TECHNISCHE UNIVERSITÄT DRESDEN

DRESDEN concept

cfaed CENTER FOR ADVANCING ELECTRONICS DRESDEN

Code Specialization

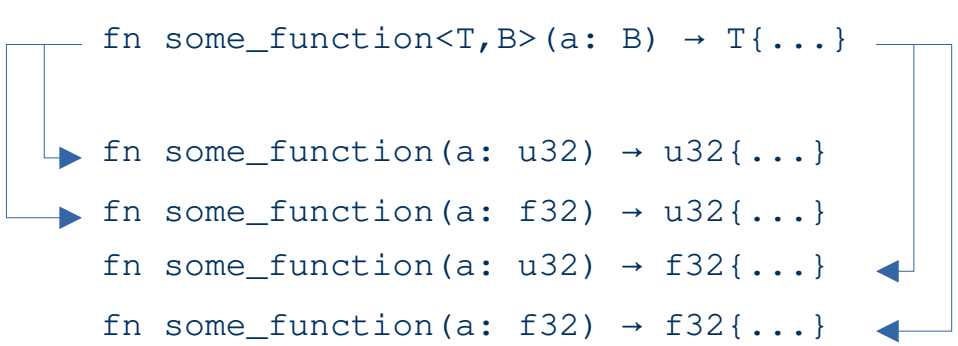
SPIR-V patching for runtime specialized code targeting GPUs

Agenda

- Motivation
- Application
- Implementation
- Testing & Benchmarking
- Outlook

Motivation
Improving shader specialization

(JIT) - Shader permutation[1]



Constant branch

```
if (const I > N){...}else{}

switch (const I){...}
```

Runtime Code specialization

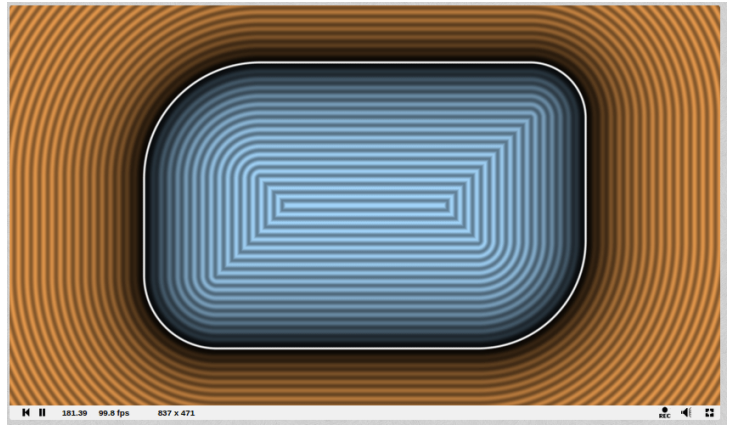
- Similar to *specialization constants*
- Whenever not all paths are known at compile-time
  - Updating calculation at runtime
  - *Context* optimized code

[1] https://www.kdab.com/shader-variants/
https://therealmjp.github.io/posts/shader-permutations-part2/

Motivation
Personal motivation

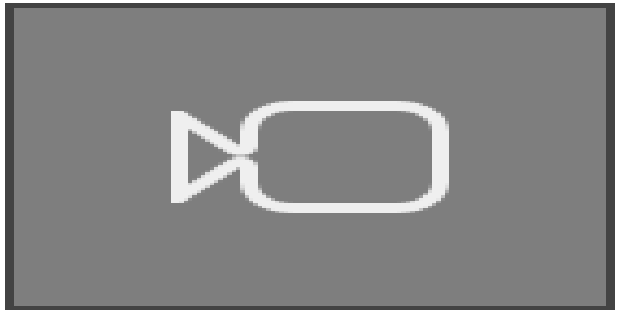
Improving Signed-Distance-Function viability for user generated content.

