Improving Khronos CTS with Mesa code coverage

Samuel Iglesias Gonsálvez
siglesias@igalia.com

igalia

X.Org Developers Conference 2020

Khronos CTS

- Open-source conformance test suite developed by the Khronos Group.
  - Mandatory to get OpenGL and/or Vulkan conformance.
- Also useful for driver development.
  - Testing implementation of new extensions, features...
  - Or to detect regressions!
- In some cases, this is the only testing suite with tests for specific extensions or features.

Khronos CTS

- Test coverage is very important!

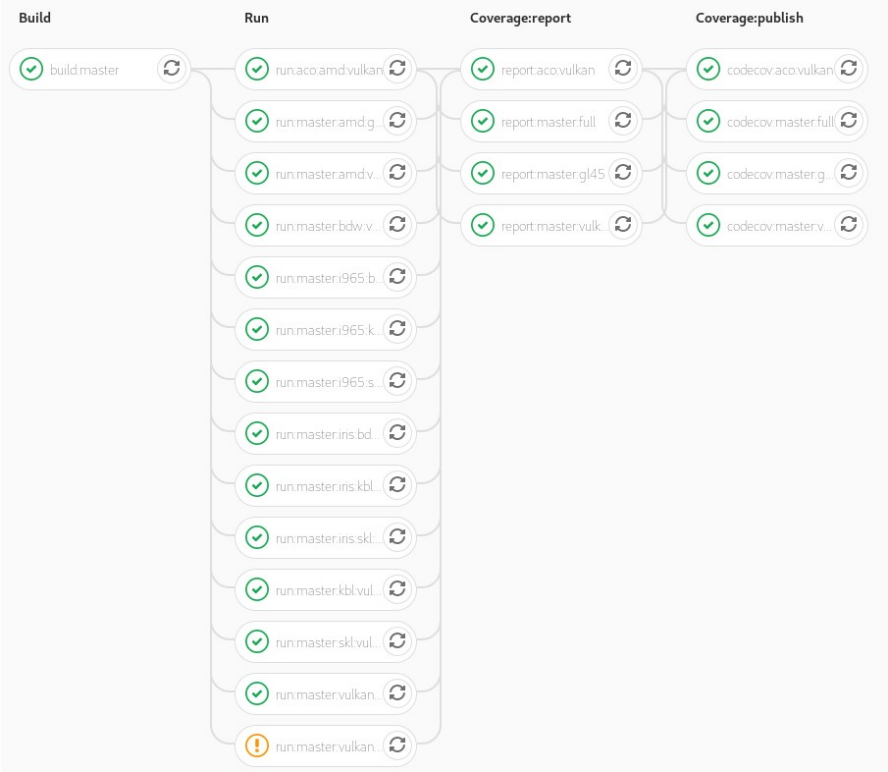
- There are several ways to come up with new tests:
  - Traditional test development.
  - Driver developers realized that there is missing test coverage.
  - Other approaches like Google’s graphicsfuzz project.

- In this talk, we are going to present another way to find missing coverage to improve existing tests or to add new ones.

Setup

- Based on our internal **GitLab CI** and **Docker** setup.
- Docker images are created with the drivers to test (**ANV, RADV**) and the CTS tests to execute.
  - Build Mesa with code coverage support enabled (**GCOV**).
  - These images are run on several machines.
  - **LCOV** to generate the HTML output of the coverage results.
    - Optionally publishing the coverage results in third-party websites (codecov.io).
      - https://codecov.io/gh/Igalia/mesa/

CI pipeline run



LCOV

LCOV - code coverage report

			Hit	Total	Coverage
Current view:	top level	**Lines:**	95676	341470	28.0 %
Test:	mesa.info	**Functions:**	6736	21612	31.2 %
Date:	2019-12-04 11:16:48				

