# Status of Vulkan on Raspberry Pi

**XDC  2022**

Iago Toral Quiroga <itoral@igalia.com>

# Contents

# Current Status

- Vulkan 1.1 conformance (October 2021).
- Vulkan 1.2 conformance (July 2022).
- Implemented 35 extensions (core and optional).
- New multisync kernel interface (thanks to Melissa Wen!)
- Improved performance (mostly compiler & sync improvements)
- Using common synchronization infrastructure in Mesa (thanks to Jason Ekstrand!).
- Android support (thanks to Roman Stratiienko!).

## Development Highlights

## Synchronization

- Single-sync and Multi-sync paths available.
  - Undesirable, single-sync to be deprecated in the future.
  - Multi-sync is more flexible and a better match for Vulkan.

- Ported to use common synchronization framework in Mesa.
  - Gained (emulated) timeline semaphores for free in the process!
  - Dropped driver submit threading in favor of threaded submit mode.
  - Incidentally, found about CPU throttling issues with drmSyncObjWait.

- Reworked barriers to better match Vulkan semantics and optimize some cases.
  - Tried harder to skip binning syncs in favor of render syncs when possible.

# nir_address_format_2x32bit_global

- Needed for VK_KHR_buffer_device_address.
- Existing nir_address_format_{64|32}bit_global not useful for us.
  - The 64bit version will inject 64-bit cast and pack/unpack instructions.
  - The 32bit version won't match Vulkan's explicit 64-bit addresses.
- The new format can honor Vulkan's 64-bit semantics without requiring 64-bit instructions.

# Double-Buffer Mode

- Help hide tile store latency at the expense of reducing tile size.
  - May cause additional shader invocations in the geometry part of the pipeline though.
  - Needs heuristics to decide when to enable.

- Experimental feature, enable via *V3D_DEBUG=db*.
  - We probably need to tune the heuristics further.
  - We may want to port to GL driver to have a larger testing ground.

## Double-Buffer Mode

- Some workloads improved:
  - Serious Sam: +7.95% fps.
  - Quake: +6.32% fps.
  - Quake3e: +4.03% fps.
  - RbDoom3: +3.59% fps.

## Compiler

- Stopped being so aggressive trying to hide latency.
  - Thread switching and TMU pipelining are quite effective at hiding latency already.
  - Over-estimating latency can delay critical paths in shaders.
  - Slight performance improvement in pretty much all workloads we tested.

- Stopped rebuilding interference graph after each spill.
  - Massive improvement for compile times in spilling shaders.
  - We still recompute liveness so we don’t hurt spilling quality.
  - We use this mostly to reduce spilling by testing multiple compile strategies.

## Pain points

## CPU jobs

- A few things in a command buffer may need CPU intervention.
- Threaded submit helps a bit.
- This is the reason we cannot reliably support SYNC_FD exports.
- I hope we can find ways to get rid of this in the future.

# fp16

- Hardware design based on emitting 2x16-bit instructions.
- Significant register allocation constraints.
- Difficult to exploit optimally in practice.
- We do support VK_KHR_16bit_storage though.

## Zink

- It requires VK_EXT_scalar_block_layout.
  - We cannot implement this optimally due to hardware restrictions.

- Alternatively, we could lower all load/store to scalar.
  - Not great for performance though.
  - … but may be worthwhile to expand testing grounds for Vulkan on Raspberry Pi.
  - Maybe expose this feature under a V3D_DEBUG setting?

# Q&A

We’re hiring: https://www.igalia.com/jobs igalia