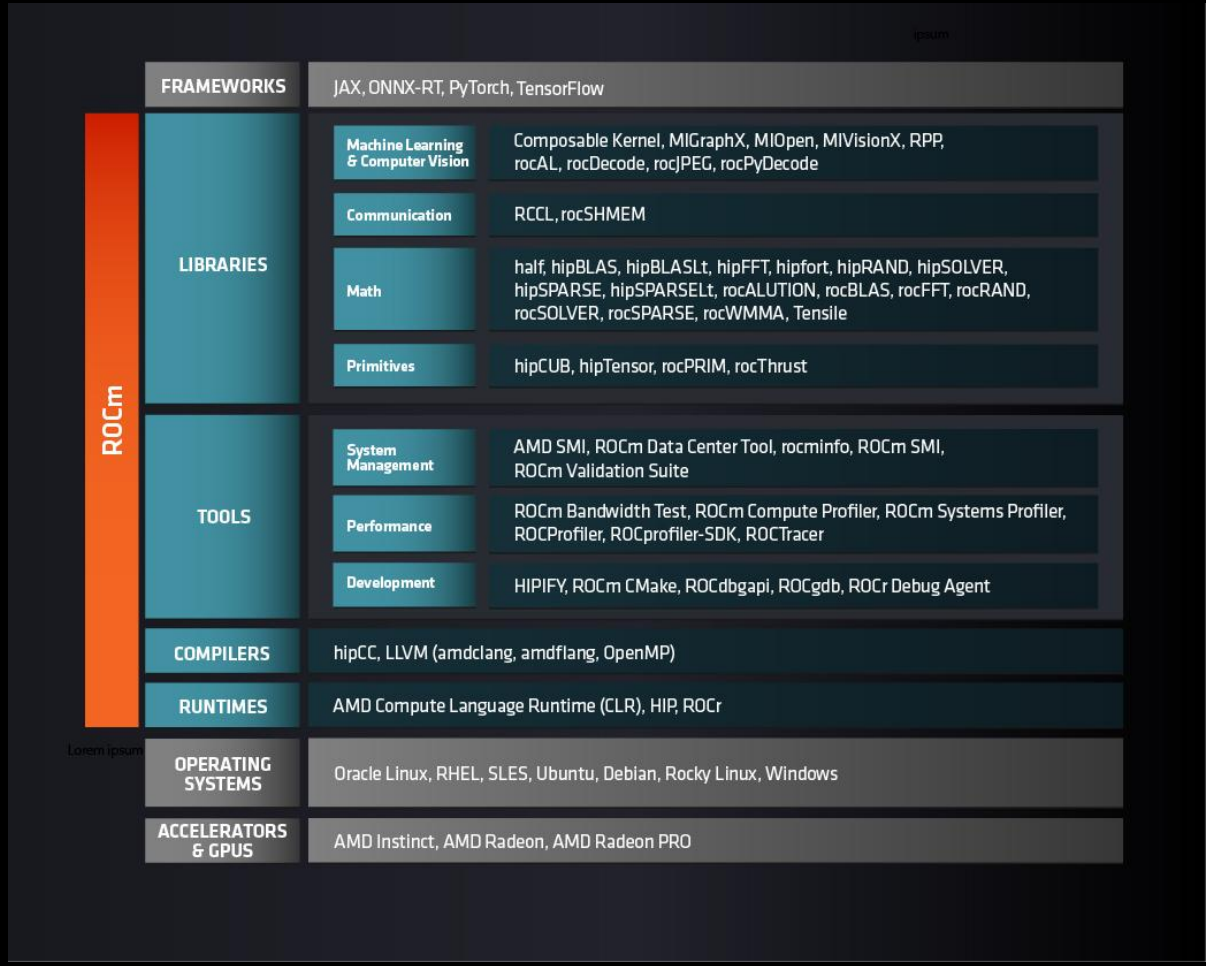
HIP NVIDIA Runtime

```
…..
#define hipEventDefault cudaEventDefault
#define hipEventBlockingSync cudaEventBlockingSync
#define hipEventDisableTiming cudaEventDisableTiming
#define hipEventInterprocess cudaEventInterprocess
#define hipEventReleaseToDevice 0 /* no-op on CUDA platform */
#define hipEventReleaseToSystem 0 /* no-op on CUDA platform */

inline static hipError_t hipDeviceSynchronize() {
  return hipCUDAErrorTohipError(cudaDeviceSynchronize());
}
….
```

https://github.com/ROCm/rocm-systems/blob/develop/projects/hipother/hipnv/include/hip/nvidia_detail/

