ARB_gl_spirv: bringing SPIR-V to Mesa OpenGL

Alejandro Piñeiro (apinheiro@igalia.com)g

igalia

FOSDEM 2018

Some news/announcement first

XDC 2018 dates confirmed

- Will be held in A Coruña, Spain
- From September 26th to September 29th
- Follow www.twitter.com/xdc2018 for updates

Conformance

- Intel Mesa driver for Linux is now OpenGL 4.6 conformant

- Conformant on day one!

- Thats includes ARB_gl_spirv tests

Topics covered

- Introduction
- Development history
- Technical decisions
- Testing
- Current status and future

Introduction

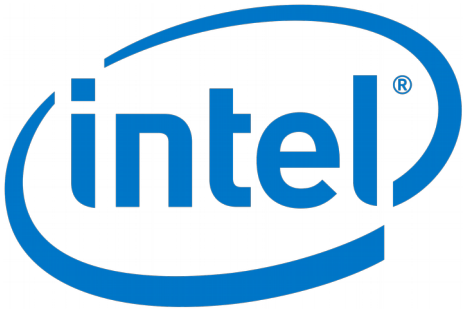
Who is doing this?

- Started by Nicolai Hähnle

- Right now:
  - Alejandro Piñeiro
  - Eduardo Lima
  - Neil Roberts

igalia

- Supported by Intel



GLSL

- Open**GL** **S**hading **L**anguage

- C-Like language used to write shaders.

- The shading source code is included on your program, so it is easy to get it back

- First anounced on 2004

SPIR-V

- Introduced as SPIR in 2011
  - Standard Portable Intermediate Representation
  - OpenCL
  - Binary format
  - Based on LLVM IR
- SPIR-V announced in 2015
  - Part of OpenCL 2.1 core and Vulkan core
  - Not based on LLVM IR anymore

OpenGL vs Vulkan

- Some applications are porting from one to the other, or want to support both

- Some interoperatibility are desired

- But one use GLSL, the other SPIR-V as shading language

GL_KHR_vulkan_glsl

- Modifies GLSL to be used for Vulkan (dec-2015)

- But not as a direct consumer, but after being compiled down to SPIR-V

- Not a driver extension, but a frontend extension

GL_ARB_gl_spirv

- Defines two things:
  - Allows SPIR-V module to be loaded on OpenGL
  - Modifies GLSL to be a source for creating SPIR-V modules for OpenGL consumption.

- Driver + frontend extension

GL_ARB_spirv_extensions

- ARB_gl_spirv is focused on SPIR-V 1.0

- This extension allows to expose which extra functionality is supported.

- Spec implies the possibility of OpenGL specific SPIR-V extensions.

Khronos tools

- Glslang
  - Khronos reference front-end for GLSL and ESSL
  - Sample SPIR-V generator

- SPIRV-tools
  - Set of tools for processing SPIR-V modules
  - Assembler, disassembler, validator, and optimizer

Base GLSL

- KHR_vulkan_glsl remove and changes several old-glsl features, and adds some vulkan-like features

- ARB_gl_spirv uses KHR_vulkan_glsl as base

- But it restores some GLSL features, removes some vulkan features, add specific ones and tweak existing ones.

Examples

- Subroutines: removed on both

- Atomic counters:
  - Removed on KHR_vulkan_glsl
  - Re-added for ARB_gl_spirv

Perfect strangers

- ARB_gl_spirv big change is having names as optional, in a SPIR-V like fashion

- Example: Frontend could get a GLSL with a ubo, and ignore the name when creating the SPIR-V

- That means that everything needs to work without any name
  - You need to use location, binding, index, etc

Development history

Pre-history

- “Interest in GL_ARB_gl_spirv” (2016-07-27)
  - Several driver developers added suggestions on how to implement it

- “[RFC] ARB_gl_spirv and NIR backend for radeonsi” (Nicolai Hähnle, 2017-05-21)
  - Starting point with some code, focused on radeonsi
  - Also starts the discussion for testing

Jumping in

- Igalia jumped in on ~September 2017

- Used Nicolai wip code as reference
  - Both mesa and piglit

- Focused first on integrate it with the mesa driver and check whats missing to get the (also wip) CTS tests passing

NIR

- Intermediate Representation of the shader, created initially for Intel, used now on other backends

- So there is a GLSL→GLSL IR→NIR→Intel IR chain on Mesa Intel drivers

- Intel Vulkan driver introduced a SPIR-V to NIR pass

But

- Right now there is not linking on NIR
- Linking is done on Mesa IR
- NIR receives all the objects already linked

What it is a linker?