Directory	Line Coverage		Functions	
/home/mesa/mesa/_build/src/amd/compiler	0.3 %	2 / 642	2.0 %	2 / 99
/home/mesa/mesa/_build/src/amd/vulkan	86.5 %	1206 / 1394	90.9 %	10 / 11
/home/mesa/mesa/_build/src/compiler/nir	20.5 %	2983 / 14518	48.9 %	249 / 509
/home/mesa/mesa/_build/src/compiler/spirv	15.3 %	229 / 1494	21.4 %	3 / 14
/home/mesa/mesa/_build/src/egl/wayland/wayland-drm	81.8 %	27 / 33	81.8 %	9 / 11
/home/mesa/mesa/_build/src/intel/compiler	100.0 %	9 / 9	100.0 %	1 / 1
/home/mesa/mesa/_build/src/intel/genxml	28.7 %	3806 / 13252	100.0 %	22 / 22
/home/mesa/mesa/_build/src/intel/isl	5.1 %	4 / 78	50.0 %	1 / 2
/home/mesa/mesa/_build/src/intel/perf	0.0 %	0 / 112625	0.0 %	0 / 684
/home/mesa/mesa/_build/src/intel/vulkan	51.1 %	792 / 1550	5.4 %	14 / 257
/home/mesa/mesa/_build/src/mapi/glapi/gen	0.0 %	0 / 7091	0.0 %	0 / 595
/home/mesa/mesa/_build/src/mapi/shared-glapi	0.0 %	0 / 4815	0.0 %	0 / 1605
/home/mesa/mesa/_build/src/vulkan/util	0.3 %	10 / 3362	1.1 %	1 / 92
/home/mesa/mesa/include/c11	64.2 %	52 / 81	75.0 %	15 / 20
/home/mesa/mesa/include/pci_ids	1.2 %	3 / 249	-	0 / 0
/usr/include/c++/7	17.2 %	39 / 227	2.2 %	12 / 543
/usr/include/c++/7/bits	4.6 %	116 / 2512	1.0 %	50 / 5211
/usr/include/c++/7/ext	41.5 %	17 / 41	1.9 %	12 / 645
/usr/lib/gcc/x86_64-linux-gnu/7/include	35.3 %	6 / 17	-	0 / 0
/usr/lib/llvm-8/include/llvm/ADT	51.7 %	92 / 178	52.4 %	54 / 103
/usr/lib/llvm-8/include/llvm/Analysis	100.0 %	1 / 1	100.0 %	1 / 1
/usr/lib/llvm-8/include/llvm/IR	24.2 %	23 / 95	21.3 %	10 / 47
/usr/lib/llvm-8/include/llvm/Support	77.5 %	31 / 40	60.0 %	12 / 20
/usr/lib/llvm-8/include/llvm/Target	66.7 %	2 / 3	66.7 %	2 / 3
/usr/local/include	30.9 %	17 / 55	31.6 %	6 / 19
amd/addrlib/src	12.3 %	45 / 365	15.1 %	8 / 53
amd/addrlib/src/core	20.2 %	745 / 3696	23.9 %	74 / 310
amd/addrlib/src/gfx10	0.0 %	0 / 1761	0.0 %	0 / 61
amd/addrlib/src/gfx9	0.0 %	0 / 2308	0.0 %	0 / 56
amd/addrlib/src/r800	28.2 %	1154 / 4098	48.4 %	75 / 155

