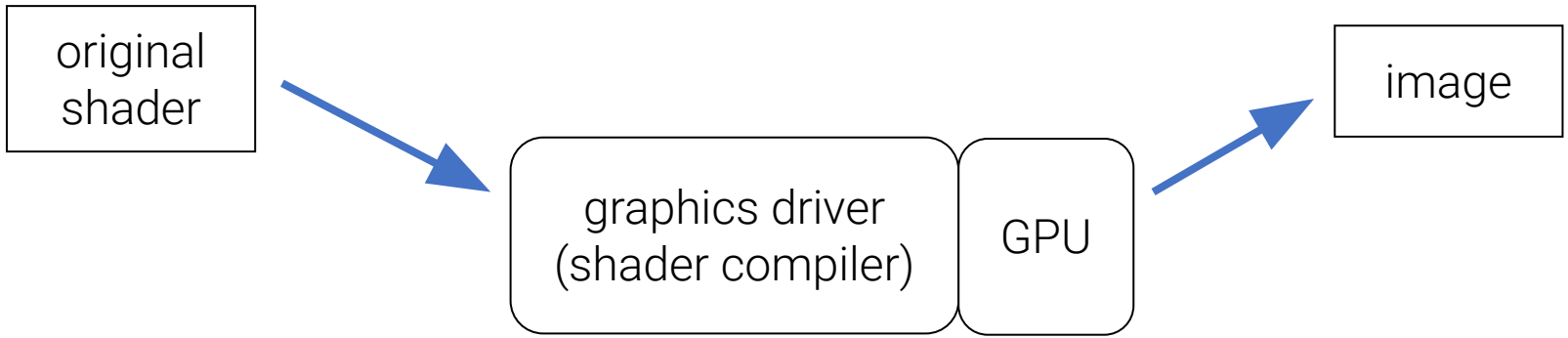
Metamorphic Testing for Graphics Compilers

Alastair F. Donaldson, Imperial College London
Hugues Evrard, Google
Paul Thomson, Google

Outline

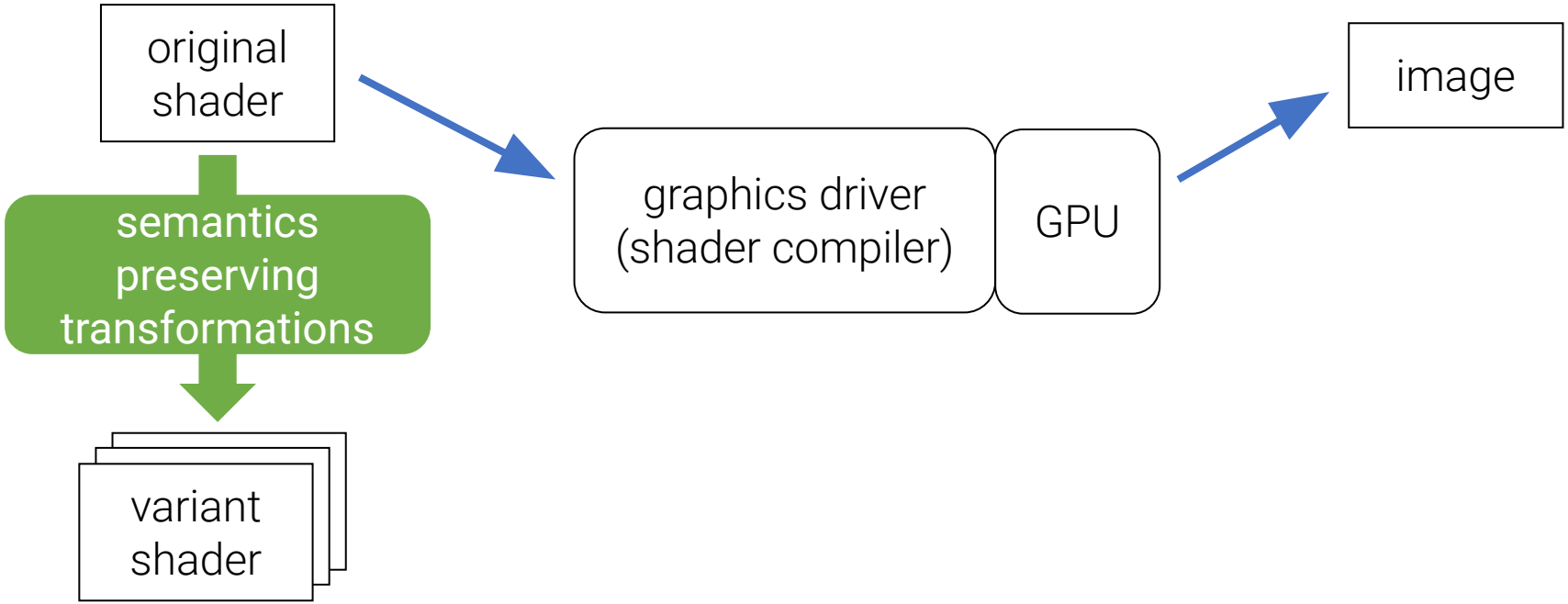
1. Overview of GraphicsFuzz
2. Growing the **Vulkan Conformance Test Suite** using GraphicsFuzz
3. Using GraphicsFuzz for **differential code coverage**
4. Finding **deeper vulnerabilities** using metamorphic testing

The GraphicsFuzz testing approach



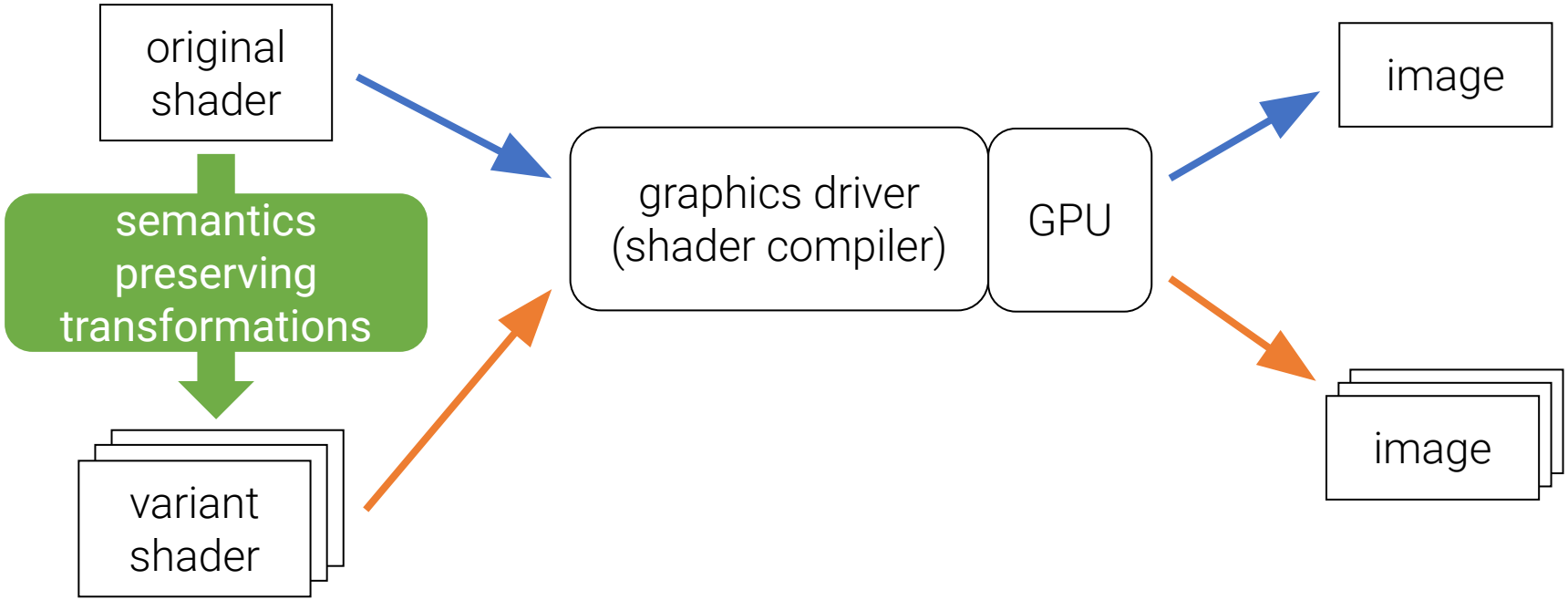
Inspired by *Equivalence Modulo Inputs* Testing (Le et al., PLDI'14)
A kind of *metamorphic* testing

The GraphicsFuzz testing approach



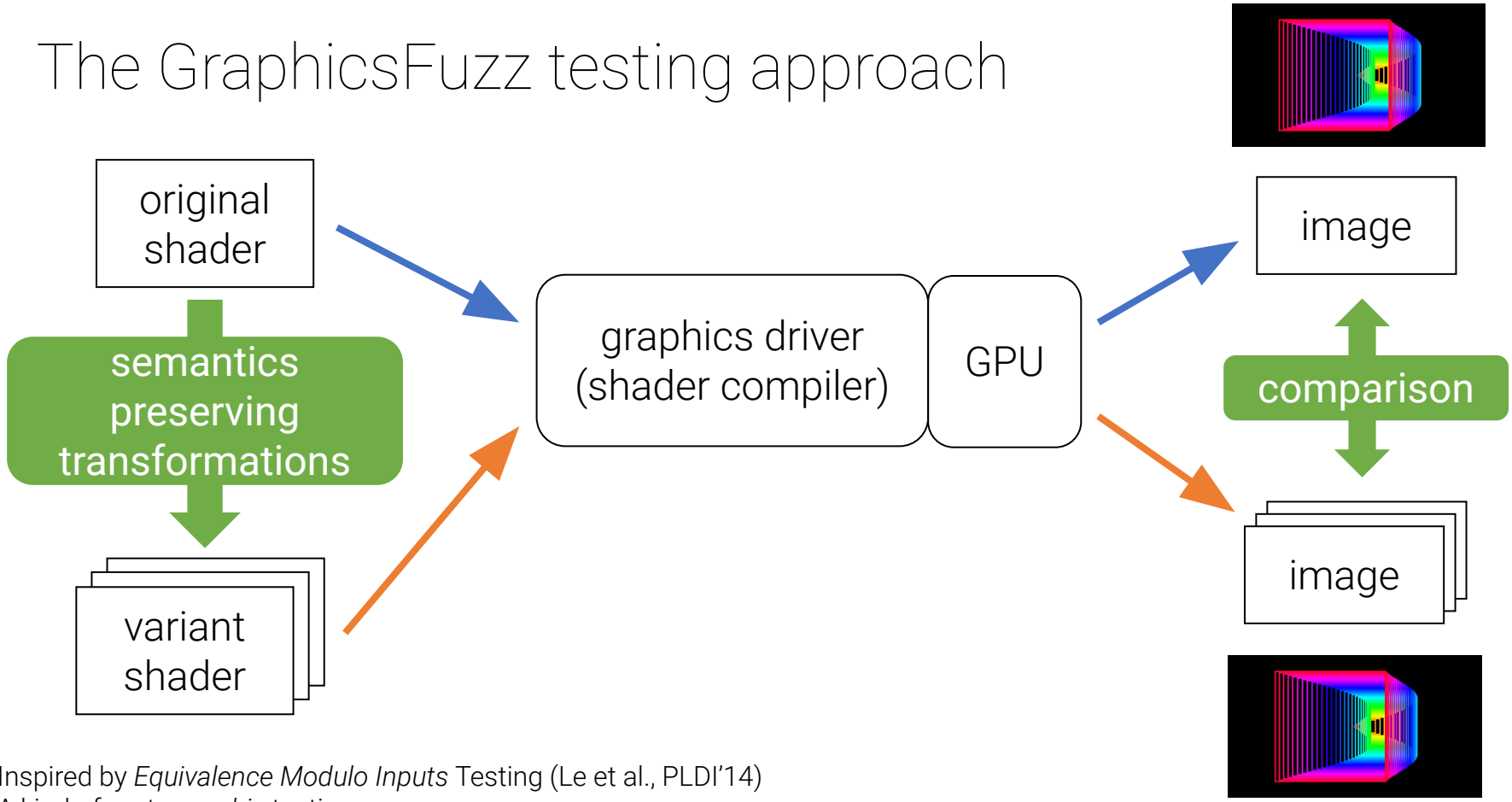
Inspired by *Equivalence Modulo Inputs* Testing (Le et al., PLDI'14)
A kind of *metamorphic* testing

