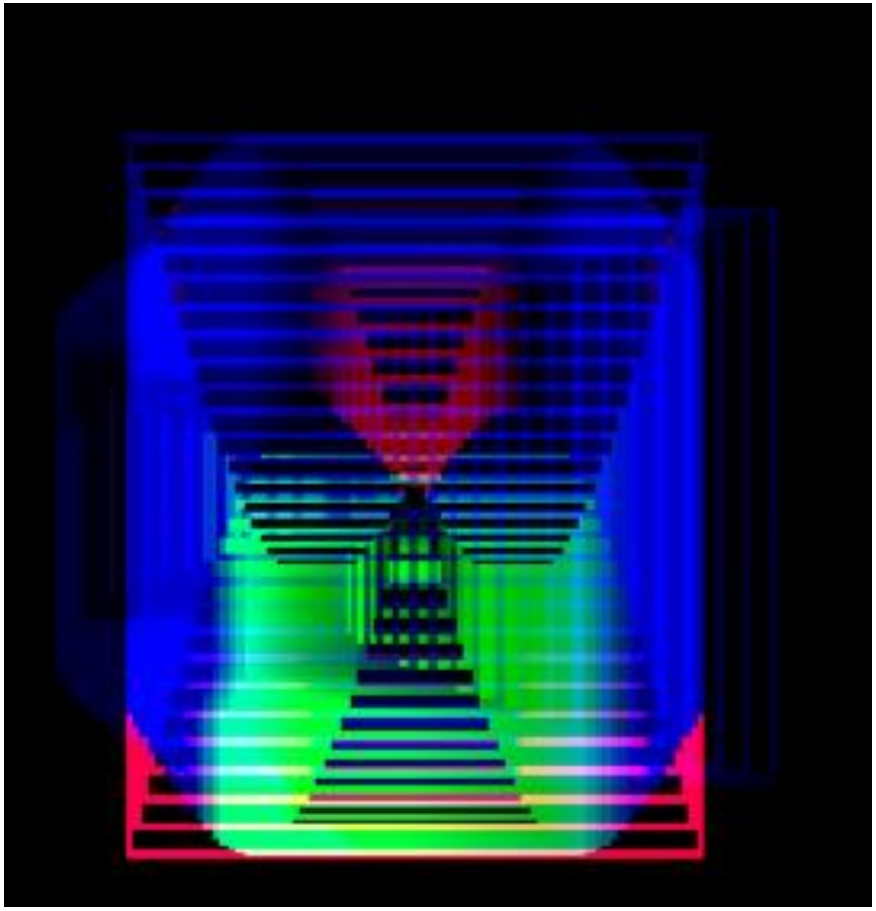
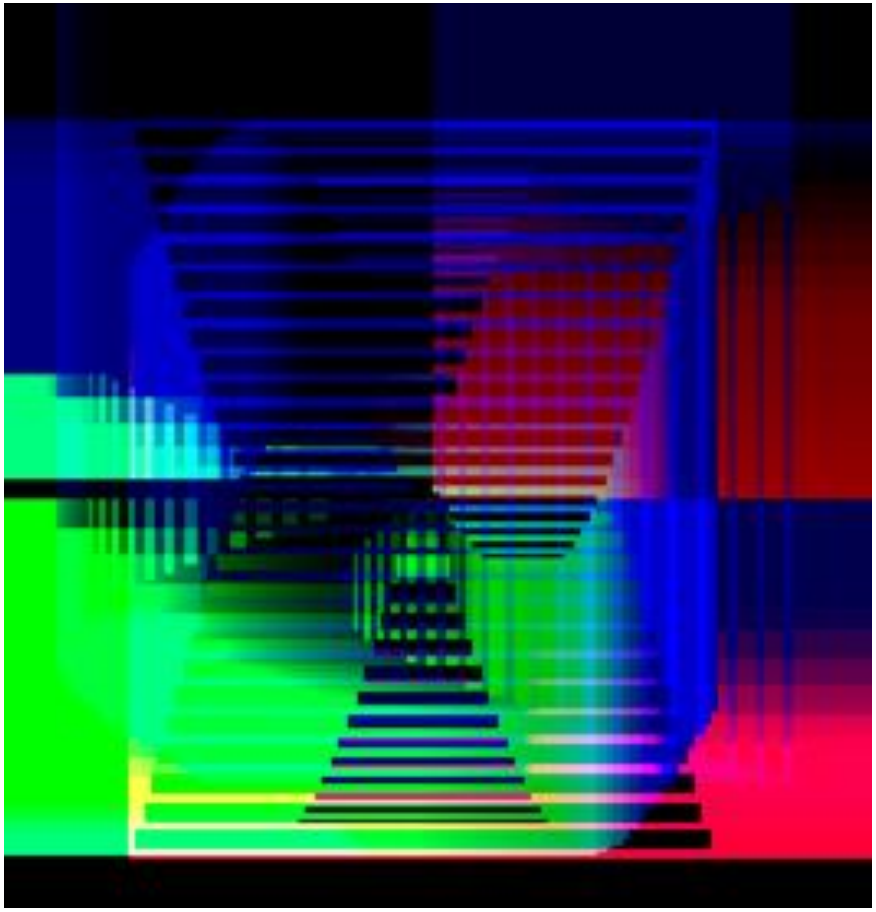
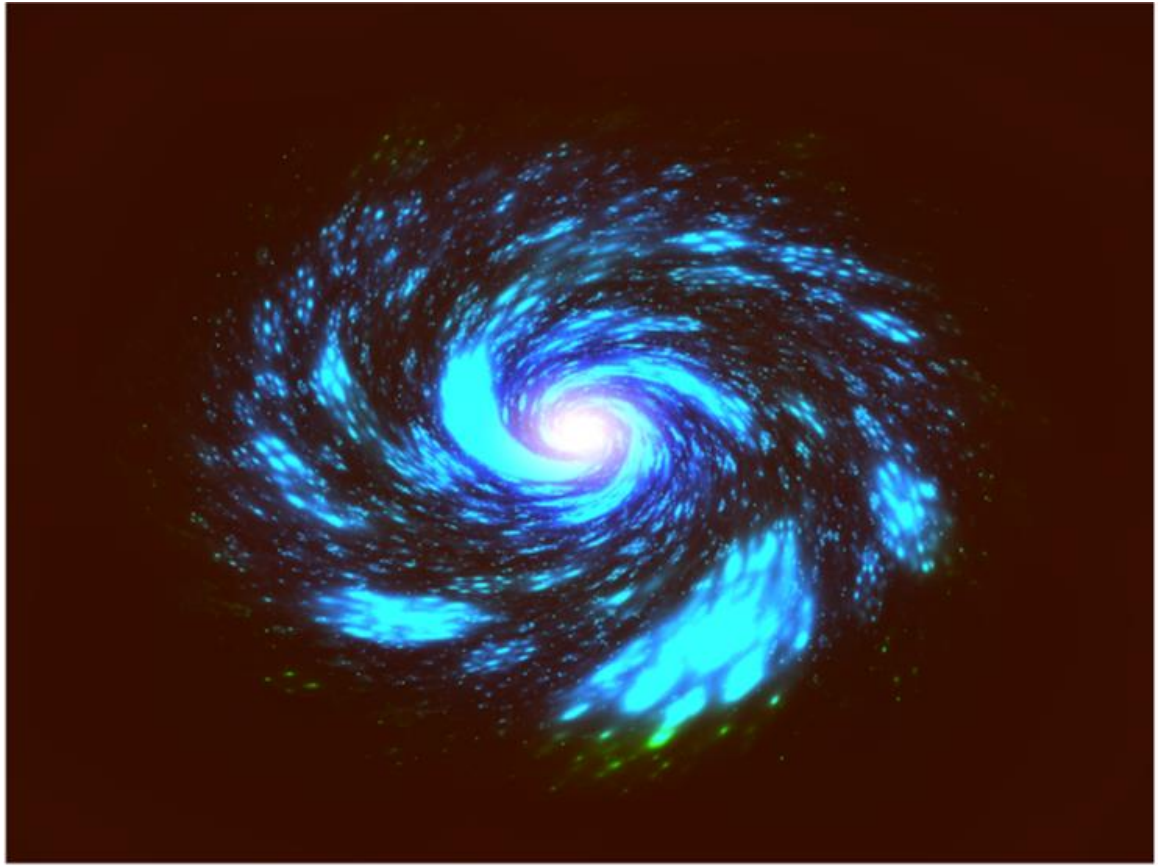
Metamorphic Testing of Android Graphics Drivers