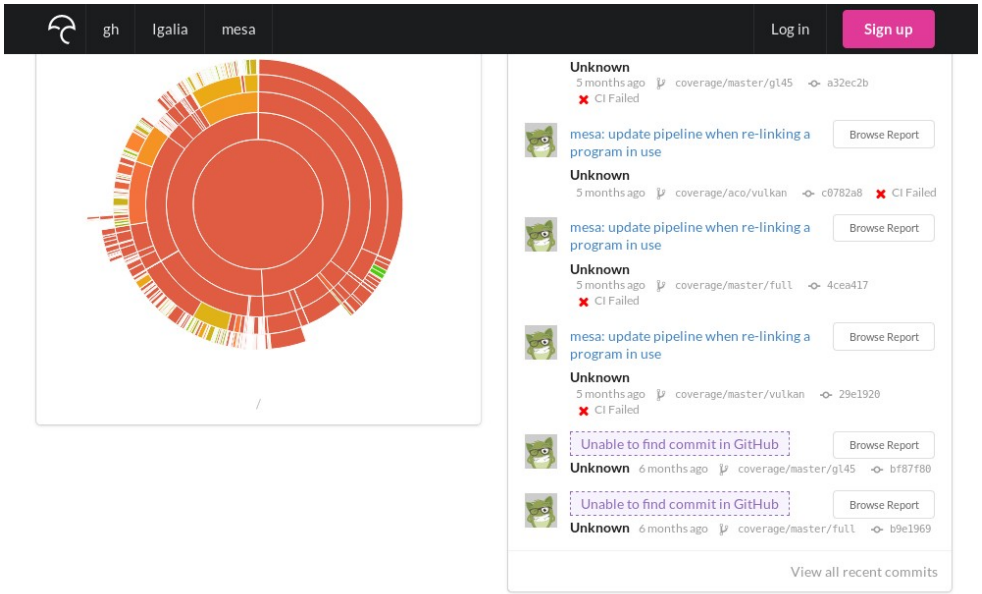
igalia

LCOV

```
6212                :
6213          30 :    case SpvOpReadClockKHR: {
6214          30 :       assert(vtn_constant_uint(b, w[3]) == SpvScopeSubgroup);
6215             :
6216             :       /* Operation supports two result types: uvec2 and uint64_t.  The NIR
6217             :        * intrinsic gives uvec2, so pack the result for the other case.
6218             :        */
6219          30 :       nir_intrinsic_instr *intrin =
6220          30 :          nir_intrinsic_instr_create(b->nb.shader, nir_intrinsic_shader_clock);
6221          30 :       nir_ssa_dest_init(&intrin->instr, &intrin->dest, 2, 32, NULL);
6222          30 :       nir_builder_instr_insert(&b->nb, &intrin->instr);
6223             :
6224          30 :       struct vtn_type *type = vtn_value(b, w[1], vtn_value_type_type)->type;
6225          30 :       const struct glsl_type *dest_type = type->type;
6226             :       nir_ssa_def *result;
6227             :
6228          30 :       if (glsl_type_is_vector(dest_type)) {
6229          18 :          assert(dest_type == glsl_vector_type(GLSL_TYPE_UINT, 2));
6230          18 :          result = &intrin->dest.ssa;
6231             :       } else {
6232          12 :          assert(glsl_type_is_scalar(dest_type));
6233          12 :          assert(glsl_get_base_type(dest_type) == GLSL_TYPE_UINT64);
6234          12 :          result = nir_pack_64_2x32(&b->nb, &intrin->dest.ssa);
6235             :       }
```

igalia

codecov.io



Files					Coverage
_build/src	176...	9,965	0	166...	5.66%
include	333	52	0	281	15.62%
src	182...	102...	0	80,...	55.96%
Project Totals (631 files)	358...	112...	0	246...	31.23%

igalia

codecov.io

```
5093  1    case SpvOpReadClockKHR: {
5094  1       assert(vtn_constant_uint(b, w[3]) == SpvScopeSubgroup);
5095
5096          /* Operation supports two result types: uvec2 and uint64_t.  The NIR
5097           * intrinsic gives uvec2, so pack the result for the other case.
5098           */
5099  1       nir_intrinsic_instr *intrin =
5100  1          nir_intrinsic_instr_create(b->nb.shader, nir_intrinsic_shader_clock);
5101  1       nir_ssa_dest_init(&intrin->instr, &intrin->dest, 2, 32, NULL);
5102  1       nir_builder_instr_insert(&b->nb, &intrin->instr);
5103
5104  1       struct vtn_type *type = vtn_value(b, w[1], vtn_value_type_type)->type;
5105  1       const struct glsl_type *dest_type = type->type;
5106          nir_ssa_def *result;
5107
5108  1       if (glsl_type_is_vector(dest_type)) {
5109  1          assert(dest_type == glsl_vector_type(GLSL_TYPE_UINT, 2));
5110  1          result = &intrin->dest.ssa;
5111          } else {
```

igalia

Issues

```
4812
4813 1     case SpvOpFragmentMaskFetchAMD:
4814       case SpvOpFragmentFetchAMD:
4815 1        vtn_handle_texture(b, opcode, w, count);
4816 1        break;
4817
4818 1     case SpvOpAtomicLoad:
4819       case SpvOpAtomicExchange:
4820       case SpvOpAtomicCompareExchange:
4821       case SpvOpAtomicCompareExchangeWeak:
4822       case SpvOpAtomicIIncrement:
4823       case SpvOpAtomicIDecrement:
4824       case SpvOpAtomicIAdd:
4825       case SpvOpAtomicISub:
4826       case SpvOpAtomicSMin:
4827       case SpvOpAtomicUMin:
4828       case SpvOpAtomicSMax:
4829       case SpvOpAtomicUMax:
4830       case SpvOpAtomicAnd:
4831       case SpvOpAtomicOr:
4832       case SpvOpAtomicXor: {
4833 1        struct vtn_value *pointer = vtn_untyped_value(b, w[3]);
4834 1        if (pointer->value_type == vtn_value_type_image_pointer) {
4835 1           vtn_handle_image(b, opcode, w, count);
4836        } else {
4837 1           vtn_assert(pointer->value_type == vtn_value_type_pointer);
4838 1           vtn_handle_atomics(b, opcode, w, count);
4839        }
4840 1        break;
4841     }
```