The GraphicsFuzz testing approach



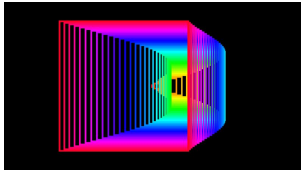
Inspired by *Equivalence Modulo Inputs* Testing (Le et al., PLDI'14)
A kind of *metamorphic* testing

The GraphicsFuzz testing approach



Inspired by *Equivalence Modulo Inputs* Testing (Le et al., PLDI'14)
A kind of *metamorphic* testing

The GraphicsFuzz testing approach



original shader

semantics preserving transformations

variant shader

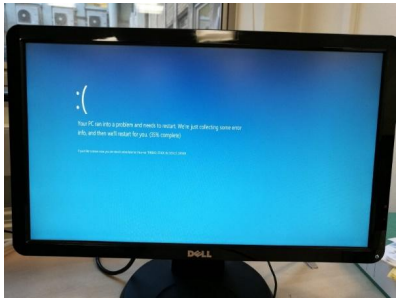
graphics driver (shader compiler)

GPU

image

diff/crash

image



Inspired by *Equivalence Modulo Inputs* Testing (Le et al., PLDI'14)
A kind of *metamorphic* testing

Example transformations

Inject dead code

```
if (false) {

    // arbitrary code

}
```

Use *uniforms* - shader inputs - to fool the compiler:

```
uniform float f; // set to 1.0 at runtime

...

if (f < 0.0) { // evaluates to false

    // arbitrary code

}
```

Example transformations

Wrap code in single iteration loop

```
// existing code
```

```
for (int i = 0; i < 1; i += 1) {

  // existing code

}
```

Again, fool the compiler with uniforms

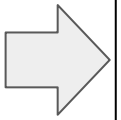
```
// existing code
```

```
uniform int a; // set to 0

uniform int b; // set to 1

uniform int c; // set to 1

for (int i = a; i < b; i += c) {

  // existing code

}
```

Example transformations

Pack scalars into vector

```
float d = 42.0;

vec2 v = vec2(1.0, 0.0)

…

d = v.x + v.y;
```

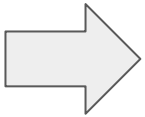


```
vec3 d_v = vec3(42.0, 1.0, 0.0);



…

d_v.x = d_v.y + d_v.z;
```

Example transformations

Add **barrier synchronization** in **compute shaders**

```
for (int i = 0; i < 10; i++) {

    // do something

    // do some more

}
```

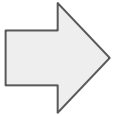


```
for (int i = 0; i < 10; i++) {

    // do something

    barrier();

    // do some more

}
```

Example transformations

Do not add barriers where control flow is **divergent**

```
for (int i = 0;
     i < gl_GlobalInvocationID.x;
     i++) {

   // do something

   // do some more

}
```

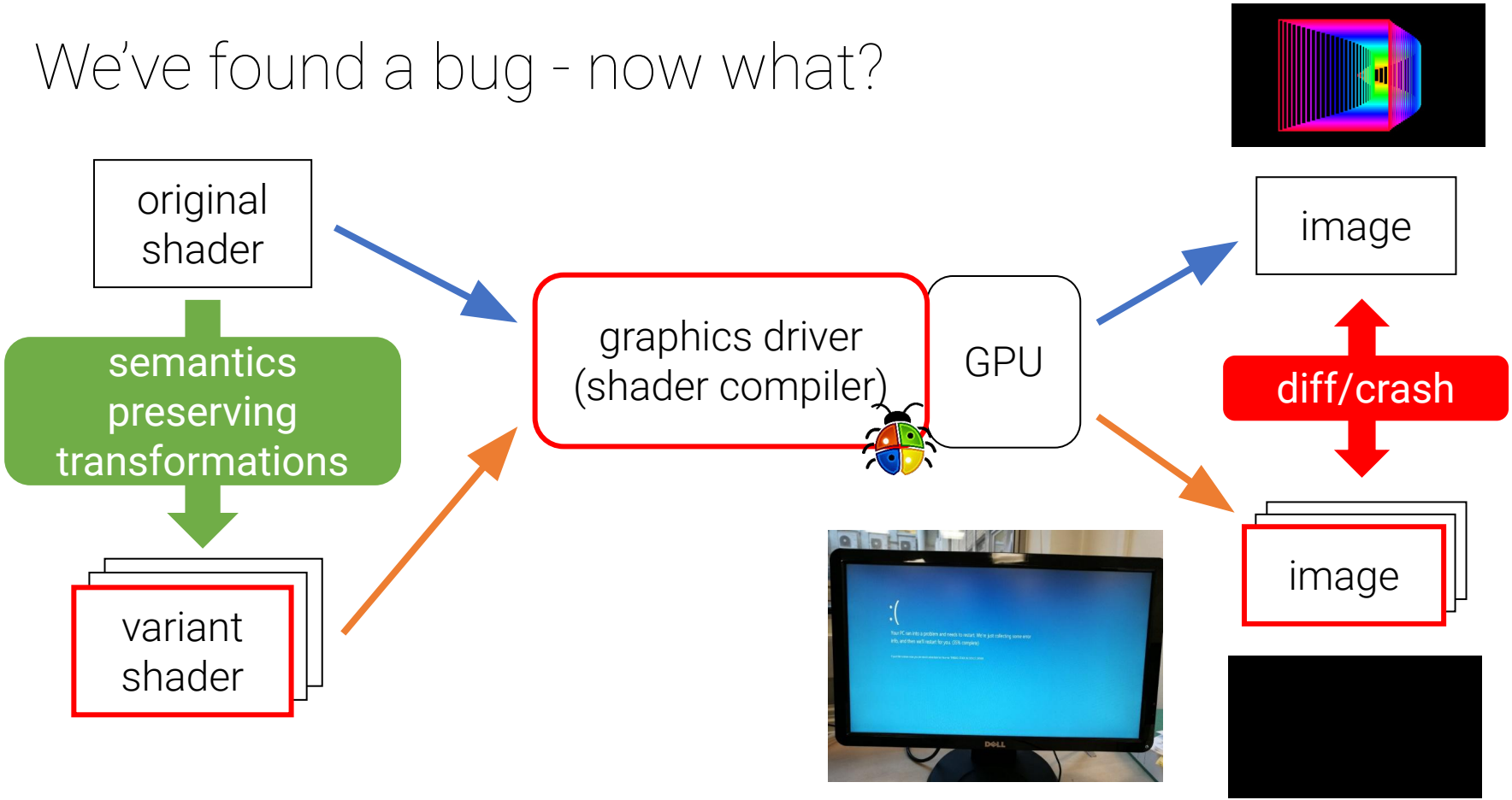


```
for (int i = 0;
     i < gl_GlobalInvocationID.x;
     i++) {

   // do something

   barrier(); // illegal

   // do some more

}
```

Concurrency and determinism

- Our metamorphic testing approach requires shaders to have *deterministic output*
- Concurrency is acceptable as long as the end result is unique
- Transformations must respect this

We've found a bug - now what?

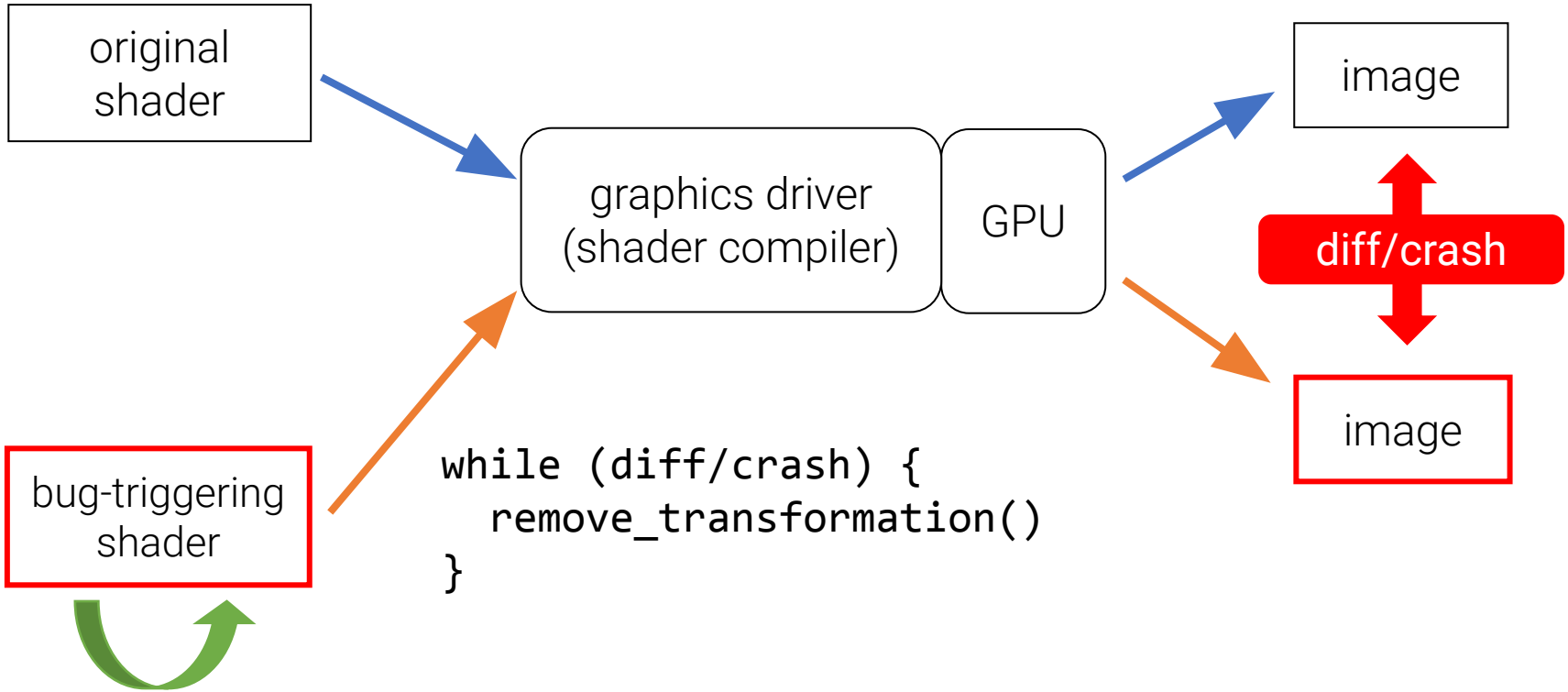


Useful?