- GPU-Interpreter too slow (Nako[2])
- Can not pre-program *all* SDF (and select via constants)
- Rendering into explicit representation looses most interesting properties
  - Low memory usage
  - possibly *infinite* spatial size
  - Optimized evaluation of implicit form



```
float sdRoundBox( in vec2 p, in vec2 b, in vec4 r )
{
    r.xy = (p.x>0.0)?r.xy : r.zw;
    r.x  = (p.y>0.0)?r.x  : r.y;
    vec2 q = abs(p)-b+r.x;
    return min(max(q.x,q.y),0.0) + length(max(q,0.0)) - r.x;
}
```

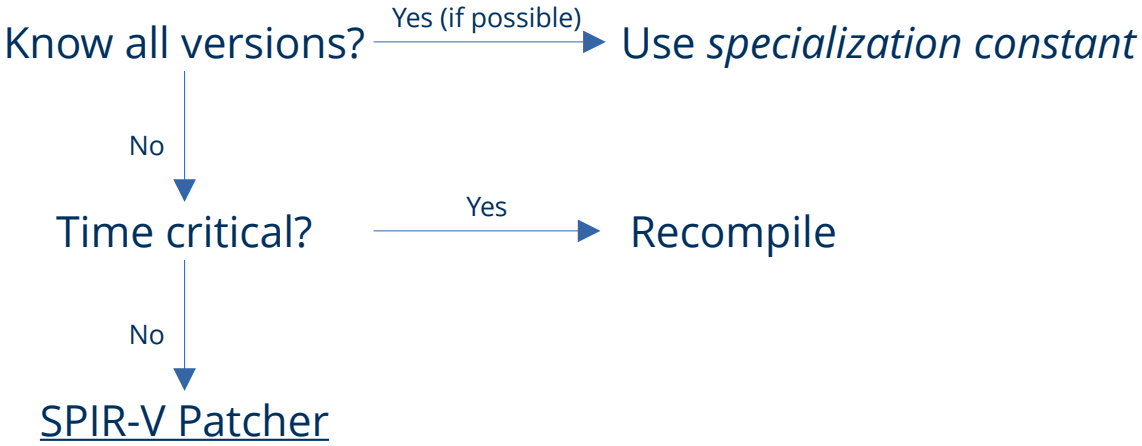
Inigo Quilez:
https://www.shadertoy.com/view/4llXD7



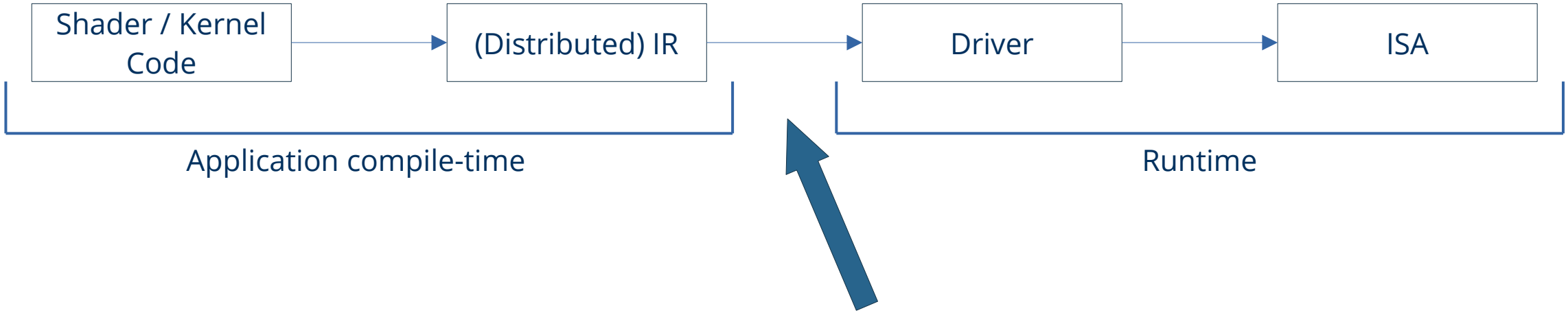
[2] https://gitlab.com/tendsinmende/nako