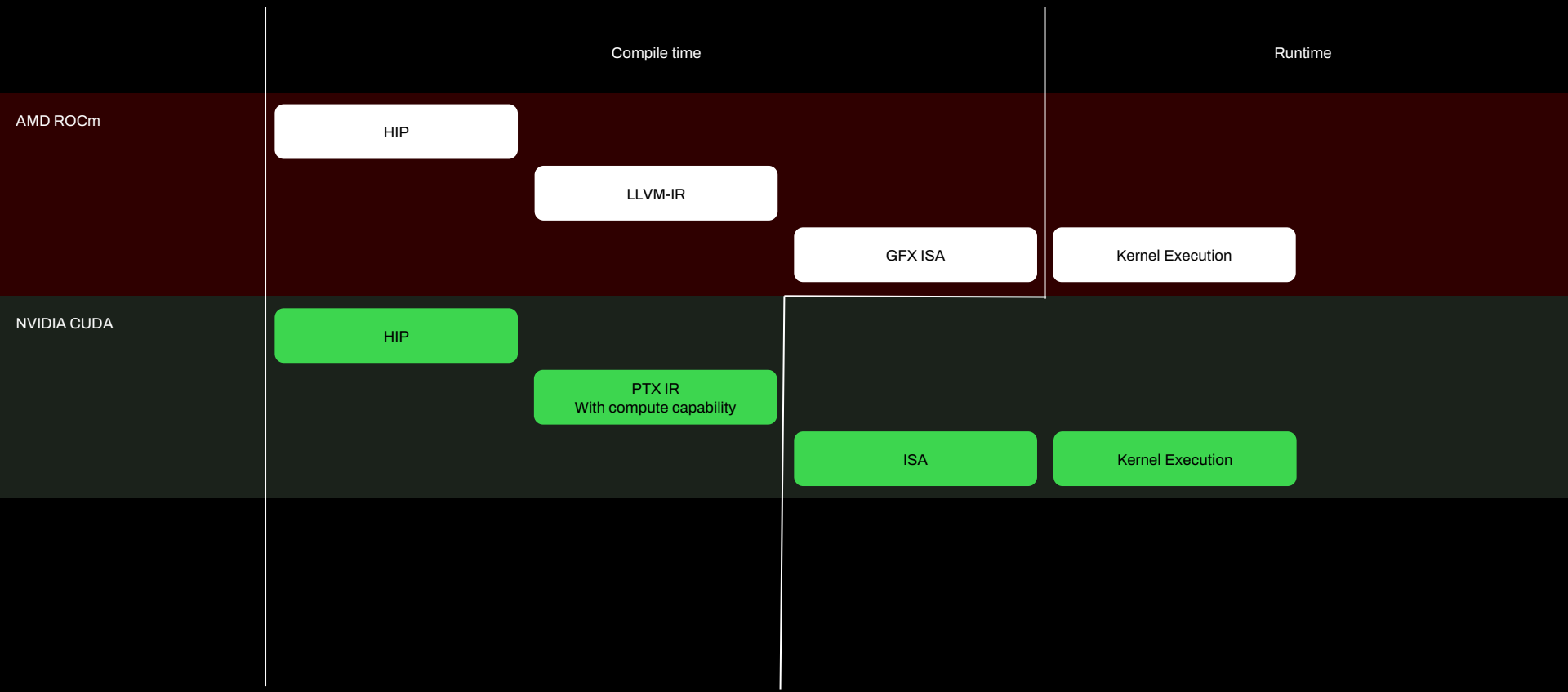
HIP AMD Runtime ROCm

- Replacements for most NVIDIA libraries that are API compatible
- LLVM based compiler