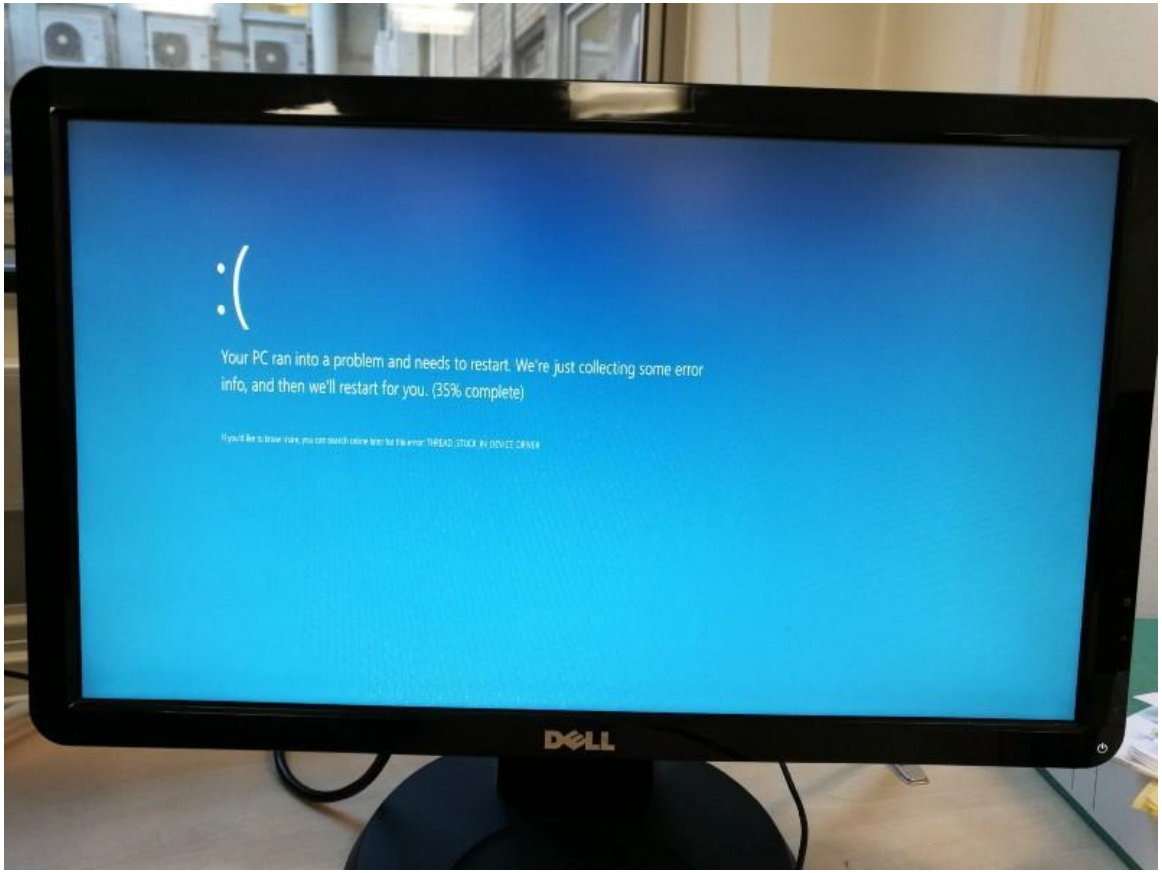
Alastair F. Donaldson
Senior Software Engineer
Android Graphics Tools Team
Google UK

First paper on the approach: Alastair F. Donaldson and Andrei Lascu: *"Metamorphic testing for (graphics) compilers"*, **MET@ICSE** 2016.








:(
Your PC ran into a problem and needs to restart. We're just collecting some error info, and then we'll restart for you. (35% complete)
DELL

GraphicsFuzz

- GraphicsFuzz finds bugs in graphics drivers using **metamorphic testing**
- In particular, it finds bugs in **shader compilers**

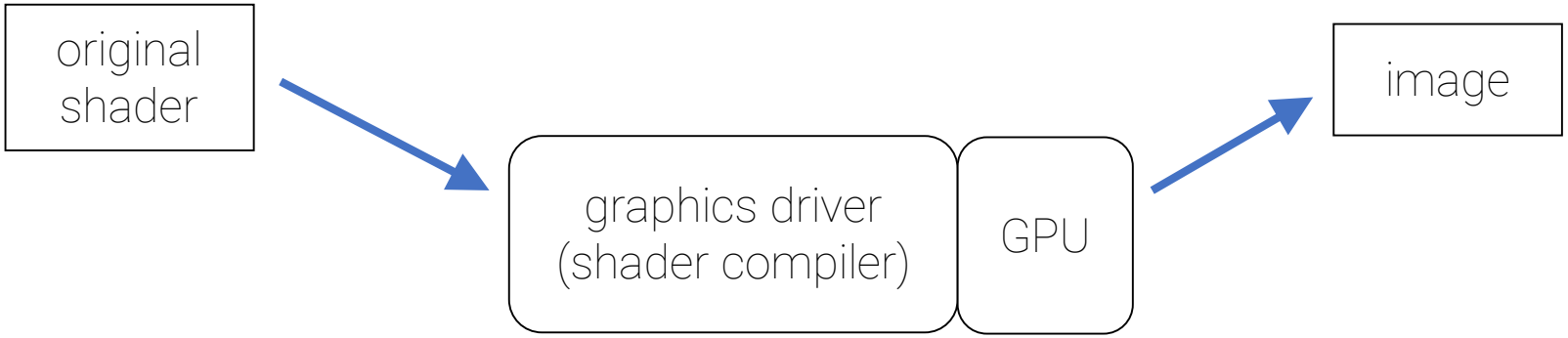
What's a shader?