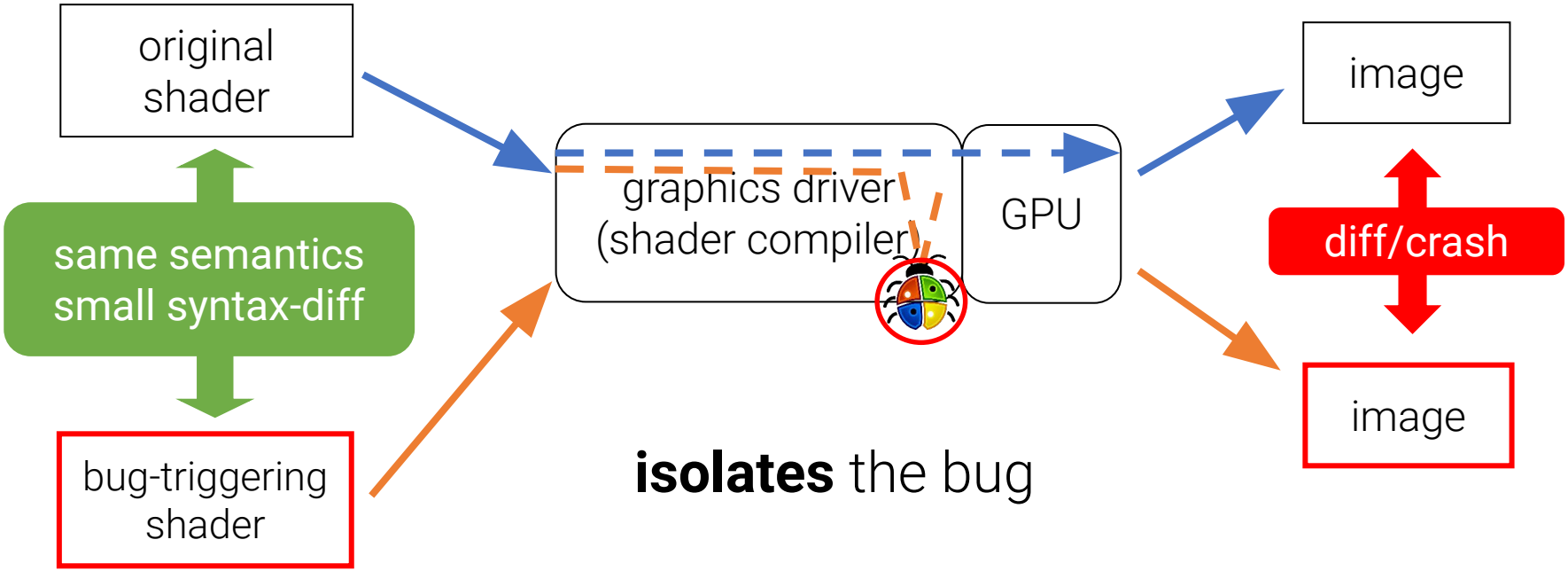
“Trust me: there’s a bug in your GPU compiler. Good luck!”

```
void main(void)
{
    vec2 uv = (gl_FragCoord.xy / resolution.xy) * 2.0 - 1.0;
    uv.x *= resolution.x / resolution.y;
    if(_GLF_DEAD(_GLF_FALSE(false, (injectionSwitch.x > injectionSwitch.y))))
        return;
    vec3 finalColor = RenderScene(uv);
    if(_GLF_DEAD(_GLF_IDENTITY(false, (false) || false)))
        {
            vec3 donor_replacementp = _GLF_FUZZED(faceforward(((++ finalColor) - faceforward(vec3(4.8, 7582.5251,
-3.4), vec3(-369.491, -9.0, 6172.7474), finalColor)), vec3(6108.1119, -181.078, 495.885), (finalColor).yzx));
            float donor_replacementtw = _GLF_FUZZED(sign(dot((EPS / vec3(53.44, 6.0, -752.725)), fract(finalColor))))
            float donor_replacementstrength = _GLF_FUZZED(38.04);
            float donor_replacementprev = _GLF_FUZZED(clamp((+ distance(time, -47.91)), (-- finalColor.g), (mouse /
EPS)[1]));
            if(_GLF_DEAD(_GLF_FALSE(false, (injectionSwitch.x > injectionSwitch.y))))
                return;
            float donor_replacementaccum = _GLF_FUZZED(distance(vec2(-349.170, -4419.3875), (- vec4(-359.006, 69.29,
-96.95, -243.116)).wz));
            if(_GLF_DEAD(_GLF_IDENTITY(false, (false ? _GLF_FUZZED((-28449 < shadowType)) : false))))
                return;
            for(
                int i = 0;
                i < 16;
```

Test-case reduction

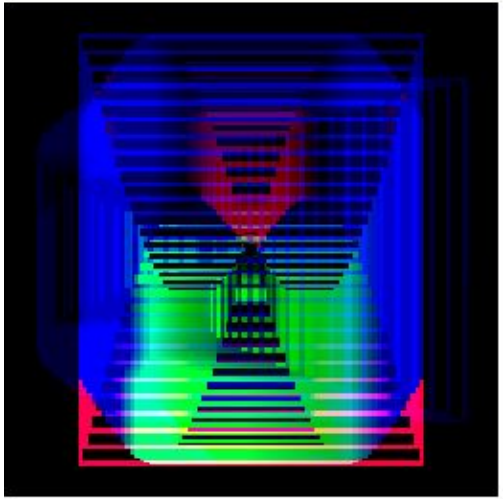


Test-case reduction

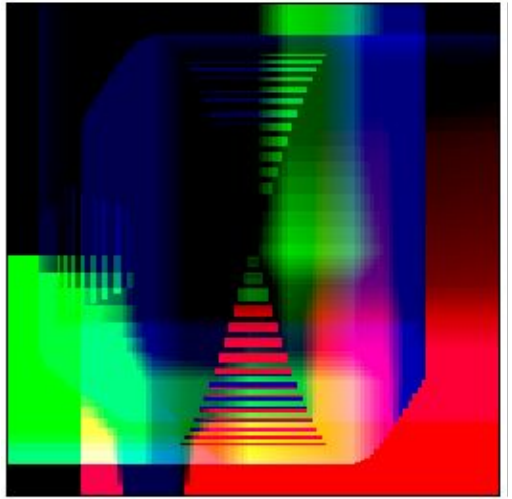


isolates the bug

A **wrong image** compiler bug



Original



Variant

Diff:

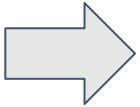
```
-   return defaultColor();
+   switch(0)
+   {
+     case 0:
+     return defaultColor();
+     break;
+   }
```

A **crash** compiler bug

```
precision highp float;

void main() {
  vec2 a = vec2(1.0);
  vec4 b = vec4(1.0);
  // Crash:
  pow(vec4(a, vec2(1.0)), b);
}
```

A **crash** compiler bug

```
precision highp float;

void main() {
  vec2 a = vec2(1.0);
  vec4 b = vec4(1.0);
  // Crash:
  pow(vec4(a, vec2(1.0)), b);
}
```



```
SIGSEGV

#00 pc 01e214  /vendor/compiler.so (LLVMGen::genPow(Operand*, Operand*))
...
```

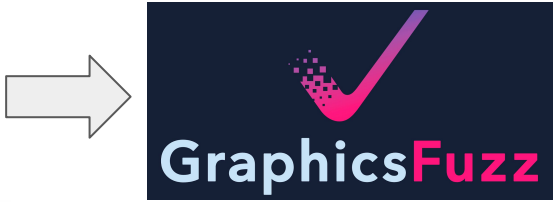
Another **crash** compiler bug

```
precision highp float;

vec3 GLF_live6mand() {
 return mix(
  uintBitsToFloat(uvec3(38730u, 63193u, 63173u)),
  floor(vec3(463.499, 4.7, 0.7)),
  vec3(1.0) + vec3(1.0)
 );
}
void main() {
 GLF_live6mand();
 _GLF_color = vec4(1.0, 0.0, 0.0, 1.0);
}
```

```
ASSERTION FAILURE

amdllpc: external/llvm/lib/Support/APFloat.cpp:1521:

llvm::lostFraction llvm::detail::IEEEFloat::
addOrSubtractSignificand(const llvm::detail::IEEEFloat &, bool):

Assertion `!carry' failed.
```

GraphicsFuzz: secure and reliable graphics drivers

Imperial College London



https://github.com/google/graphicsfuzz

When we joined Google...

The **Vulkan** API is the future of graphics

Our goal: use GraphicsFuzz to improve Vulkan shader compilers

When we joined Google...

The **Vulkan** API is the future of graphics

Our goal: use GraphicsFuzz to improve Vulkan shader compilers

Problem:

- GraphicsFuzz was designed for OpenGL, not Vulkan!