Application

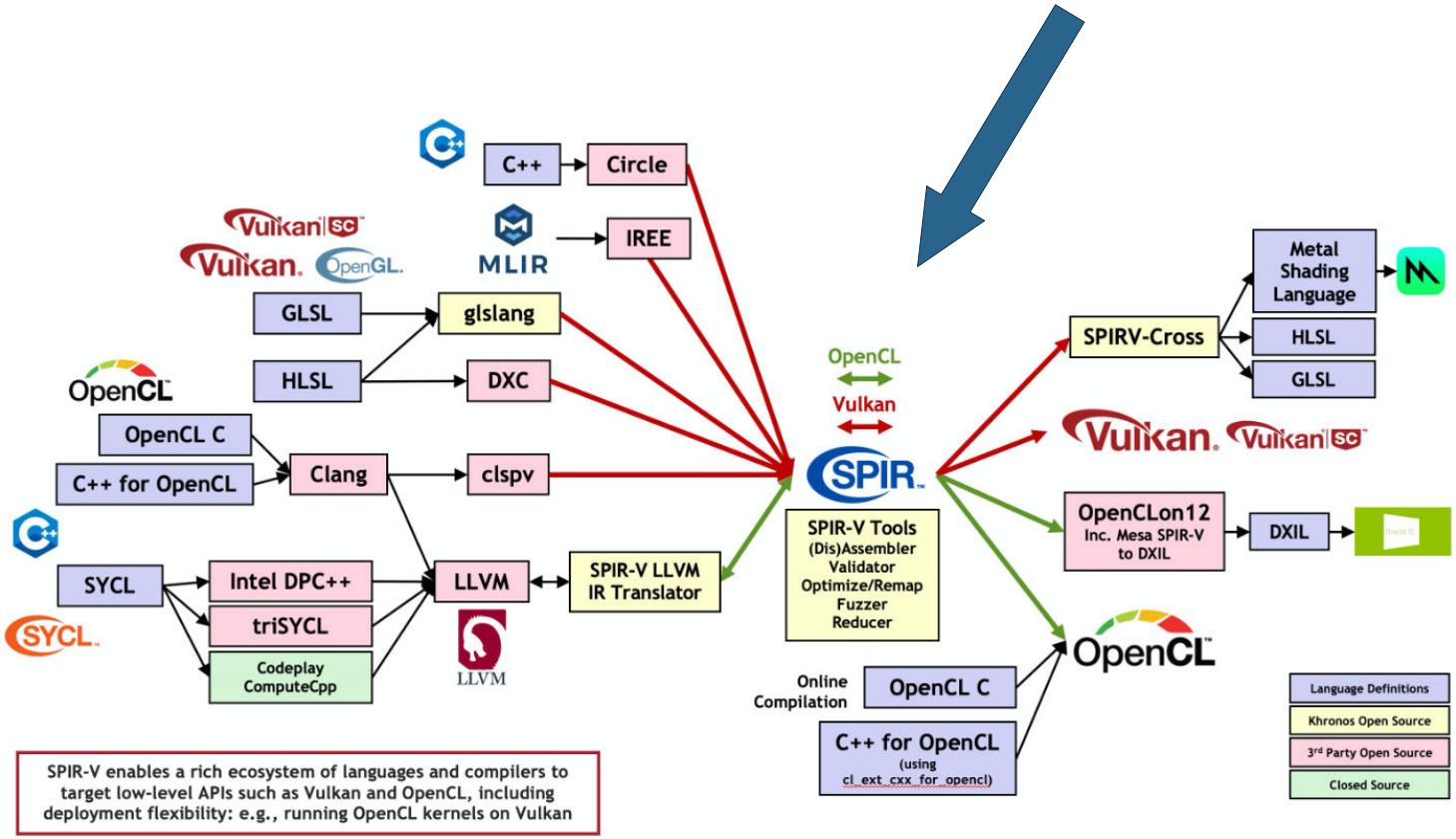
- User generated code
- Runtime modified code
- Performance evaluation



How to patch
Choosing an abstraction level



How to patch
Chosing an abstraction level



How to patch
Choosing an abstraction level

Challenges

- SPIR-V Dialects (Vulkan, OpenCL, actual implementation of cross-compiler)
- Differences across dialects (sometimes mutual exclusive)