- **OpenGL ES** and **Vulkan** are graphics programming models
- They orchestrate execution of operations and programs across GPU hardware
- A **shader** is a program that runs across GPU hardware
- A shader is like a kernel in CUDA / OpenGL

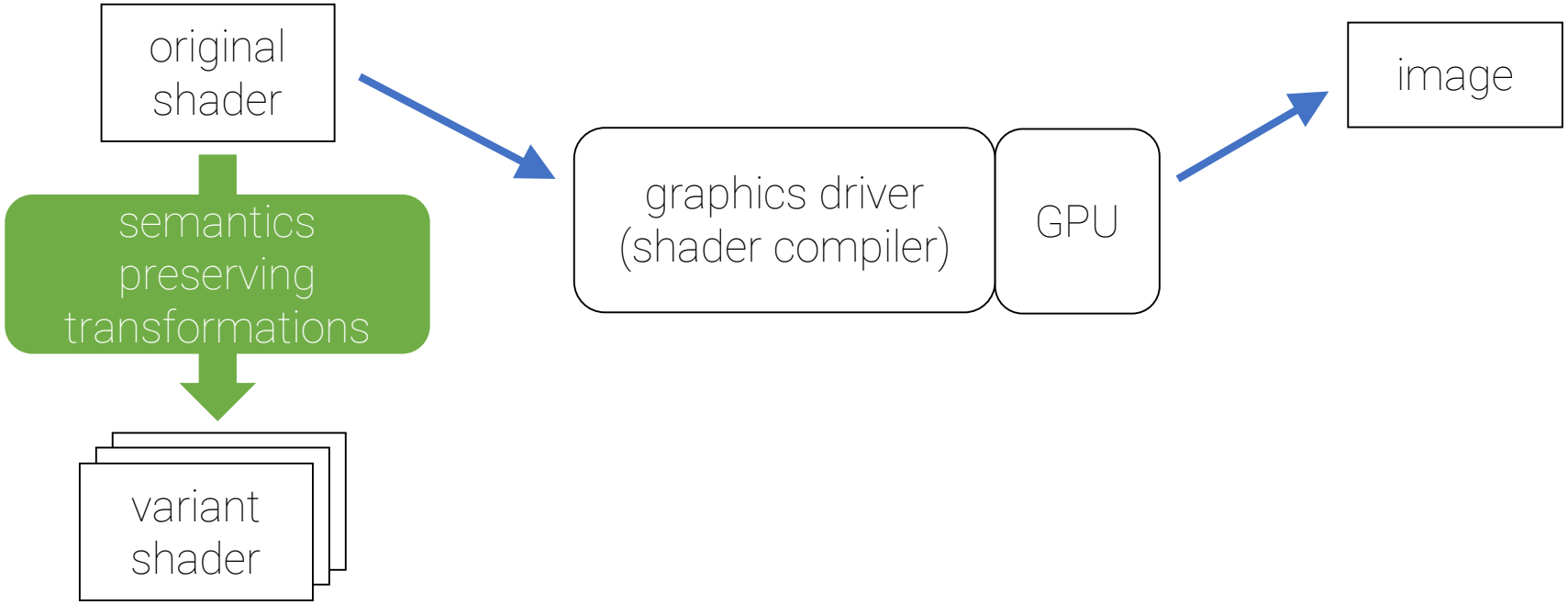
GLSL: the OpenGL shading language

SPIR-V: the Vulkan shading language

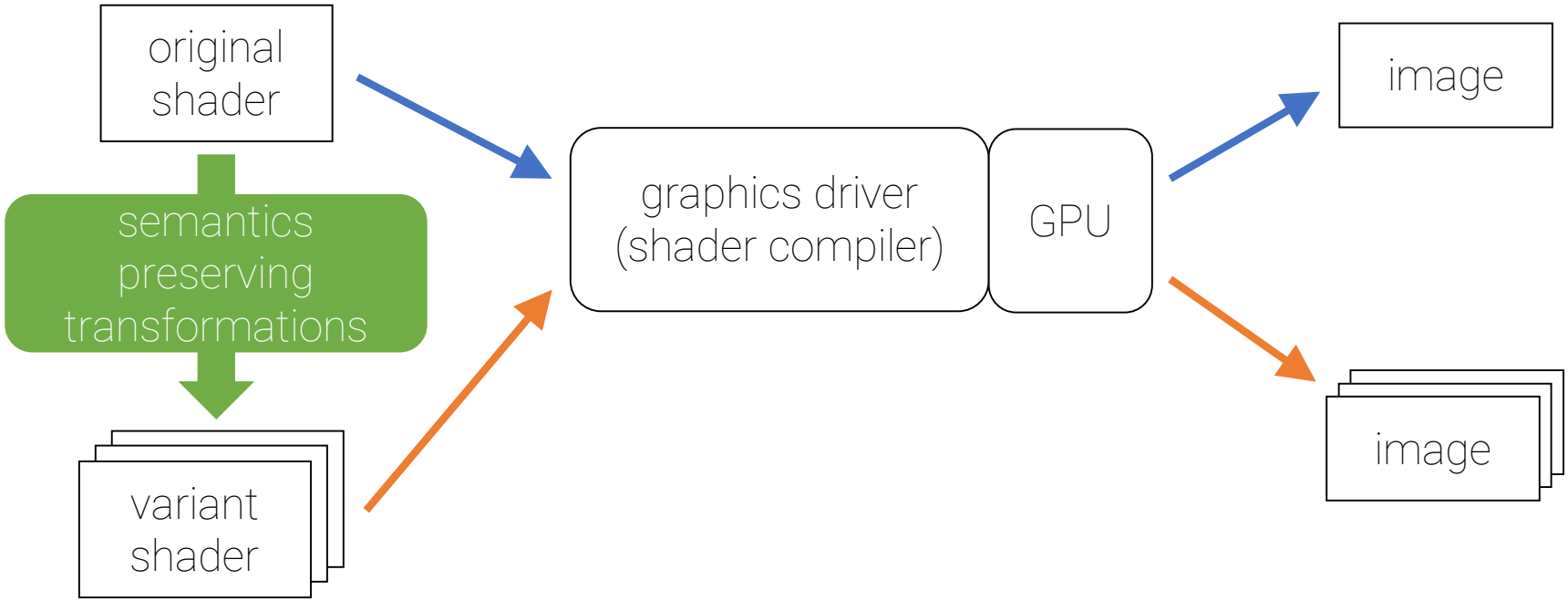
The GraphicsFuzz testing approach



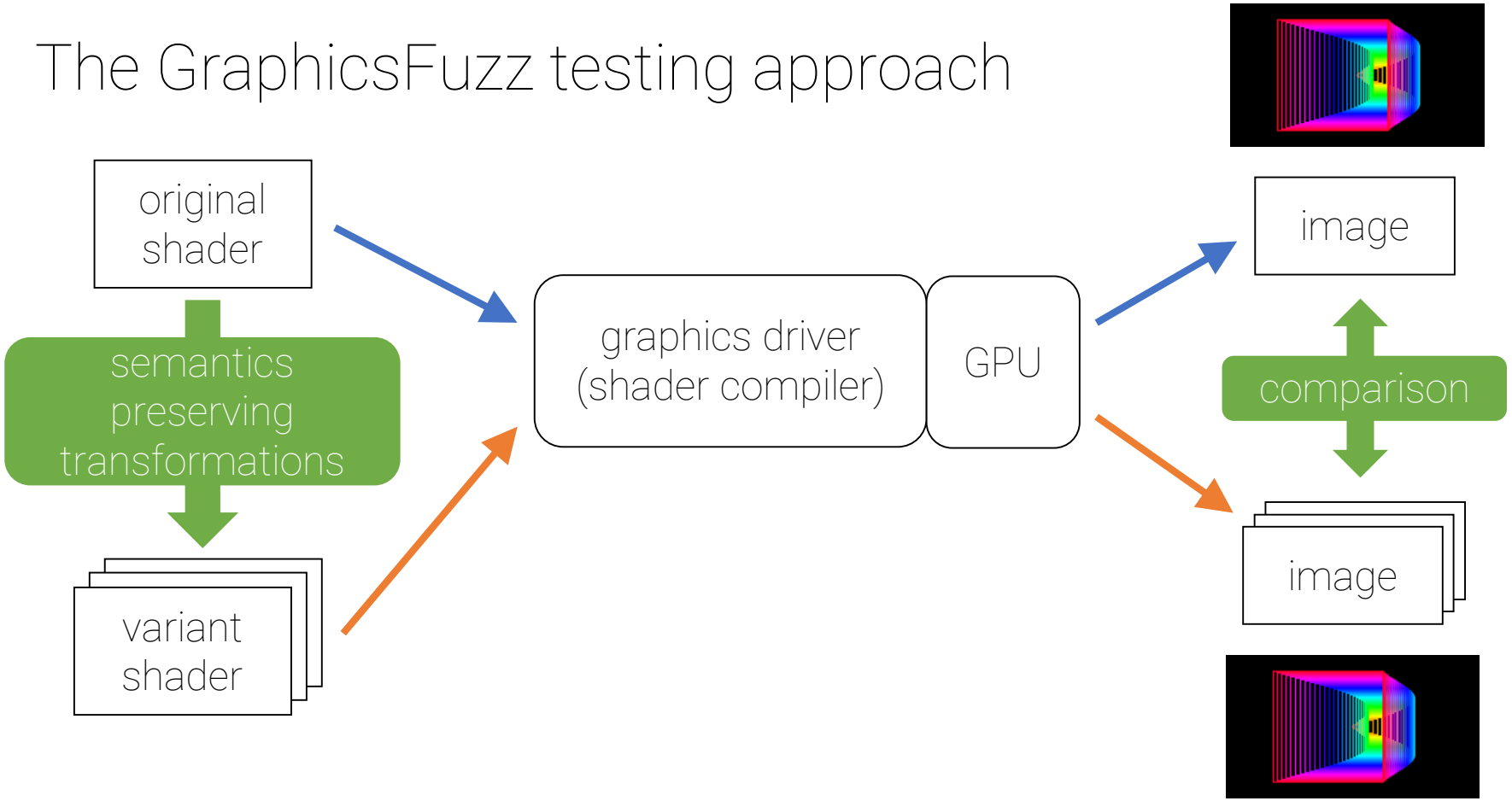
The GraphicsFuzz testing approach



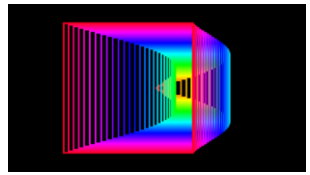
The GraphicsFuzz testing approach



The GraphicsFuzz testing approach



The GraphicsFuzz testing approach



original shader

semantics preserving transformations

variant shader

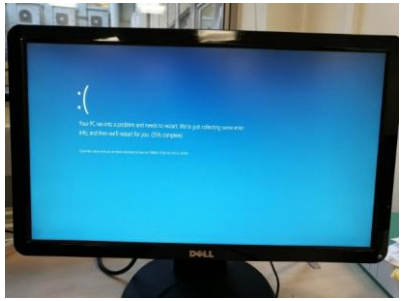
graphics driver (shader compiler)

GPU

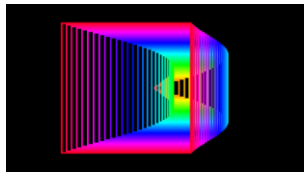
image

diff/crash

image



The GraphicsFuzz testing approach



original shader

semantics preserving transformations

variant shader

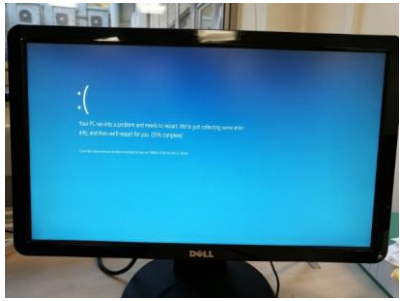
graphics driver (shader compiler)