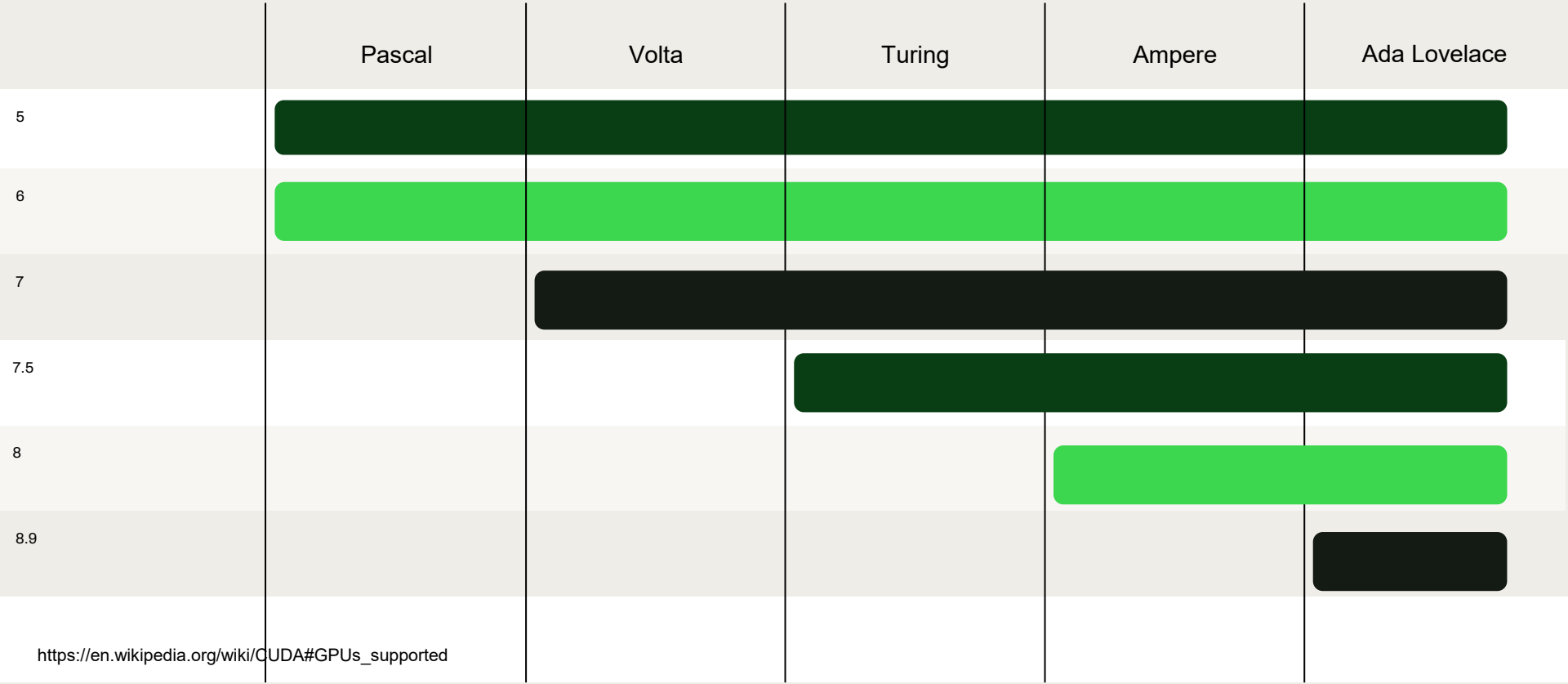


https://rocm.docs.amd.com/en/latest/what-is-rocm.html

Kernel binary portability

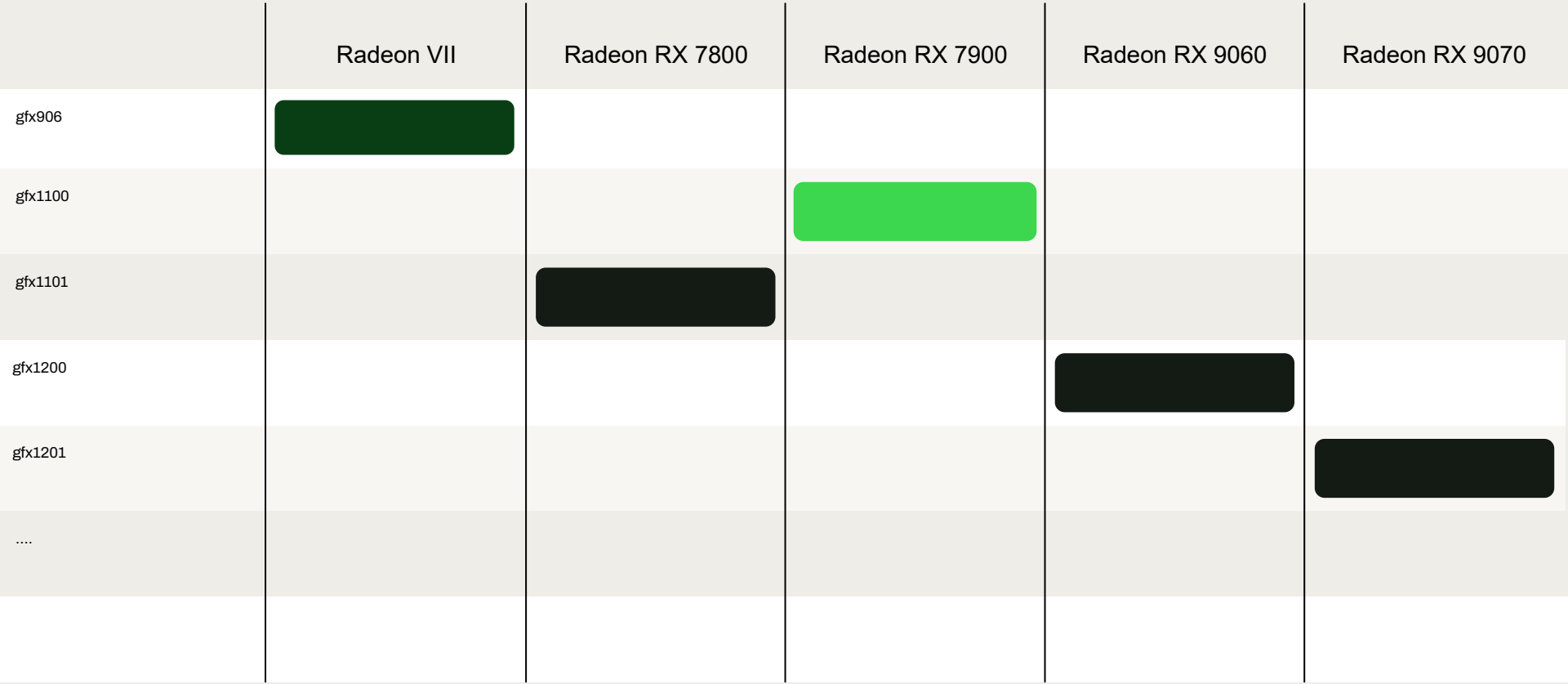


Nvidia Compute capabilities



https://en.wikipedia.org/wiki/CUDA#GPUs_supported

AMD ISA


Radeon VII
Radeon RX 7800
Radeon RX 7900
Radeon RX 9060
Radeon RX 9070
gfx906
gfx1100
gfx1101
gfx1200
gfx1201
....

GStreamer HIP Integration

Introducing GstHip

- New GPU backend integrating **AMD HIP** into GStreamer
- Provides unified GPU pipelines for **AMD** and **NVIDIA**

- Agenda
  - GstHip vs GstCuda overview
  - Unified backend handling
  - Core library & plugin

GstHip vs GstCuda

- HIP and CUDA are structurally very similar
  - Both follow the same programming model
- CUDA provides two APIs: Driver API and Runtime API
- HIP offers only the CUDA Runtime API compatible one

- GstCuda uses Driver API
- GstHip is based on the Runtime API
  - Since HIP has no Drive API
- Minor differences mostly in context/device management

Unified Backend Handling

- HIP SDK supports both AMD & NVIDIA but requires build-time selection
- Single GStreamer binary supports both backends
  - Dynamic HIP / CUDA library loading at runtime
- No HIP / CUDA SDK required during build