igalia

Issues

```
3040           :    case SpvOpAtomicLoad:
3041        3656 :       if (opcode == SpvOpAtomicLoad)
3042        2106 :       { int unused = 0; }
3043           :    case SpvOpAtomicSMin:
3044        3695 :       if (opcode == SpvOpAtomicSMin)
3045          39 :       { int unused = 0; }
3046           :    case SpvOpAtomicUMin:
3047        3731 :       if (opcode == SpvOpAtomicUMin)
3048          36 :       { int unused = 0; }
3049           :    case SpvOpAtomicSMax:
3050        3755 :       if (opcode == SpvOpAtomicSMax)
3051          24 :       { int unused = 0; }
3052           :    case SpvOpAtomicUMax:
3053        3785 :       if (opcode == SpvOpAtomicUMax)
3054          30 :       { int unused = 0; }
3055           :    case SpvOpAtomicAnd:
3056        3851 :       if (opcode == SpvOpAtomicAnd)
3057          66 :       { int unused = 0; }
3058           :    case SpvOpAtomicOr:
3059        3917 :       if (opcode == SpvOpAtomicOr)
3060          66 :       { int unused = 0; }
3061           :    case SpvOpAtomicXor:
3062        3986 :       if (opcode == SpvOpAtomicXor)
3063          69 :       { int unused = 0; }
3064        3986 :       image = *vtn_value(b, w[3], vtn_value_type_image_pointer)->image;
3065        3986 :       scope = vtn_constant_uint(b, w[4]);
3066        3986 :       semantics = vtn_constant_uint(b, w[5]);
3067        3986 :       break;
3068           :
```