GPU

image

diff/crash

image



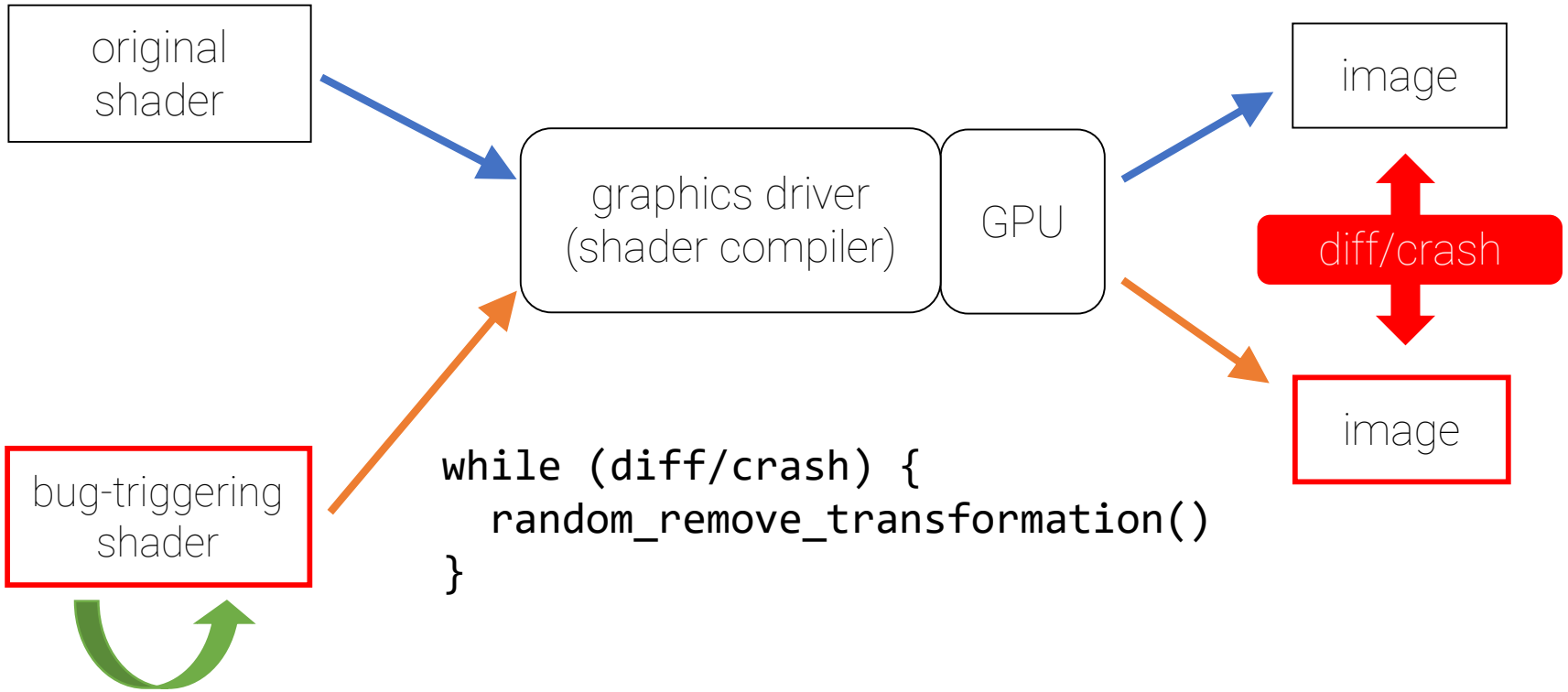
Metamorphic testing

Useful?