Testing Vulkan compilers via OpenGL

GLSL: the OpenGL shading language

SPIR-V: the Vulkan shading language

Testing Vulkan compilers via OpenGL

GLSL: the OpenGL shading language

SPIR-V: the Vulkan shading language

glslang: compiles GLSL into SPIR-V

Gets us into Vulkan world

Testing Vulkan compilers via OpenGL

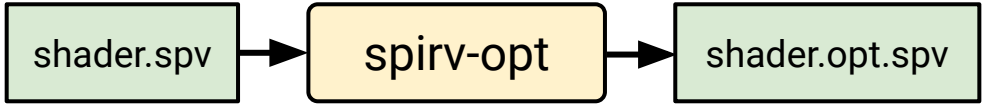
GLSL: the OpenGL shading language

SPIR-V: the Vulkan shading language

glslang: compiles GLSL into SPIR-V

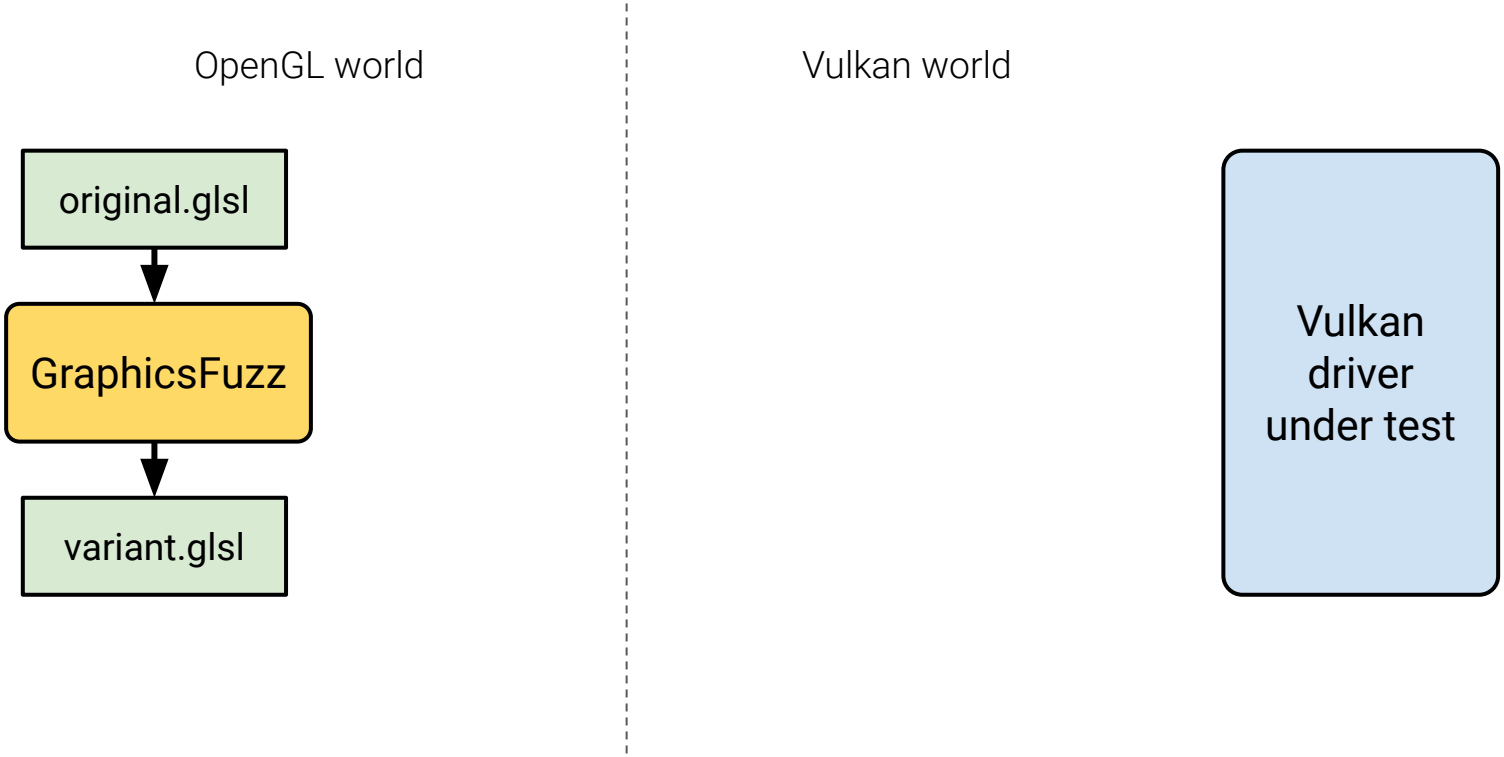
Gets us into Vulkan world

spirv-opt: source-to-source SPIR-V optimizer

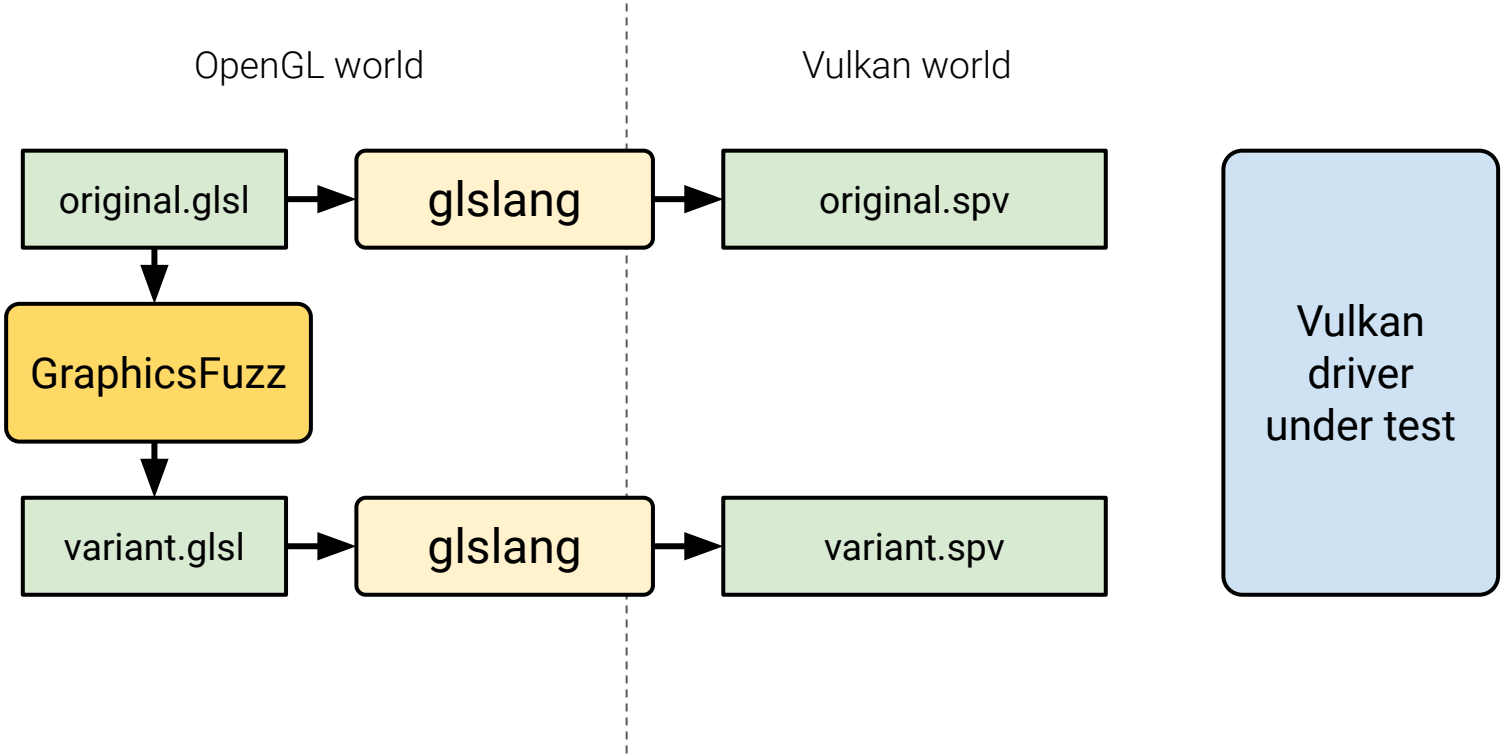


Makes Vulkan world more interesting!