Advantages

- Cross vendor
- Cross-compiles to many proprietary targets (DX12, Metal)
- Open source drivers down to ISA level on Linux (Mesa, ACO)
- Multiple source compiler (GLSL, HLSL, Rust, C, C++)

Patches
Finding a function

Challenges

- Per-Module type IDs
- often modules are completely inlined
- Type composition
- Type splitting based on decorations

Solution

Finding functions

- Based on signature
- Based on debug names
- Merge module’s IDs before matching

Patches
Constant Patching

- Knows template and patch in advance
- Knows destination function signature or name
- Matching function signature

1. find function in template
2. add linking capability
3. link-import (name)
4. link-export (name) in patch
5. call `spirv-link`
6. Check for valid linked module

Patches
Dynamic Patching

- Knows template
- Does **not** know patch's code
- Knows destination function signature or name

1. Copy destination function signature
2. Rewrite all call-sites to new function ID
3. Call user defined function with signature information
   (return ID / type and argument IDs / types)
4. Link return value to return ID
5. Verify correct type IDs

```
%81 = OpFunctionCall %uint %no_inline_function__calculation %80 %46
                     ...

; Function no_inline_function__calculation
%no_inline_function__calculation = OpFunction %uint
DontInline %41
        %107 = OpFunctionParameter %uint
        %108 = OpFunctionParameter %uint
        %109 = OpLabel
               OpLine %9 18 4
        %110 = OpIAdd %uint %107 %108
               OpNoLine
               OpReturnValue %110
               OpFunctionEnd
```

```
  %81 = OpFunctionCall %uint %111 %80 %46
                  ...

        ; Function 111
%111 = OpFunction %uint None %41
%112 = OpFunctionParameter %uint
%113 = OpFunctionParameter %uint
%114 = OpLabel
%115 = OpIMul %uint %112 %113
       OpReturnValue %115
       OpFunctionEnd
```

Testing
Code and runtime verification

Patch-Time verification

→ Use `spirv-val`[3] to check against SPIR-V Spec.

→ Possibly implement custom *verification-pass*

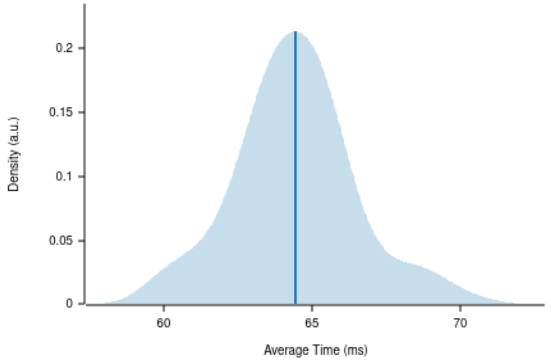
Runtime verification

- Vulkan validation layer
- Driver site error reporting
- Vulkan testbench

[3] https://github.com/KhronosGroup/SPIRV-Tools

Benchmarking
Patching

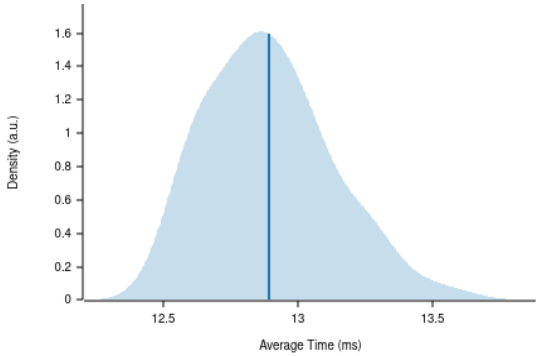
→ Time taken for a whole, usable (assembled) shader module



Additional Statistics:

	Lower bound	Estimate	Upper bound
R^2	0.0016507	0.0017127	0.0016472
Mean	64.015 ms	64.401 ms	64.798 ms
Std. Dev.	1.6653 ms	2.0013 ms	2.3032 ms
Median	63.894 ms	64.359 ms	64.905 ms
MAD	1.2733 ms	1.5827 ms	2.0045 ms