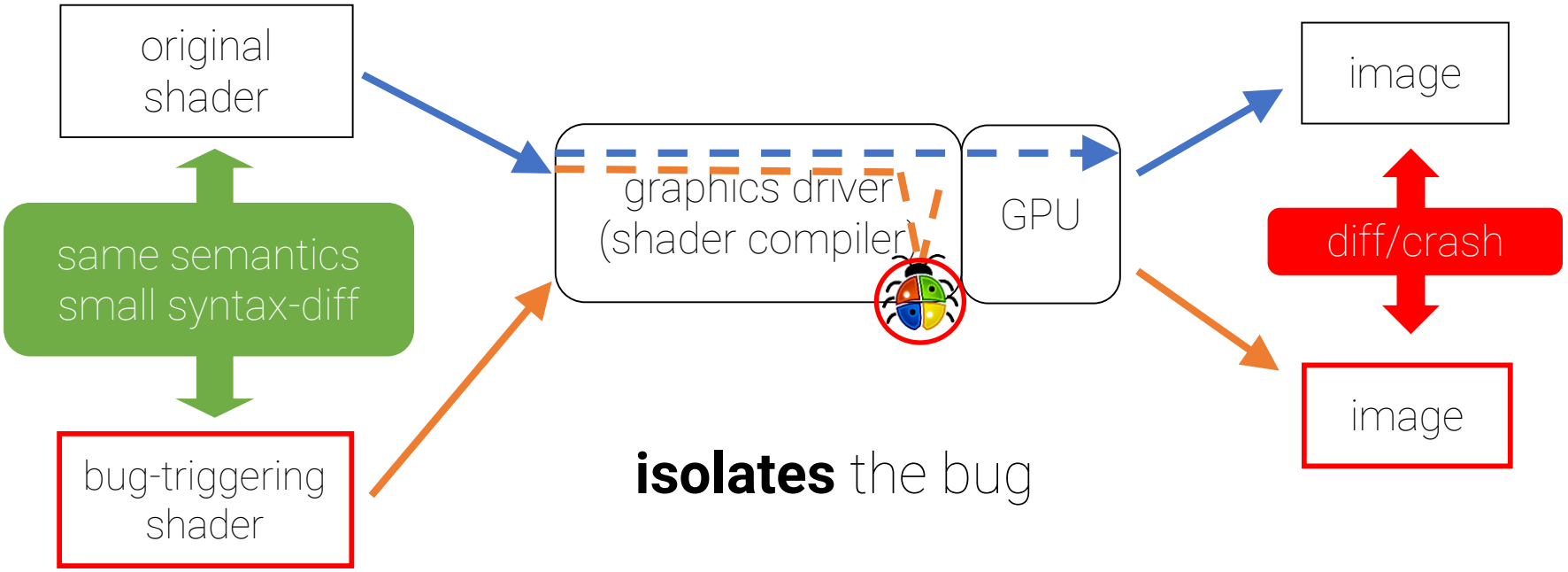
"Trust me: there's a bug in your GPU compiler. Good luck!"

```
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

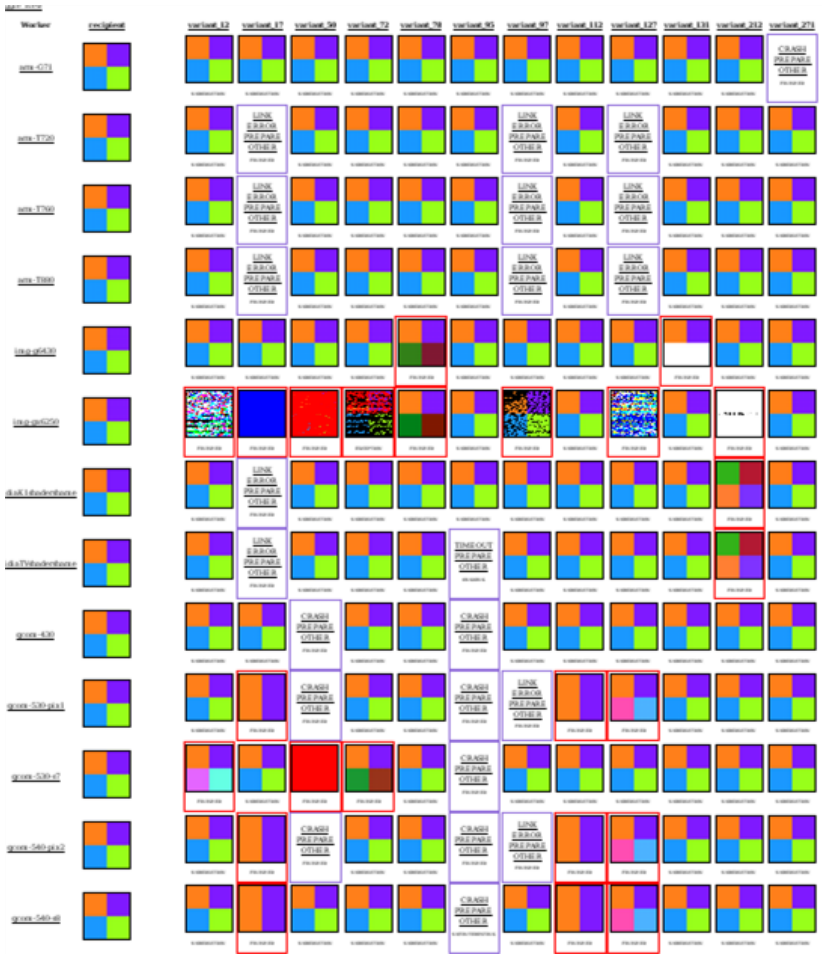


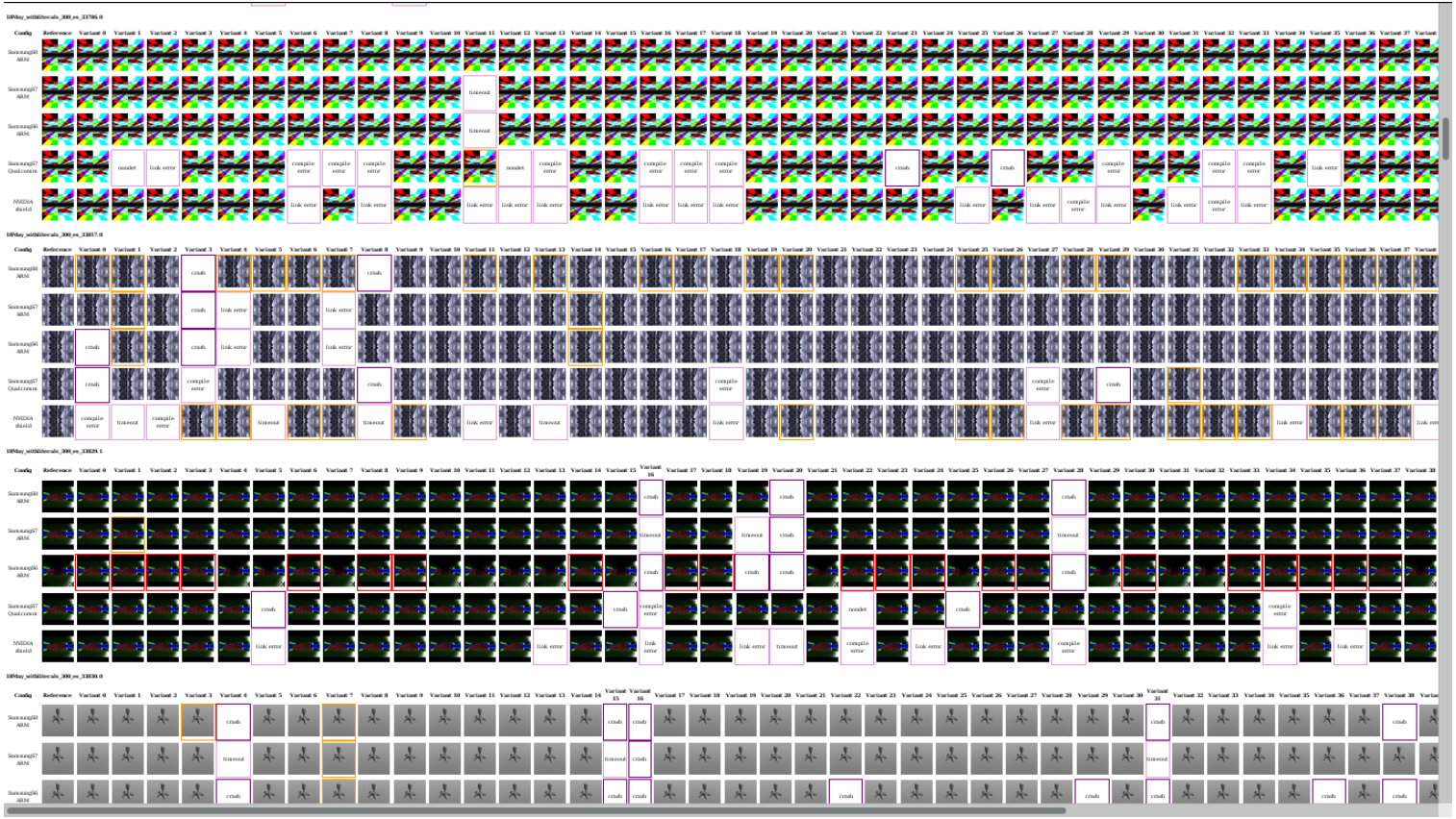
Test-case reduction



Reducer: small difference

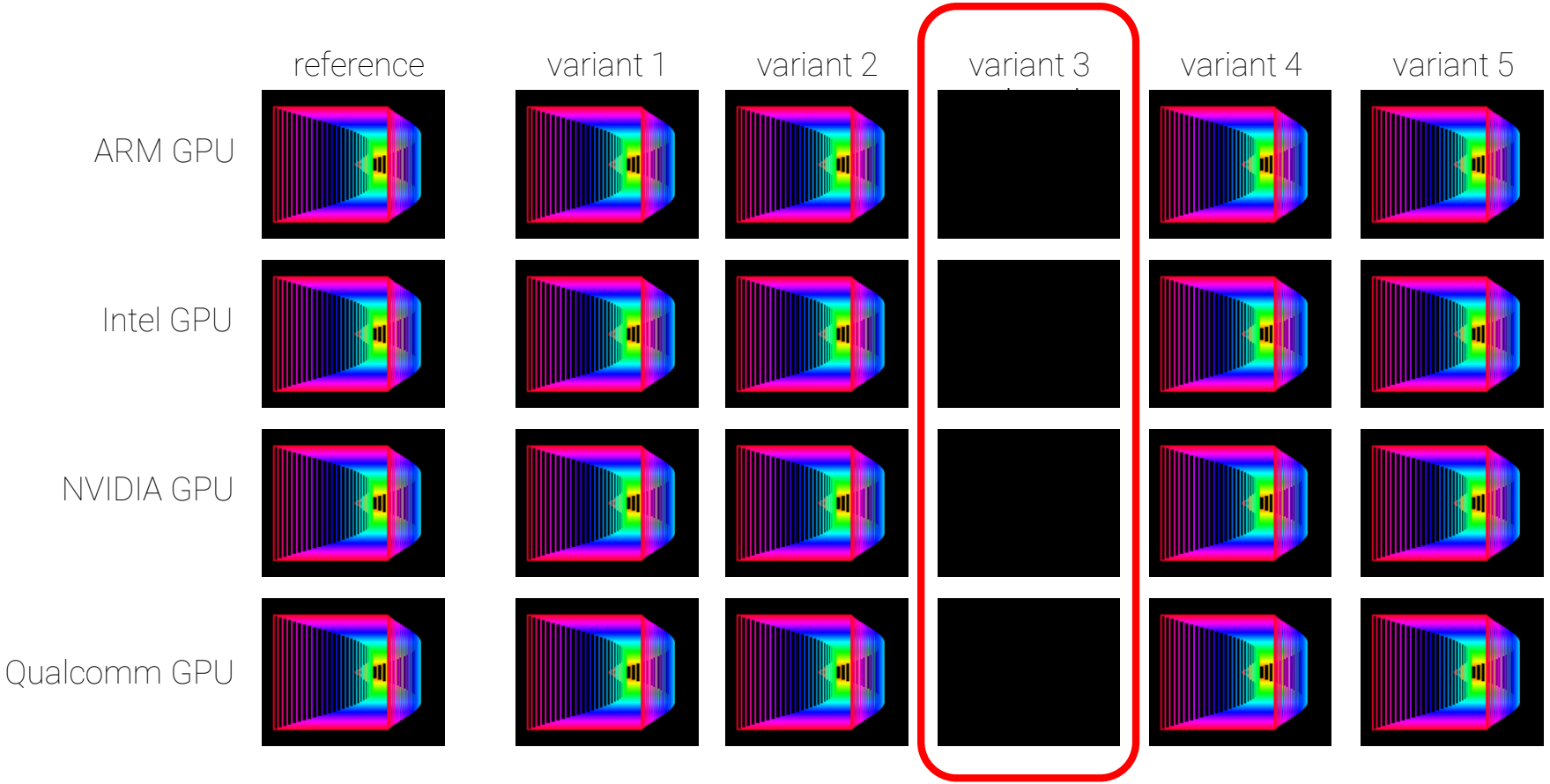
```
$ diff  reference.frag  variant_reduced.frag
194a201,204
>     if (gl_FragCoord.x < -100.0) //always false
>         {
>             return;
>         }
```


Worker
recipient
variant_12
variant_17
variant_30
variant_72
variant_78
variant_95
variant_97
variant_112
variant_127
variant_131
variant_212
variant_271
arm-v7l
arm-T720
arm-T760
arm-T880
img-gl6430
img-gt6250
disK1thudenthave
disTWthudenthave
gcore-430
gcore-530-gtx1
gcore-530-v7
gcore-540-gtx2
gcore-540-t8
LINK ERROR FILE PARE OTHER
CRASH FILE PARE OTHER
TIMEOUT FILE PARE OTHER


HMay_withliterals_300_ys_33786.0