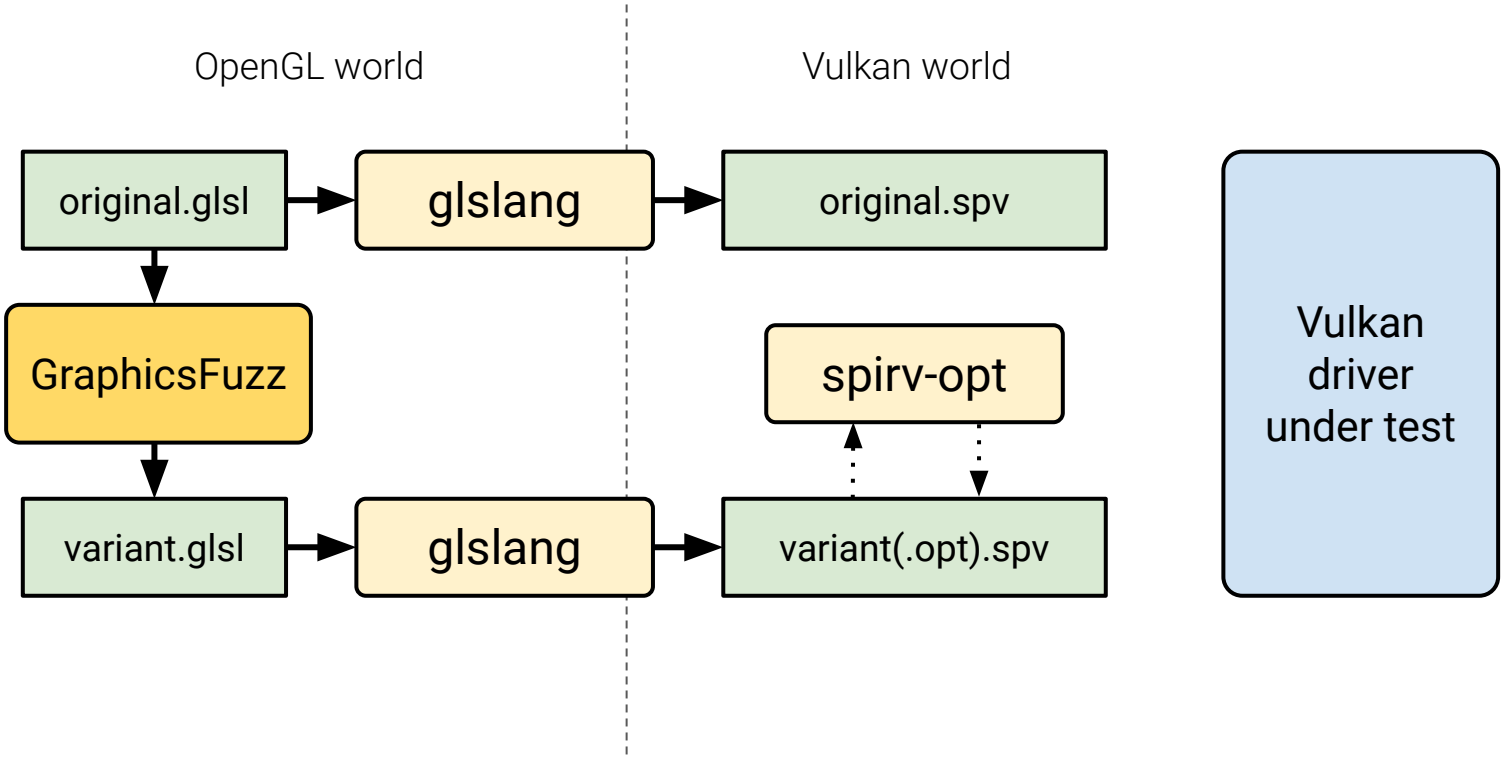
Testing Vulkan from OpenGL



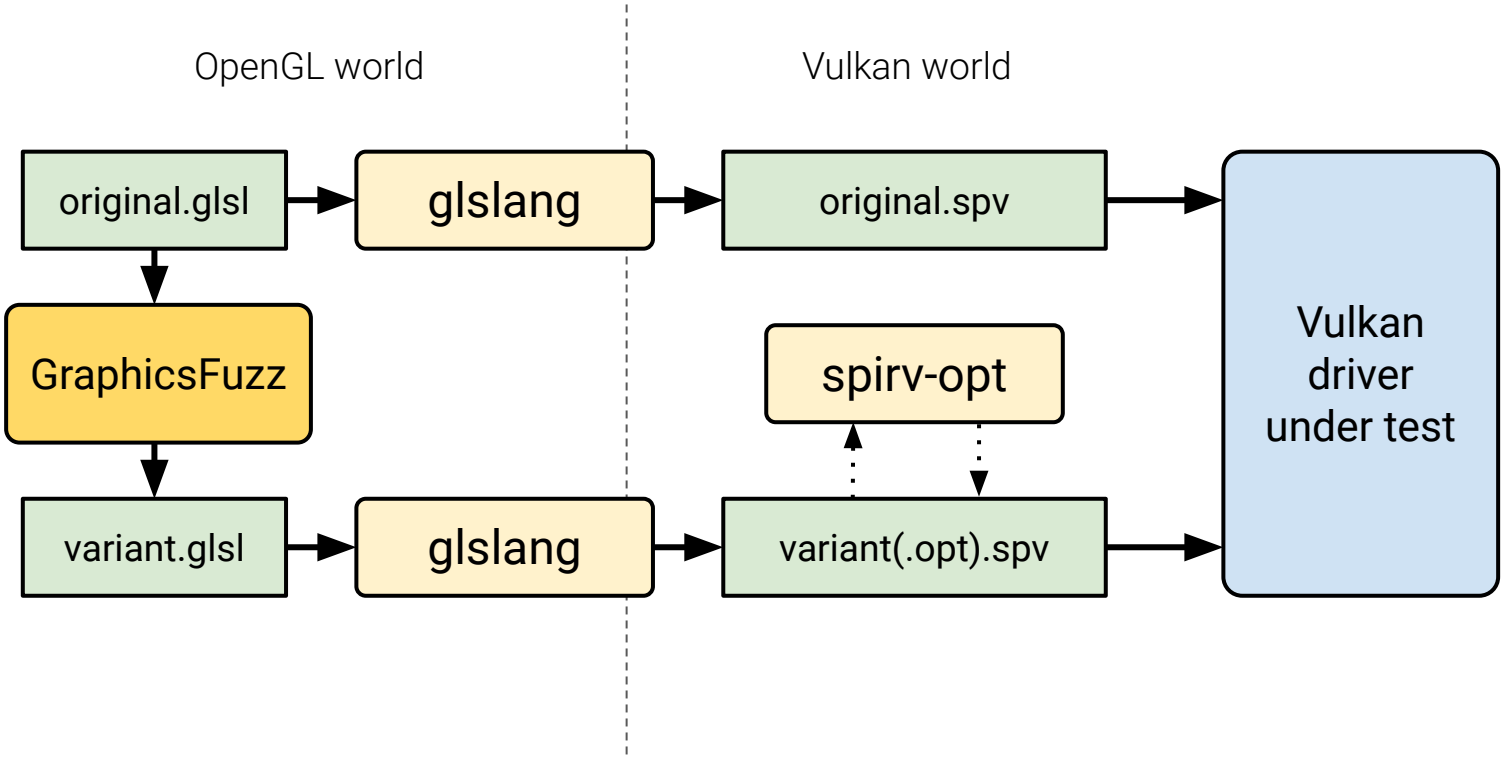
Testing Vulkan from OpenGL



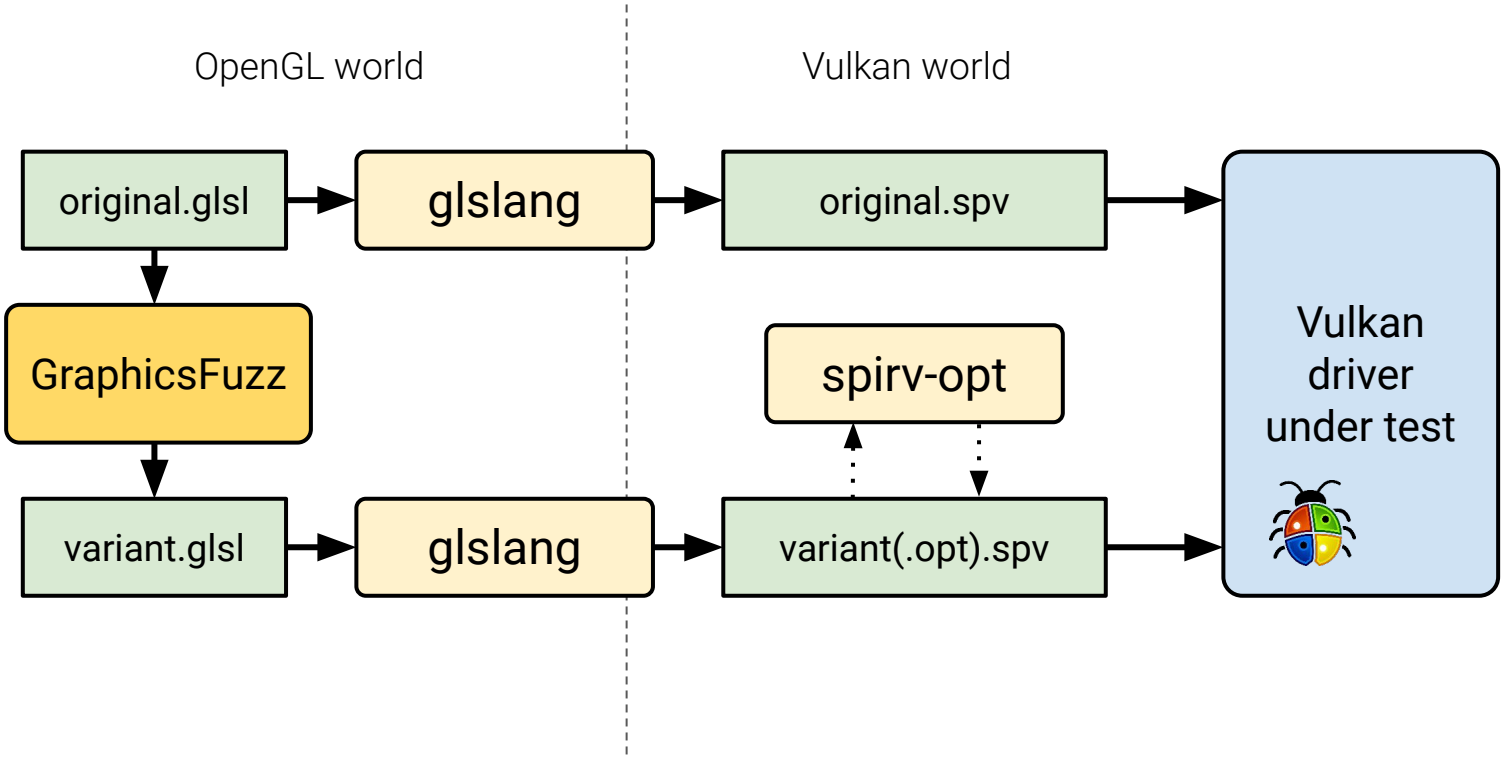
Testing Vulkan from OpenGL



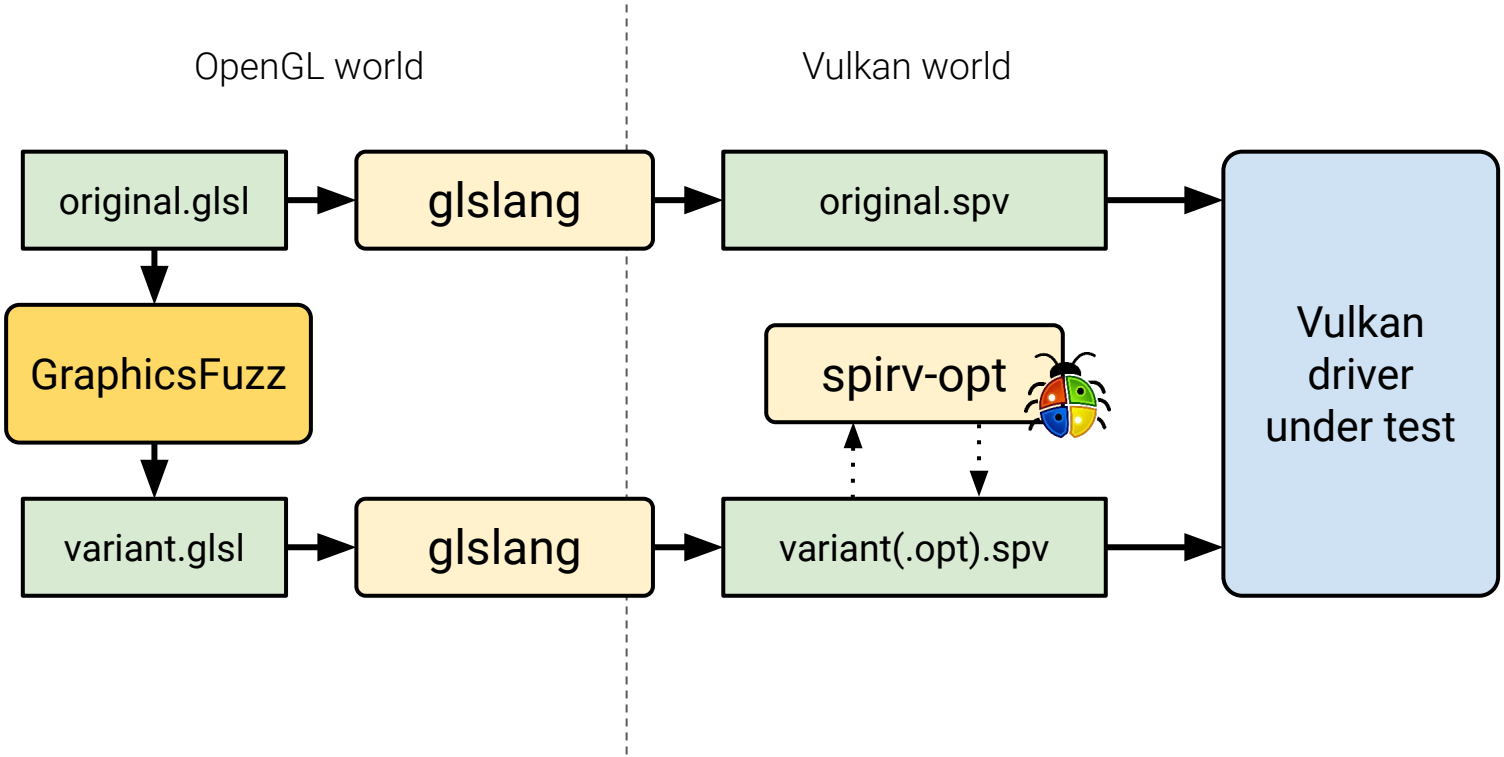
Testing Vulkan from OpenGL



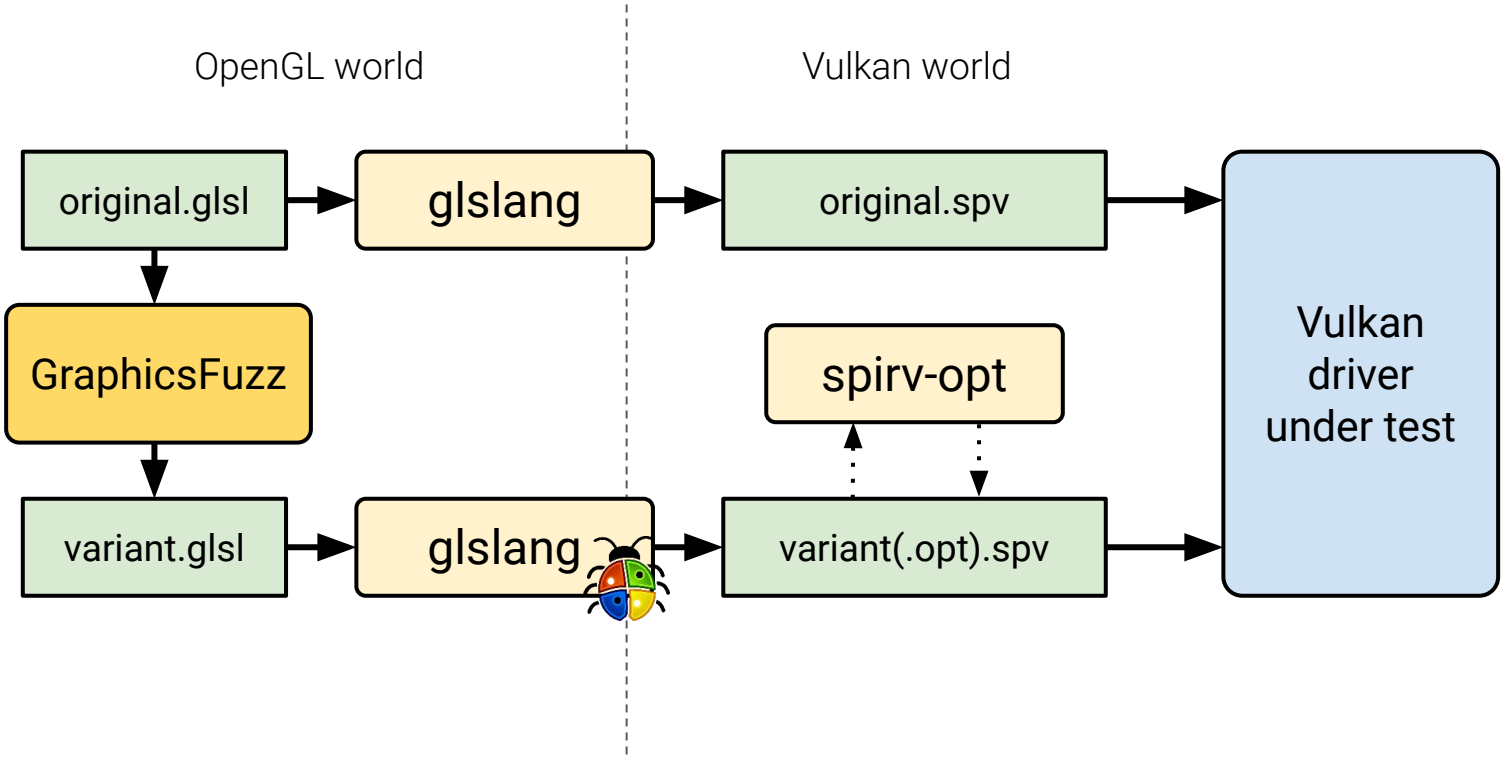
Testing Vulkan from OpenGL



Testing Vulkan from OpenGL



Testing Vulkan from OpenGL



Where Should Bug Reports Go?

Report to developer?

Limited value:

- Might not get fixed at all
- Fix might not propagate to end user devices
- No value to the rest of the ecosystem

Where Should Bug Reports Go?

Report to developer?