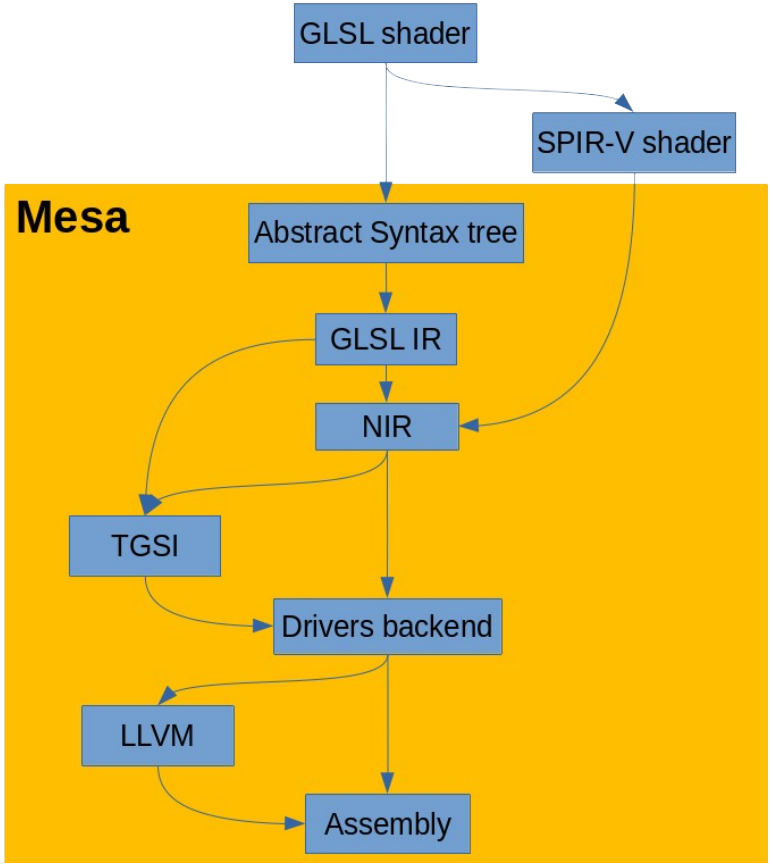
igalia

Mesa code coverage analysis

- We manually check several places.
  - Vulkan API entrypoints for each driver.

```
3486  1  void genX(CmdDraw)(
3487        VkCommandBuffer                             commandBuffer,
3488        uint32_t                                    vertexCount,
3489        uint32_t                                    instanceCount,
3490        uint32_t                                    firstVertex,
3491        uint32_t                                    firstInstance)
3492     {
3493  1     ANV_FROM_HANDLE(anv_cmd_buffer, cmd_buffer, commandBuffer);
3494  1     struct anv_graphics_pipeline *pipeline = cmd_buffer->state.gfx.pipeline;
3495  1     const struct brw_vs_prog_data *vs_prog_data = get_vs_prog_data(pipeline);
3496
3497  1     if (anv_batch_has_error(&cmd_buffer->batch))
3498           return;
3499
3500  1     genX(cmd_buffer_flush_state)(cmd_buffer);
```

Mesa shader compiler



igalia

Mesa code coverage analysis

- For SPIR-V, we look at different places:
  - **src/compiler/spirv/**
    - However, sometimes the SPIR-V opcode handling is generic enough so we are not able to see missing coverage.
  - **Driver backend compiler**.
    - In some cases, the drivers process these “generic” cases separately, so any missing coverage is easy to spot.

Example

- Thanks to **ACO**, we found that some relational operators were not tested for 64-bit data types.

```
2315 1    case nir_op_fge: {
2316 1       if (instr->src[0].src.ssa->bit_size == 32)
2317 1          emit_comparison(ctx, instr, aco_opcode::v_cmp_ge_f32, dst);
2318       else if (instr->src[0].src.ssa->bit_size == 64)
2319          emit_comparison(ctx, instr, aco_opcode::v_cmp_ge_f64, dst);
2320 1       break;
2321    }
```

Conclusions

- Setup based on open-source tools: Gitlab CI and Docker.
- CTS coverage is pretty good nowadays.
- Mesa code coverage analysis is completely manual right now :-(
- This work is complementary to **traditional test development** (like developing tests for new extensions) and **to other approaches** like Google's **graphicsfuzz** project.

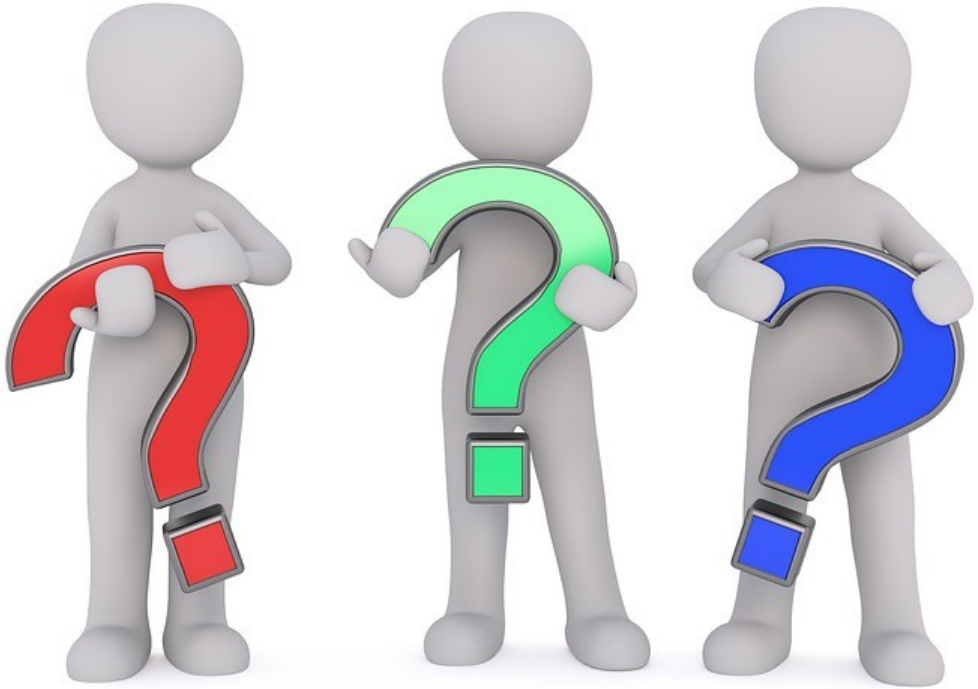
igalia

Conclusions

- We are not alone doing code coverage on drivers: Swiftshader driver started doing it some months ago.

- CTS coverage could improve a lot if all the drivers do the same and submit issues to VK-GL-CTS issue tracker.
  - This work benefits everybody!

Questions?



igalia


igalia