HMay_withliterals_300_ys_33817.0
HMay_withliterals_300_ys_33829.1
HMay_withliterals_300_ys_33830.0

GraphicsFuzz tests itself!

Bug in GraphicsFuzz



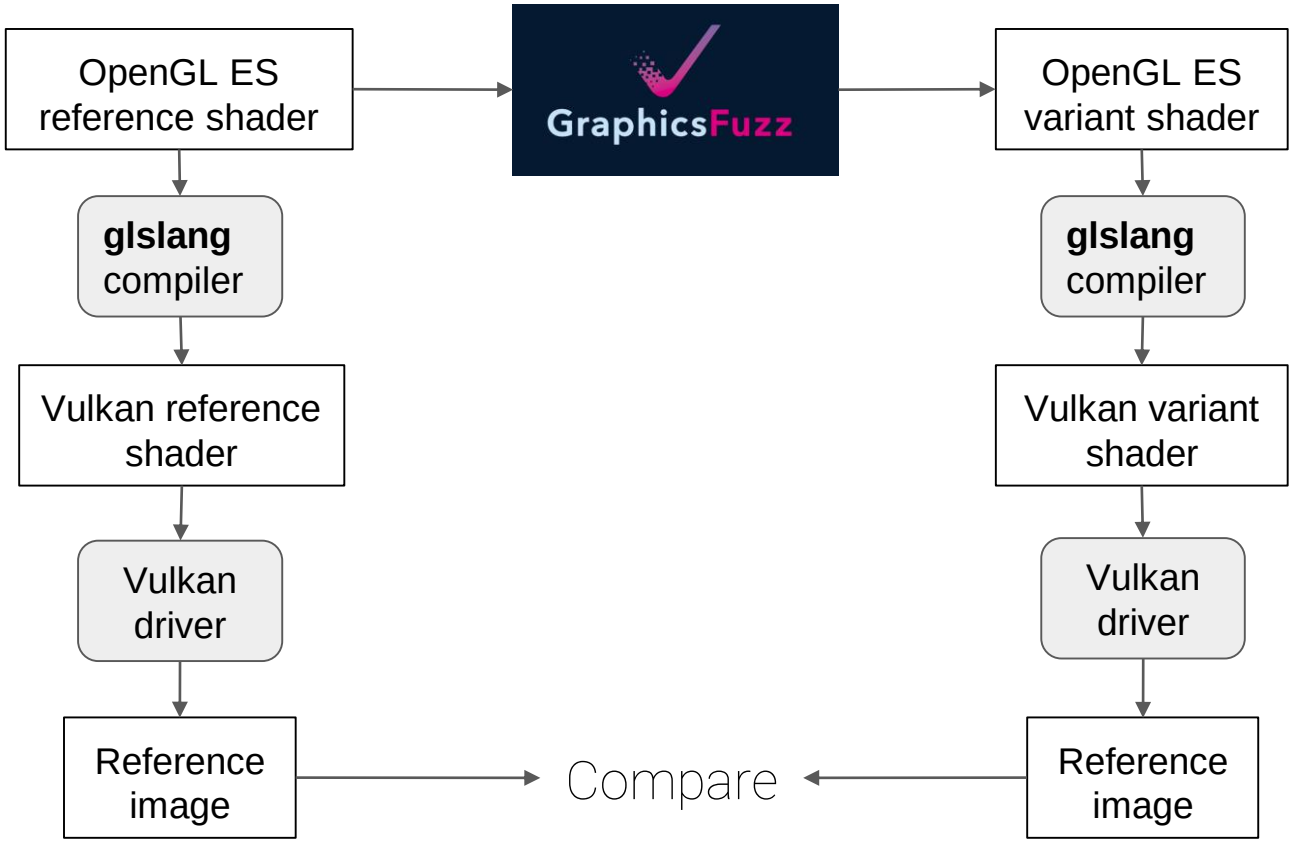
Testing Android graphics drivers

- Gaming on mobile becoming increasingly in demand
- Mobile graphics drivers need to be reliable
- OpenGL ES: legacy API for graphics
- Vulkan: modern graphics

GraphicsFuzz at Google: make Vulkan as reliable as possible

But GraphicsFuzz works on the OpenGL shading language...

Testing Vulkan via translation



Android Compatibility Test Suite (CTS)

- Each Android dessert release (.... Marshmallow, Nougat, Oreo, Pie ...) has associated ***compatibility test suite*** (CTS)
- Original equipment manufacturers (OEMs) must pass CTS for their devices to be official Google Android
- New, more rigorous tests can be added to CTS for future Android versions
- Tests for bugs found by GraphicsFuzz are being added to CTS for Android Q and R