Limited value:

- Might not get fixed at all
- Fix might not propagate to end user devices
- No value to the rest of the ecosystem

Add to Vulkan Conformance Test Suite?

High value:

- All GPU makers run this daily
- Devices must pass CTS -> bug fixed for all future devices
- Contributes bug-inducing test to everyone

Vulkan Conformance Test Suite (CTS)

Search or jump to… / Pull requests Issues Trending Explore

KhronosGroup / **VK-GL-CTS** Watch Star 312 Fork

Code Issues 21 Pull requests 4 Actions Wiki Releases 64 ···

master **22** branches **64** tags

alegal-arm Update vk.xml revision ··· 483a71c 7 days ago **16,719** commits

.github	Add CONTRIBUTING.md for Github	3 years ago
android	Add cube compat. to array image copy tests	7 days ago
data	Add const matrix multiply test	2 years ago
doc	Add support for properly styling SPIR-V sources from …	7 months ago

About

Khronos Vulkan, OpenGL, and OpenGL ES Conformance Tests

www.khronos.org/

opengl-es vulkan opengl opengl-es-cts vulkan-cts opengl-cts

Adding a test to CTS requires care

Test outcome cannot depend on **undefined** or **implementation-defined** behaviour

Adding a test to CTS requires care

Test outcome cannot depend on **undefined** or **implementation-defined** behaviour

Worst-case scenario: an invalid test gets admitted to CTS

Adding a test to CTS requires care

Test outcome cannot depend on **undefined** or **implementation-defined** behaviour

Worst-case scenario: an invalid test gets admitted to CTS

Challenging undefined behaviours

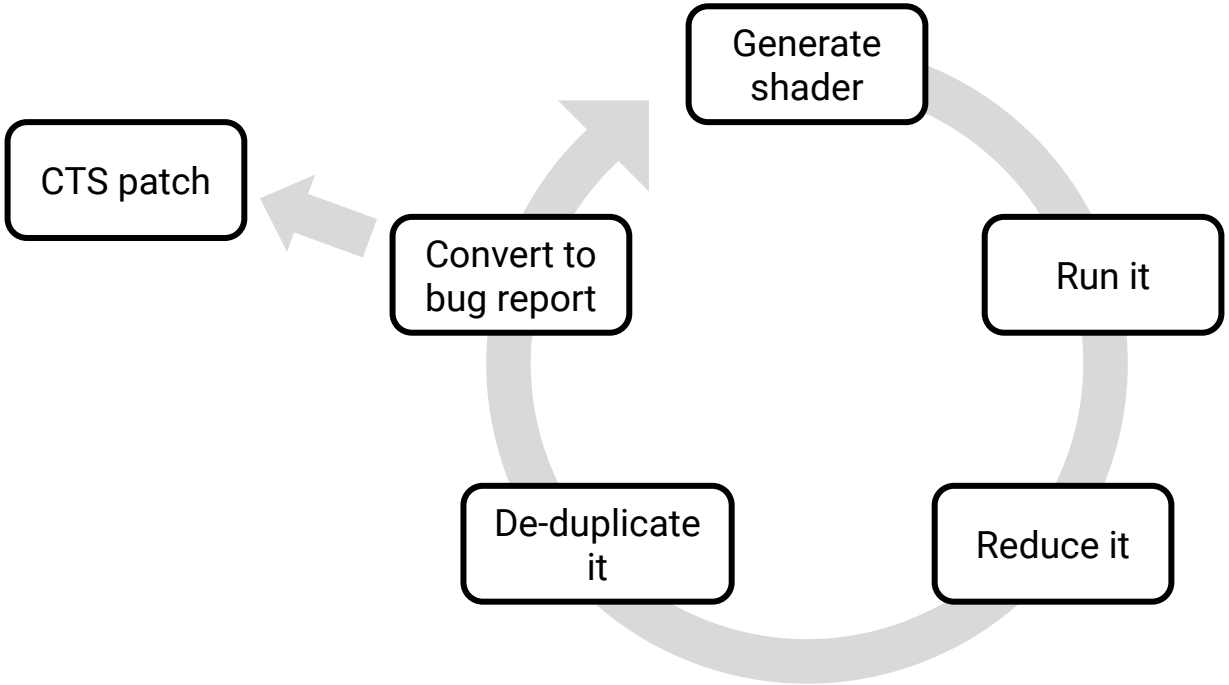
- Infinite loops
- Out-of-bounds accesses
- Uninitialized accesses