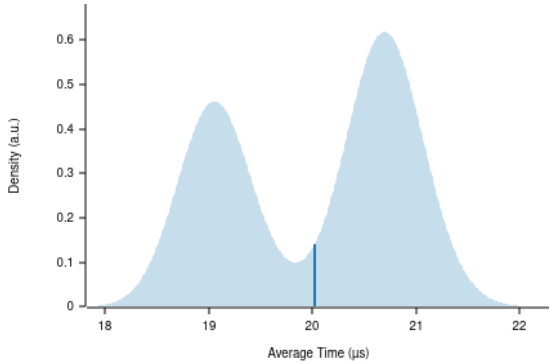
GLSLang compiletime



Additional Statistics:

	Lower bound	Estimate	Upper bound
R^2	0.0101258	0.0105023	0.0101001
Mean	12.849 ms	12.893 ms	12.939 ms
Std. Dev.	196.74 µs	229.97 µs	259.52 µs
Median	12.817 ms	12.874 ms	12.917 ms
MAD	176.06 µs	243.48 µs	289.44 µs

Constant Patching



Additional Statistics:

	Lower bound	Estimate	Upper bound
Slope	20.519 µs	20.603 µs	20.674 µs
R^2	0.9340341	0.9359712	0.9346036
Mean	19.856 µs	20.018 µs	20.178 µs
Std. Dev.	773.98 ns	820.95 ns	850.50 ns
Median	19.433 µs	20.599 µs	20.650 µs
MAD	163.67 ns	397.84 ns	1.2133 µs

Dynamic Patching

Benchmarking
Runtime

→ Runtime of the whole compute shader

→ Measured via GPU timestamps

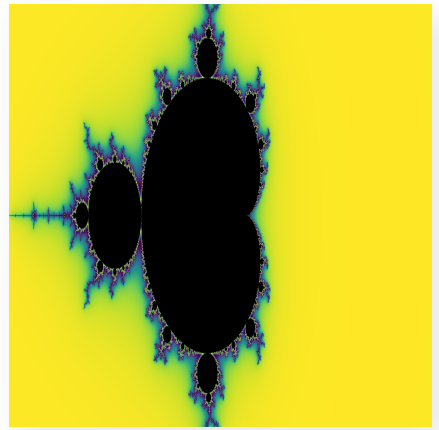
Time taken to calculate 2048²px Mandebrot set @ max 1024 iterations

Dedicated Graphics				Integrated Graphics		
Name	avg	min	max	avg	min	max
Unmodified	0.05ms	0.02ms	0.07ms	0.35ms	0.37ms	0.4ms
Rust-GPU compiled	2.31ms	2.3ms	2.35ms	11.41ms	11.41ms	11.42ms
Patched Runtime	1.55ms	1.50ms	1.58ms	13.6ms	13.65ms	13.7ms

Constant Replace runtime

Dedicated Graphics				Integrated Graphics		
Name	avg	min	max	avg	min	max
Unmodified	0.05ms	0.02ms	0.07ms	0.35ms	0.37ms	0.4ms
Rust-GPU compiled	2.0ms	2.05ms	2.2ms	11.45ms	11.45ms	11.45ms
Patched Runtime	1.55ms	1.50ms	1.58ms	13.6ms	13.65ms	13.7ms

Dynamic Replace runtime



Outlook

Higher level

→ Patch simulation code

→ SDF-DSL

→ Performance testbench for emitted SPIR-V

Lower level

→ Emit driver specific IR (NIR?) for performance gain

→ Evaluate other, similar IR level (SPIR-T, NIR, whatever ACO uses internally)

→ Somehow prevent *pipeline rebuild*

TECHNISCHE UNIVERSITÄT DRESDEN

DRESDEN concept

cfaed CENTER FOR ADVANCING ELECTRONICS DRESDEN

Thanks for attending!

And check out the code at
gitlab.com/tendsinmende/spv-patcher if you want!