- Ensures that associated bugs will be fixed and will not return

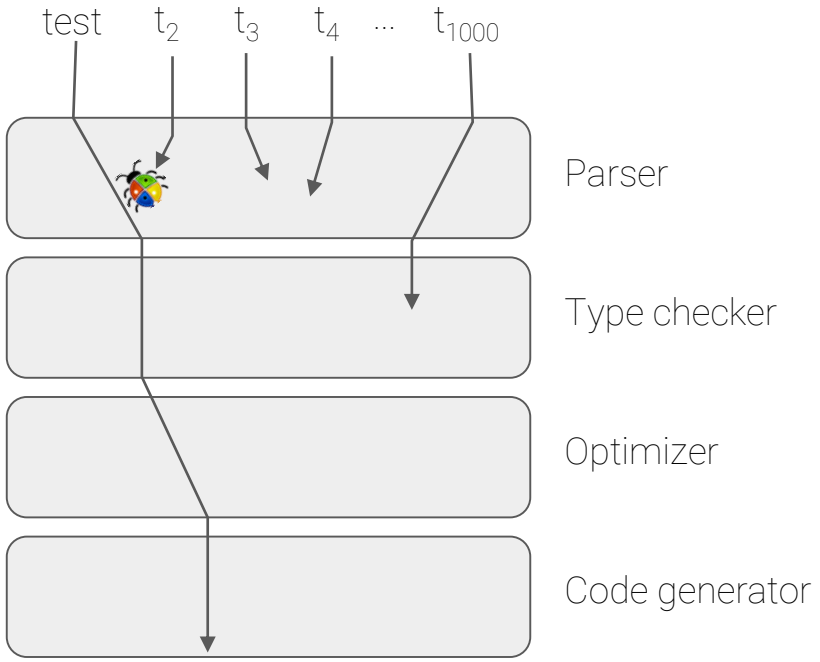
Metamorphic testing for finding vulnerabilities

- Chrome web browser: billions of users
- Lots of people trying to attack Chrome
- **ClusterFuzz**: continuous fuzzing of Chromium via:
  - Santiziers (address sanitizer, memory sanitizer, thread sanitizer …)
  - Coverage-guided mutation-based fuzzing (libFuzzer, similar to AFL)
- WebGL - OpenGL for the web - is implemented in Chrome
- WebGL vulnerabilities are thus a concern
- Chrome security **do not** care about wrong images (not exploitable!)

How does metamorphic testing fit in?

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



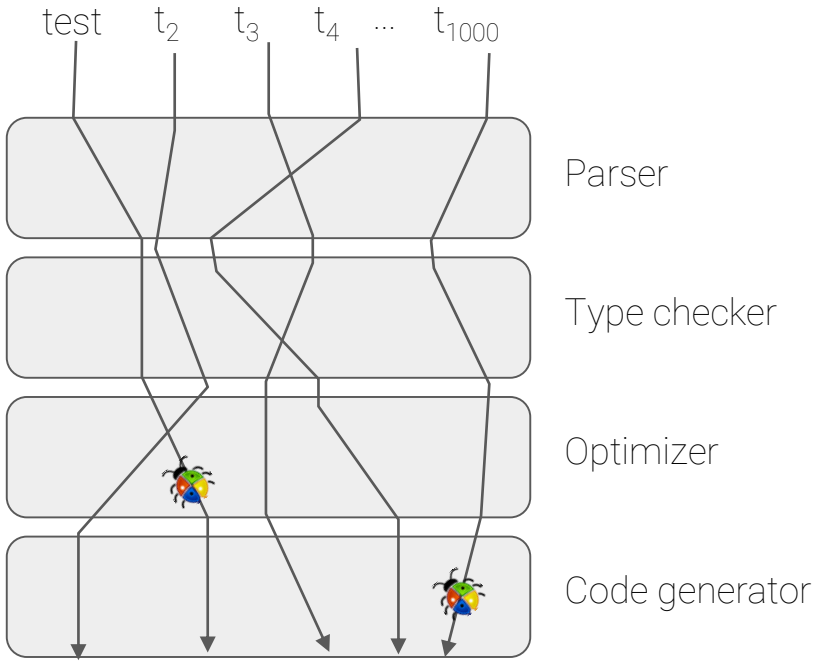
Layered SUT, e.g. a compiler

Metamorphic testing

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

Metamorphic testing conveniently produces well-formed inputs