Challenging implementation-defined behaviour

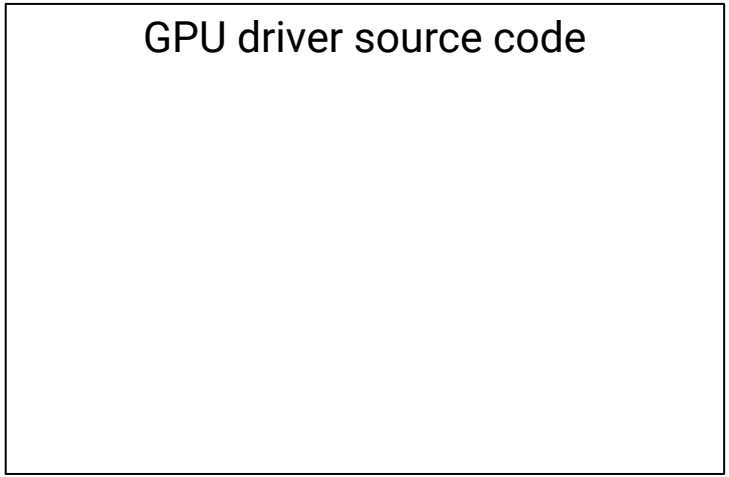
- Floating-point precision

Safeguards against these discussed in ECOOP 2020 paper

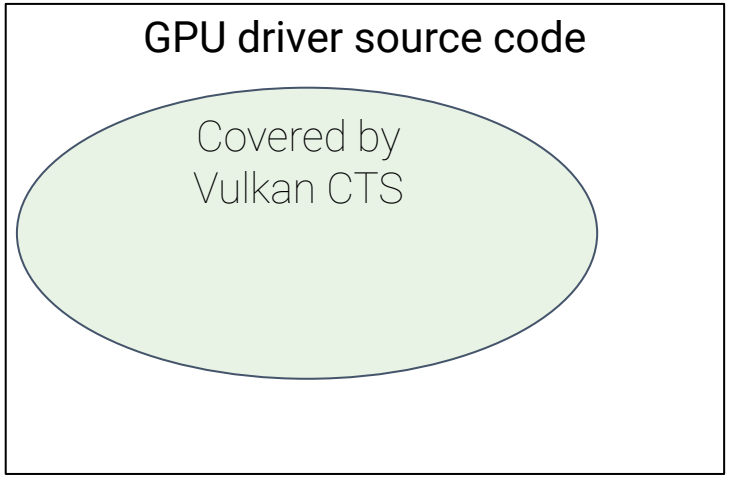
GraphicsFuzz → CTS workflow



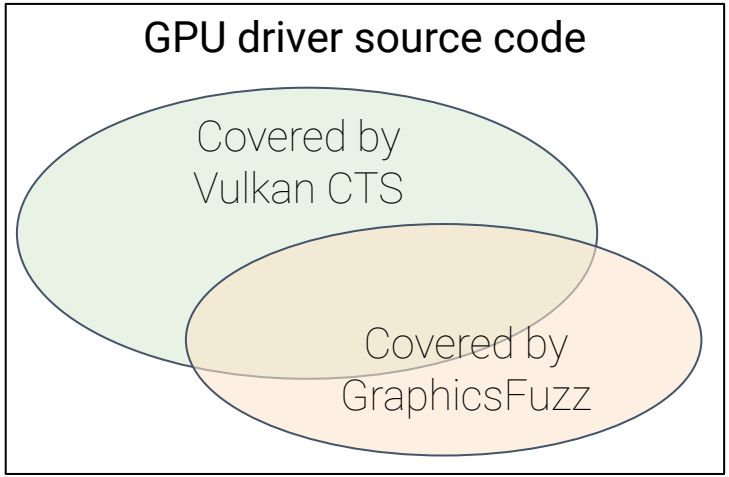
Beyond Bugs: Differential Code Coverage



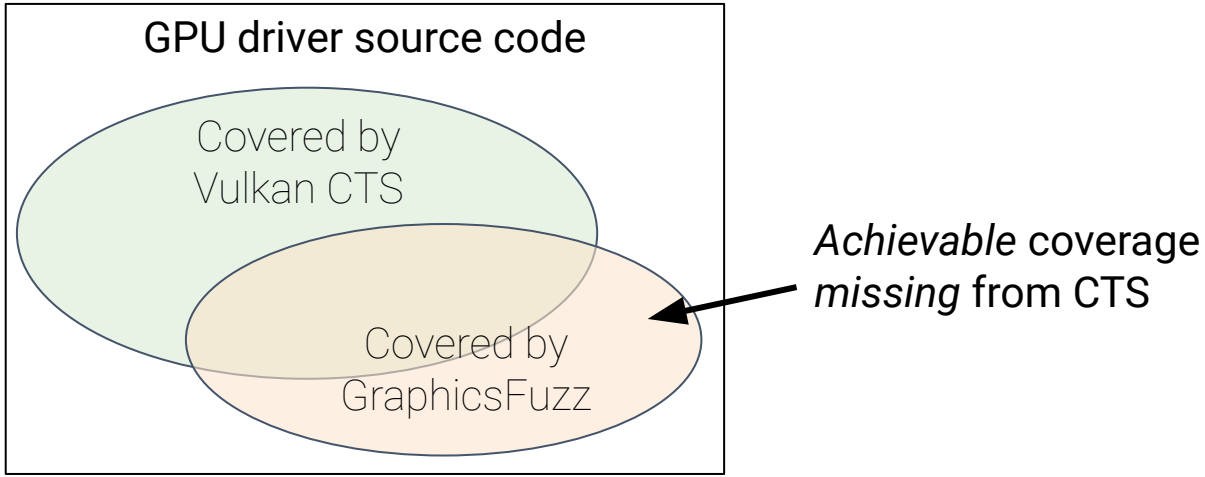
Beyond Bugs: Differential Code Coverage



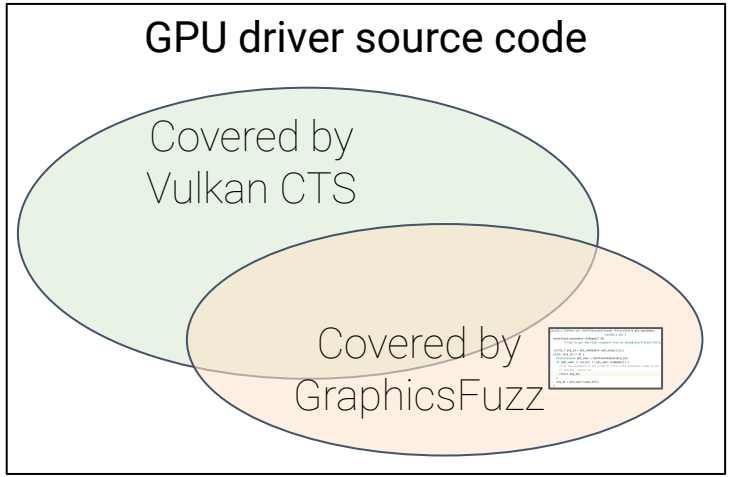
Beyond Bugs: Differential Code Coverage



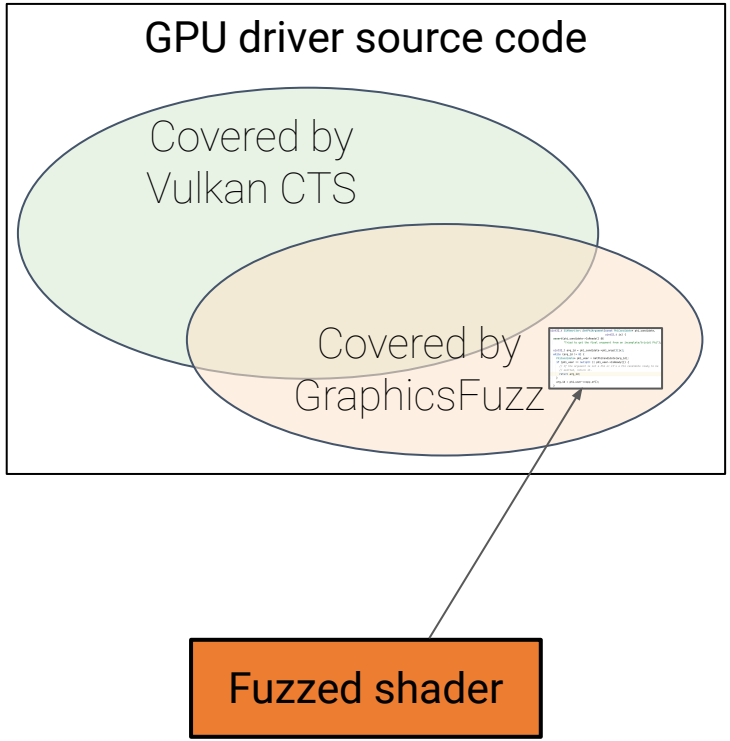
Beyond Bugs: Differential Code Coverage



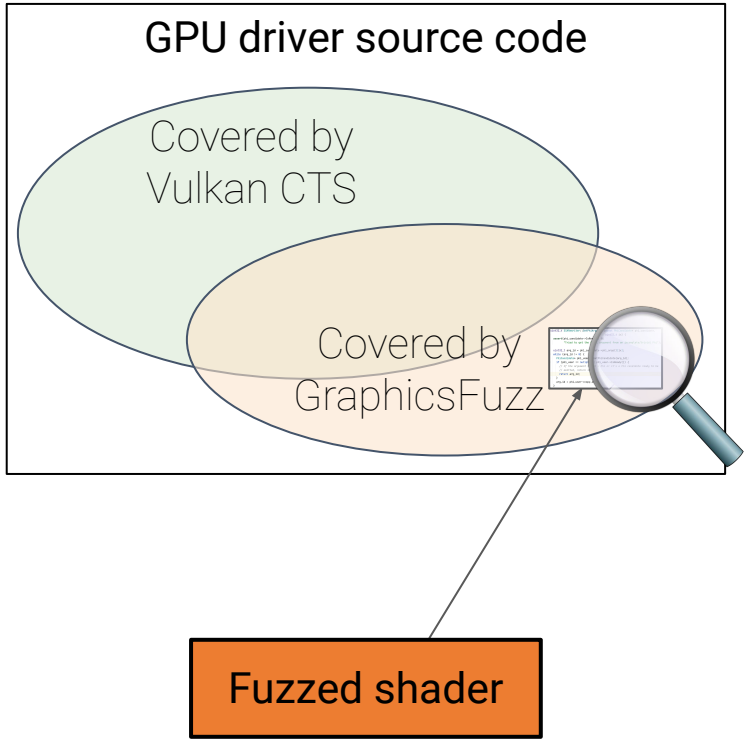
Using Reduction to Synthesise High Coverage Tests



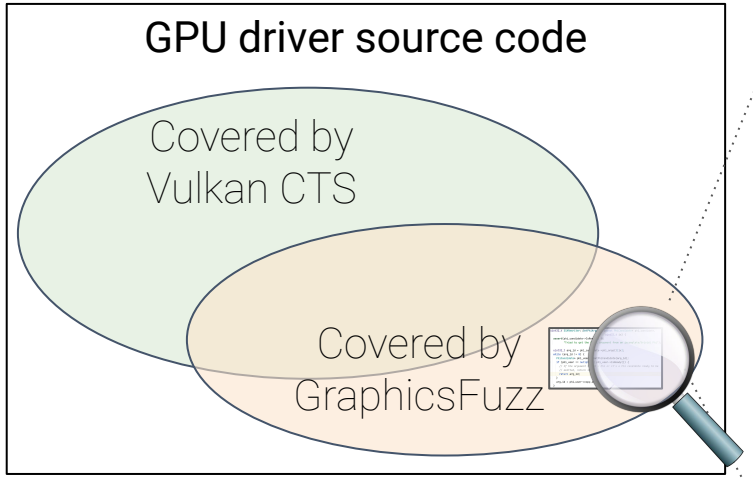
Using Reduction to Synthesise High Coverage Tests



Using Reduction to Synthesise High Coverage Tests



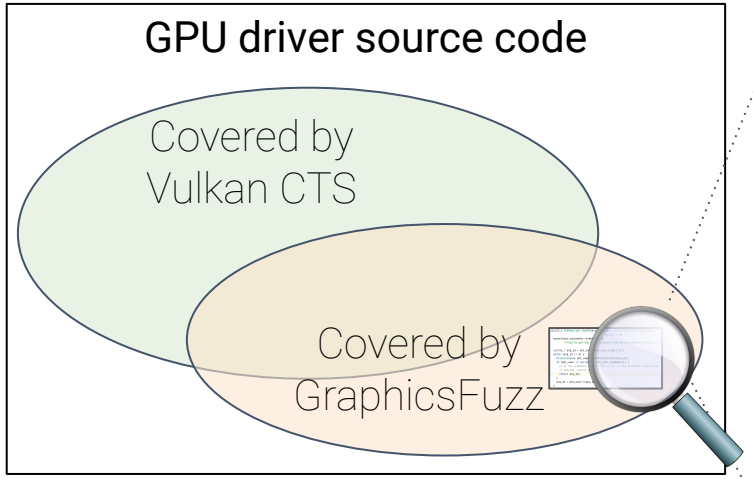
Using Reduction to Synthesise High Coverage Tests




```
uint32_t SSARewriter::GetPhiArgument(const PhiCandidate* phi_candidate,
                                     uint32_t ix) {
  assert(phi_candidate->IsReady() &&
         "Tried to get the final argument from an incomplete/trivial Phi");

  uint32_t arg_id = phi_candidate->phi_args()[ix];
  while (arg_id != 0) {
    PhiCandidate* phi_user = GetPhiCandidate(arg_id);
    if (phi_user == nullptr || phi_user->IsReady()) {
      // If the argument is not a Phi or it's a Phi candidate ready to be
      // emitted, return it.
      return arg_id;
    }
    arg_id = phi_user->copy_of();
  }
```

Fuzzed shader

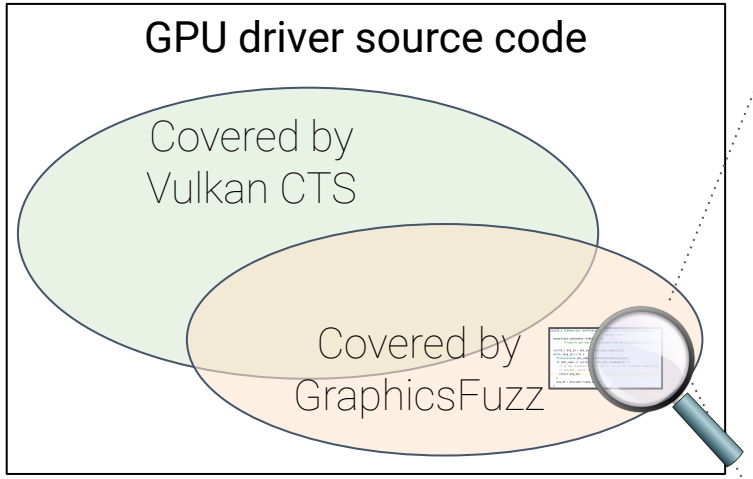
Using Reduction to Synthesise High Coverage Tests



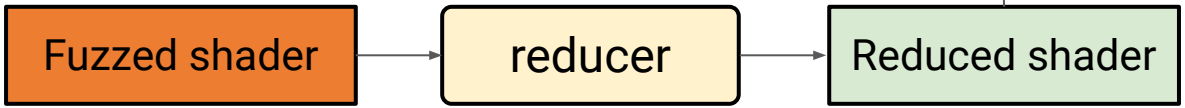

```
uint32_t SSARewriter::GetPhiArgument(const PhiCandidate* phi_candidate,
                                     uint32_t ix) {
  assert(phi_candidate->IsReady() &&
         "Tried to get the final argument from an incomplete/trivial Phi");

  uint32_t arg_id = phi_candidate->phi_args()[ix];
  while (arg_id != 0) {
    PhiCandidate* phi_user = GetPhiCandidate(arg_id);
    if (phi_user == nullptr || phi_user->IsReady()) {
      // If the argument is not a Phi or it's a Phi candidate ready to be
      // emitted, return it.
      assert (false && "GraphicsFuzz COVERAGE");
      return arg_id;
    }
    arg_id = phi_user->copy_of();
  }
```

Fuzzed shader

Using Reduction to Synthesise High Coverage Tests

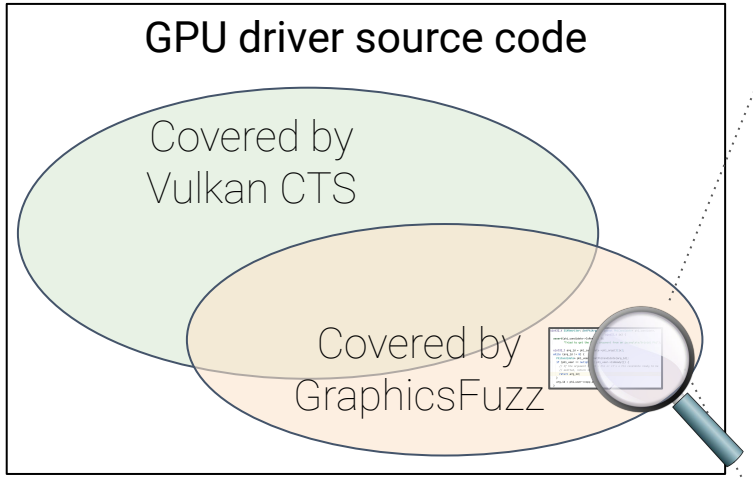



```
uint32_t SSARewriter::GetPhiArgument(const PhiCandidate* phi_candidate,
                                     uint32_t ix) {
  assert(phi_candidate->IsReady() &&
         "Tried to get the final argument from an incomplete/trivial Phi");

  uint32_t arg_id = phi_candidate->phi_args()[ix];
  while (arg_id != 0) {
    PhiCandidate* phi_user = GetPhiCandidate(arg_id);
    if (phi_user == nullptr || phi_user->IsReady()) {
      // If the argument is not a Phi or it's a Phi candidate ready to be
      // emitted, return it.
      assert (false && "GraphicsFuzz COVERAGE");
      return arg_id;
    }
    arg_id = phi_user->copy_of();
  }
```



A form of *cause reduction* (Groce et al., ICST'14)

Using Reduction to Synthesise High Coverage Tests




```
uint32_t SSARewriter::GetPhiArgument(const PhiCandidate* phi_candidate,
                                     uint32_t ix) {
  assert(phi_candidate->IsReady() &&
         "Tried to get the final argument from an incomplete/trivial Phi");

  uint32_t arg_id = phi_candidate->phi_args()[ix];
  while (arg_id != 0) {
    PhiCandidate* phi_user = GetPhiCandidate(arg_id);
    if (phi_user == nullptr || phi_user->IsReady()) {
      // If the argument is not a Phi or it's a Phi candidate ready to be
      // emitted, return it.
      return arg_id;
    }
    arg_id = phi_user->copy_of();
  }
```

Reduced shader

A small shader that obtains the extra coverage!