- GstHip provides unified wrappers for native HIP API calls
  - Native HIP API: `hipInit(unsigned int flags)`
  - Wrapper API: `HipInit(GstHipVendor vendor, unsigned int flags)`

GstHip Public Library

- Provides core objects for resource sharing between plugins and app

- **GstHipStream**: HIP stream abstraction
- **GstHipEvent**: synchronization primitive (like fence)
- **GstHIPDevice**: represents backend device
- **GstHipMemory**: HIP device memory abstraction

GstHip Plugin

- Functionally similar to nvcodec plugin – excluding codec part

- hipupload / hipdownload
  - Transfer between system ↔ GPU and HIP ↔ CUDA memory
- hipconvert / hipscale / hipconvertscale
  - Color conversion and scaling via GPU compute
- hipcompositor
  - Video composition directly on GPU

Future Work

- GstHip mirrors GstCuda design and already supports both HIP & CUDA backends in one binary
- Codec support is main missing feature (HIP = compute-only)
  - **NVIDIA**: plan to extend nvcodec with GstHIP integration
  - **AMD**: add Vulkan interop for decoder / encoder pipelines
- **Inference**: integrate with ONNX plugin for unified GPU + AI workflows

Questions?

Max Campbell
Firmware Engineer
@ Veo Technologies ApS

chaos.social/@0m_ax
Matrix: @max:0m.ax

Seungha Yang
GStreamer developer
@ Centricular