(Other well-formed input generation methods have same benefits.)



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

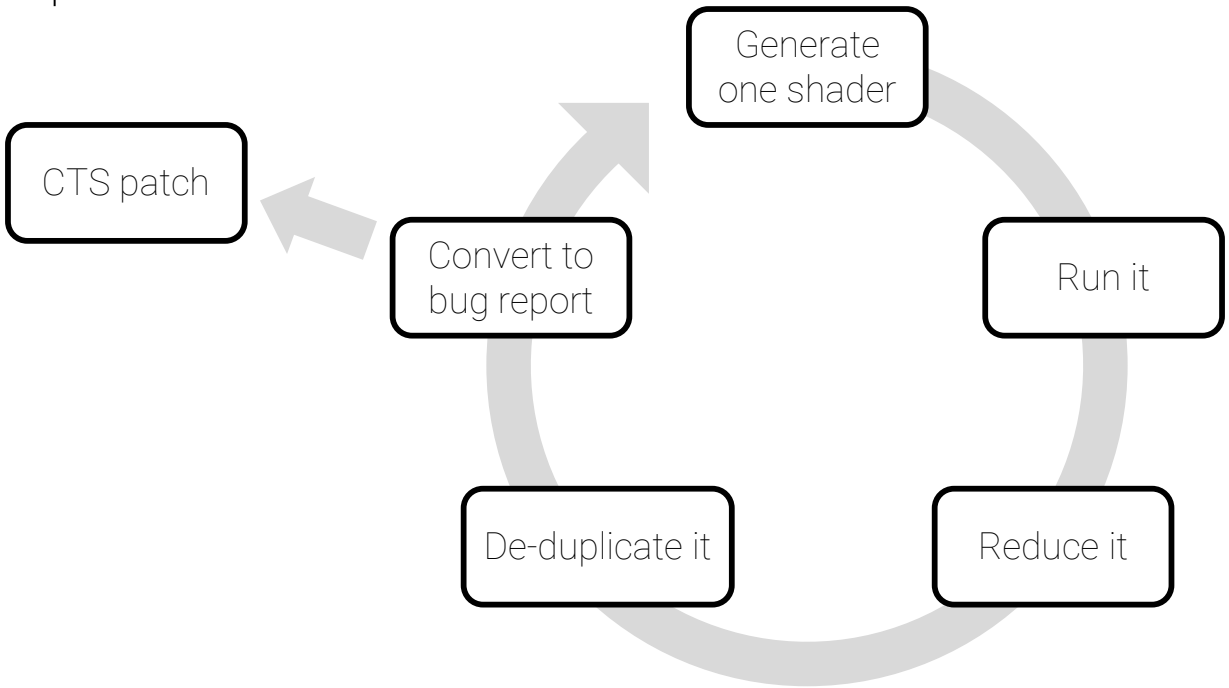
Sanitizer: address (ASAN)

Recommended Security Severity: High

The metamorphic approach complements mutation-based fuzzing

We are trying out the combination of metamorphic + coverage-guided

GraphicsFuzz auto



Fuzzing shaders from games

Summary

- GraphicsFuzz: metamorphic testing for **shader compilers**
- Families of equivalent shaders via **semantics-preserving transformations**
- Finds bugs in **Android Vulkan drivers**
- Tests exposing bugs are added to the **Compatibility Test Suite**
- Surprisingly, GraphicsFuzz finds WebGL vulnerabilities
  - Metamorphic approach keeps shaders valid
  - Valid shaders can go **deep**
- The future:
  - Direct fuzzing for Vulkan
  - Better automation
  - Fuzzing at the level of graphics APIs

Thank you!