- “Program that take two or more objects generated by the compiler and links them to create a executable program”

- Abstracting *a lot*, the GLSL linker does:
  - Gather info from all the objects
  - Validates that all together makes sense

Reusing IR linker

- GLSL IR linker is heavily based on the ir-variables

- Nicolai Hähnle first approach was:
  - Use existing vulkan spir to nir
  - Convert nir variables to ir variable
  - Re-use as much possible IR linker
  - Use nir shader after that

- Good approach for bootstrap

Coding and technical decisions

First steps

- Initial focus was getting the CTS tests working for the Intel Mesa driver

- That gave us a better understanding of what was missing

More spirv to nir

- Current spirv_to_nir pass was focused on Vulkan

- Missed support for several features needed by OpenGL, supported on SPIR-V:
  - Atomic counters
  - Transform feedback/geometry streams
  - Tessellation
  - OpenGL-friendly ubos/ssbos tweaking

Starting to rethink linking

- IR-variable→nir variable approach was good to get some support quickly supported
  - Example: atomic counters

- But seemed somewhat artificial

- *But* the spec tweaked too much GLSL needs, specially when linking

Poster boy: ubos

- GLSL IR linking code for ubos is based on the name.

```
/* This hash table will track all of the uniform blocks that have been
 * encountered.  Since blocks with the same block-name must be the same,
 * the hash is organized by block-name.
 */
```

- Explicit binding is optional.

- Without explicit binding, it is assigned during the linking

Poster boy: ubos (II)

- Under ARB_gl_spirv names are optional
  - Needs to work without them
- Explicit binding is mandatory
- Shared and packed layout are not supported
  - Only st140 possible (all ubos are active)
- Not too much GLSL IR linker to reuse here

Big decision going

- We need to rewrite a good bunch of the linker
  - It is really worth to over-complicate an already existing linker?

- All the info is already on the nir shader

- Timothy Arceri was adding some linking-related nir helpers

NIR based linker

- Listing all the reasons:
  - What we have right is NIR
  - Linking would be already different on several aspects
  - People were already adding nir-based linking utilities
- Scope defined:
  - It will be initially centered on ARB_gl_spirv

Focusing on passing CTS

- With a clear dev plan, we focus on getting the CTS tests passing
  - Clearly they covered most of the spec
  - They weren’t too many (8)

- Tricky: they were also a WIP at the moment
  - We used some of our time testing, reviewing, submitting feedback, and even fixes
  - The patchset reached v21!

We got it!

- We got all the tests passing ~Oct/Nov

- Next step was cleaning, and start to submit patches (more on this later)

- Clean enough for the CTS submission.

- So we are done yet? Not really ...

Testing

More testing needed

- Passing CTS is not enough to be considered production ready

- There are several aspects, especially execution tests, that needs more coverage

- Two main approaches:
  - Improve piglit
  - Work on a GLSL→SPIR-V backend

piglit

- Piglit is an open-source test suite for OpenGL implementation

- Heavily used by piglit developers

- shader_runner: run .shader_text txt file format:
  - shader source
  - Values for the uniforms, ubos, ssbos, etc
  - Check if linking or rendering was correct.

piglit – gl_spirv support (I)

- Nicolai added support for ARB_gl_spirv

- New script that parses .shader_test and call glslang to create the SPIR-V binaries
  - It includes ad-hoc attempts to “fix” the shader

- Support on shader_runner to load a SPIR-V binary or include a SPIR-V text format and use spirv-tools

piglit – gl_spirv support (II)

- You can easily switch using GLSL or SPIR-V for the same test

- You can write tests easily

- Invaluable tool at this stage

- We loved it!
  - Thanks

piglit – gl_spirv support(III)

- We added some features:
  - Feed ubo support without using names
  - To test SPIR-V execution testing without no names

- In any case, it is not clear if all this will go upstream
  - Some doubs on mesa-dev for the approach
  - Some see glslang dependency as a no-go

GLSL to SPIR-V backend (I)

- Suggested by Jason on “Adding a SPIR-V back-end to the GLSL compiler” (Jason Ekstrand, 2017-05-26). Main Advantages:
  - Provide another GLSL to SPIR-V compiler
  - Optimizations
  - ARB_gl_spirv testing

GLSL to SPIR-V backend (II)

- “The first of the real SPIR-V work” (Ian Romanick, v1 2017-10-11, v2 2017-11-21)
  - First version of the back-end
  - Some patches reviewed
  - Some features pending (like ubos)

GLSL to SPIR-V backend(III)

- For ARB_gl_spirv the idea would do this:
  - GLSL→GLSL IR→SPIR-V→NIR
  - Conditionally.
  - Internally it would need to do the “fixing”
- Would allow to run piglit tests without changes
- But:
  - Not finished
  - We would still need to modify how to feed data

Current status and future

What’s working

- A little of everything is partially covered:
  - Uniforms
  - Atomic counters
  - UBOs and SSBOs
  - Tessellation shaders
  - Transform feedback/Geometry streams

- We have plenty of programs working
- ARB_spirv_extensions fully complete

What’s missing

- Poulish all the previous features
- Arrays of arrays
  - Some support for ubos on the linker
  - Failing on the spirv to nir pass
- Multisample Image Array
- Validation
- **More testing**

Upstreaming

- Right now our mesa development branch has ~80 patches

- Plan is sending them in small batches

- We already sent a first patchset
  - “Initial gl_spirv and spirv_extensions support in Mesa and i965” (Eduardo Lima, 2017-11-17)
  - Partly reviewed. V4 sent in January.

Questions?