Manually improve oracle then submit to Vulkan CTS

Improving the Oracle

Automatically-reduced test that increases coverage

```
void main() {

  dot(vec2(1.0, 0.0), vec2(0.0))

}
```

Manually edited test with stronger oracle

```
void main() {
 if(dot(vec2(1.0, 0.0), vec2(0.0)) == 0.0) // precise check
     _GLF_color = RED;  // we expect red
 else
     _GLF_color = BLACK;
}
```

Impact of GraphicsFuzz on CTS so far

442 tests added

- 178 bug tests
- 264 coverage tests

master VK-GL-CTS / external / vulkancts / data / vulkan / amber / **graphicsfuzz** / Add

afd and **alegal-arm** Add a batch of GraphicsFuzz tests ··· 248a497 14 day

..

access-new-vector-inside-if-condition.amber	dEQP-VK.graphicsfuzz.access-new-vector-inside-if-condition
always-discarding-function.amber	dEQP-VK.graphicsfuzz.always-discarding-function
always-false-if-in-do-while.amber	Fix invalid SPIR-V in GraphicsFuzz tests
always-false-if-with-discard-return.amber	dEQP-VK.graphicsfuzz.always-false-if-with-discard-return
arr-value-set-to-arr-value-squared.amber	dEQP-VK.graphicsfuzz.arr-value-set-to-arr-value-squared
array-idx-multiplied-by-for-loop-idx.amber	Add a batch of GraphicsFuzz tests
assign-array-value-to-another-array-2.amber	Add a batch of GraphicsFuzz tests
assign-array-value-to-another-array.amber	Add a batch of GraphicsFuzz tests
barrier-in-loop-with-break.amber	dEQP-VK.graphicsfuzz.barrier-in-loop-with-break
break-in-do-while-with-nested-if.amber	dEQP-VK.graphicsfuzz.break-in-do-while-with-nested-if

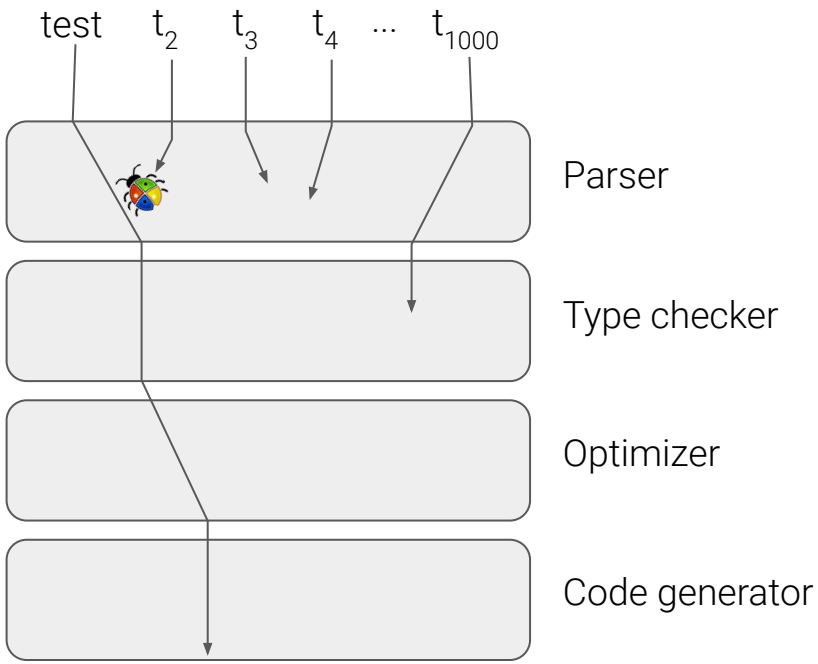
Metamorphic testing for finding vulnerabilities

- Chrome web browser: billions of users -> lots of **attackers**
- **ClusterFuzz**: continuous fuzzing of Chrome
- WebGL vulnerabilities are thus a concern
- Chrome security **do not** care about wrong images (not exploitable!)

How does metamorphic testing help?

Mutation-based fuzzing (AFL)

- Most mutated inputs: **invalid**
- Great for finding vulnerabilities in parsers
- Parsers are a first point of attack
- Not good for finding bugs deeper in system under test

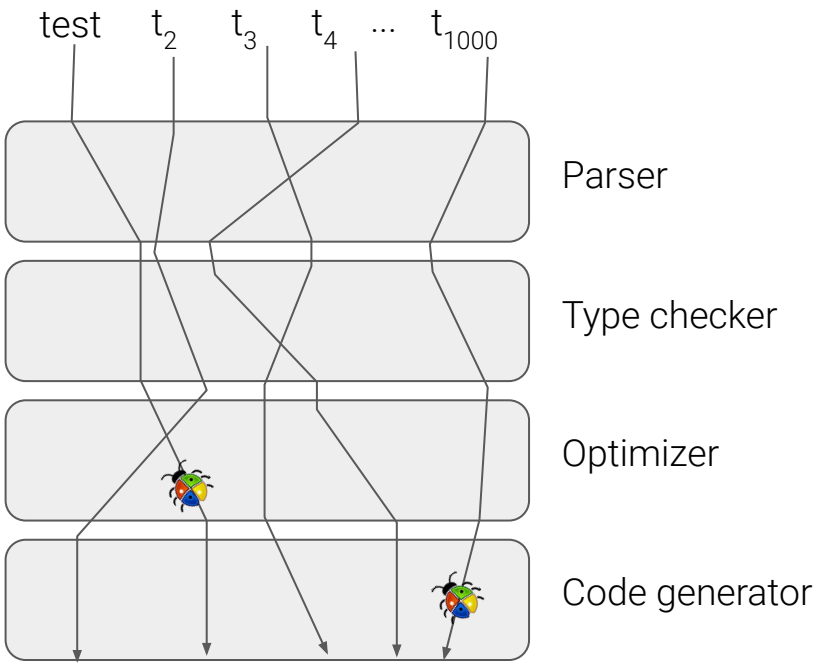


Layered SUT, e.g. a compiler

Metamorphic testing

- Original valid => variants valid
- Finds **deep** vulnerabilities
- Does not find bugs triggered by malformed inputs

Metamorphic testing conveniently produces well-formed inputs



Layered SUT, e.g. a compiler

GraphicsFuzz + ClusterFuzz finds WebGL vulnerabilities

Issue 912508: Heap-buffer-overflow in sh::SetUnionArrayFromMatrix

Code

Reported by ClusterFuzz on Thu, Dec 6, 2018, 6:06 AM EST

Detailed report: https://clusterfuzz.com/testcase?key=5177583668559872

Fuzzer: metzman_graphicsfuzz_crash_fuzzer
Job Type: linux_asan_chrome_mp
Platform Id: linux

Crash Type: Heap-buffer-overflow WRITE 4
Crash Address: 0x6250005fc100
Crash State:
  sh::SetUnionArrayFromMatrix
  sh::TIntermConstantUnion::FoldAggregateBuiltIn
  sh::TIntermAggregate::fold

Sanitizer: address (ASAN)

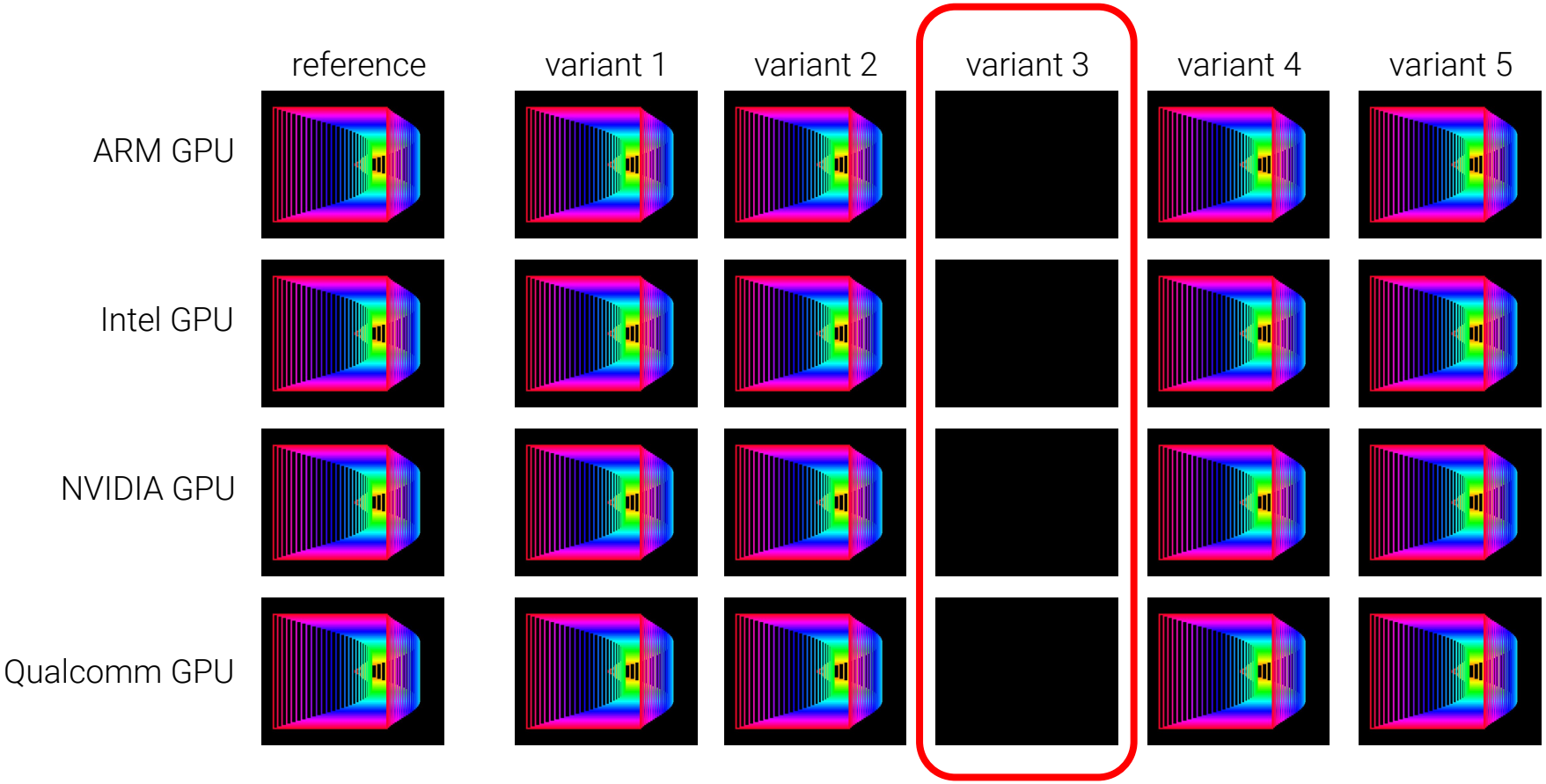
Recommended Security Severity: High

The metamorphic approach complements mutation-based fuzzing

Metamorphic + coverage-guided would be nice to try!

GraphicsFuzz tests itself!

Bug in GraphicsFuzz



Summary and Ongoing Work

- GraphicsFuzz finds bugs in shader compilers
- Cross-compilation allows us to target Vulkan
- On finding bugs we contribute **conformance tests**
- Differential coverage + test case reduction allows us to find and fill conformance test suite **coverage gaps**
- Metamorphic testing complements mutation-based fuzzing for finding **vulnerabilities**

Current work: direct fuzzing techniques for Vulkan shader compilers

Summary and Ongoing Work

- GraphicsFuzz finds bugs in shader compilers
- Cross-compilation allows us to target Vulkan
- On finding bugs we contribute **conformance tests**
- Differential coverage + test case reduction allows us to find and fill conformance test suite **coverage gaps**
- Metamorphic testing complements mutation-based fuzzing for finding **vulnerabilities**

Current work: direct fuzzing for SPIR-V, and the WebGPU shading language

Thank you! Questions?