



PowerVR: A Year of Progress

Simon Perretta
Frank Binns

Who we are



Simon Perretta

- Joined Imagination as a grad in 2018 as part of the OpenCL driver team
 - Moved to maintaining shader debugger and proprietary compiler team not long after
- Moved to open-source team as sole compiler developer in early 2021
 - OSS compiler team has since tripled in size 🎉

Frank Binns

- Joined Imagination as a grad in Jan 2010 working on an OpenVG driver
- Quickly moved to proprietary drivers for SGX and beyond supporting Linux
 - DRM/KMS drivers, dma-buf, dma-fences, EGL/WSI, Wayland, X11, ChromeOS, Tizen
- Tech lead on open source driver stack since work began in summer of 2020

XDC 2025 status recap



Passing Vulkan 1.0 CTS

- Conformant on [AXE-1-16M](#) and [BXS-4-64](#)

Vulkan 1.2 support merged into Mesa during XDC

- However, not quite passing all of Vulkan 1.2 CTS

Still needed to set PVR_I_WANT_A_BROKEN_VULKAN_DRIVER

- Dropped for BXS-4-64 in Mesa 25.3



Kernel and Firmware Update



Kernel and firmware update



Volcanic architecture support

Firmware booting on emulator and simulator



Significant work towards enabling firmware updates

Significantly easier to pick up improvements and bug fixes
Newer releases needed to support more recent GPUs



Significant work towards exposing and consuming performance counters

Groundwork carried out by Adrian Larumbe



Lots of kernel driver bug fixing



Added initial set of tests to IGT



Mesa Update



Vulkan 1.2 conformant



Imagination Technologies 2026-04-25 Vulkan_1_2		981
PowerVR B-Series BXS-4-64	CTS Version: 1.4.3.3 CPU: aarch64 OS: Linux sk-am68 6.16.0-rc4 (Ubuntu 24.04.2 LTS) DISPLAY: vulkan_headless	

Imagination Technologies 2026-08-27 Vulkan_1_2		990
PowerVR B-Series BXM-4-64	CTS Version: Vulkan_1_2 CPU: riscv64 OS: Linux beaglev-ahead 7.2.0-rc2-g656c754c016a (Ubuntu 24.04.4 LTS) Display: vulkan_headless Driver: Mesa PVR	

Vulkan features



depthBiasClamp
descriptorBindingInlineUniformBlockUpdateAfterBind
dynamicRendering
earlyFragmentMultisampleCoverageAfterSampleCounting
fragmentStoresAndAtomics
fullDrawIndexUint32
globalPriorityQuery
indexTypeUint8
inlineUniformBlock
maintenance4
maintenance5
multiDrawIndirect
nonSeamlessCubeMap
nonStrictWideLinesUseParallelogram
pipelineExecutableInfo
presentId
presentWait
primitiveTopologyListRestart
samplerMirrorClampToEdge
samplerYcbcrConversion
shaderImageGatherExtended
shaderInputAttachmentArrayDynamicIndexing
shaderIntegerDotProduct
shaderSampledImageArrayDynamicIndexing
shaderStorageBufferArrayDynamicIndexing
shaderStorageImageArrayDynamicIndexingdeviceMemoryReport
shaderStorageTexelBufferArrayDynamicIndexing
shaderSubgroupUniformControlFlow
shaderUniformBufferArrayDynamicIndexing
shaderUniformTexelBufferArrayDynamicIndexing

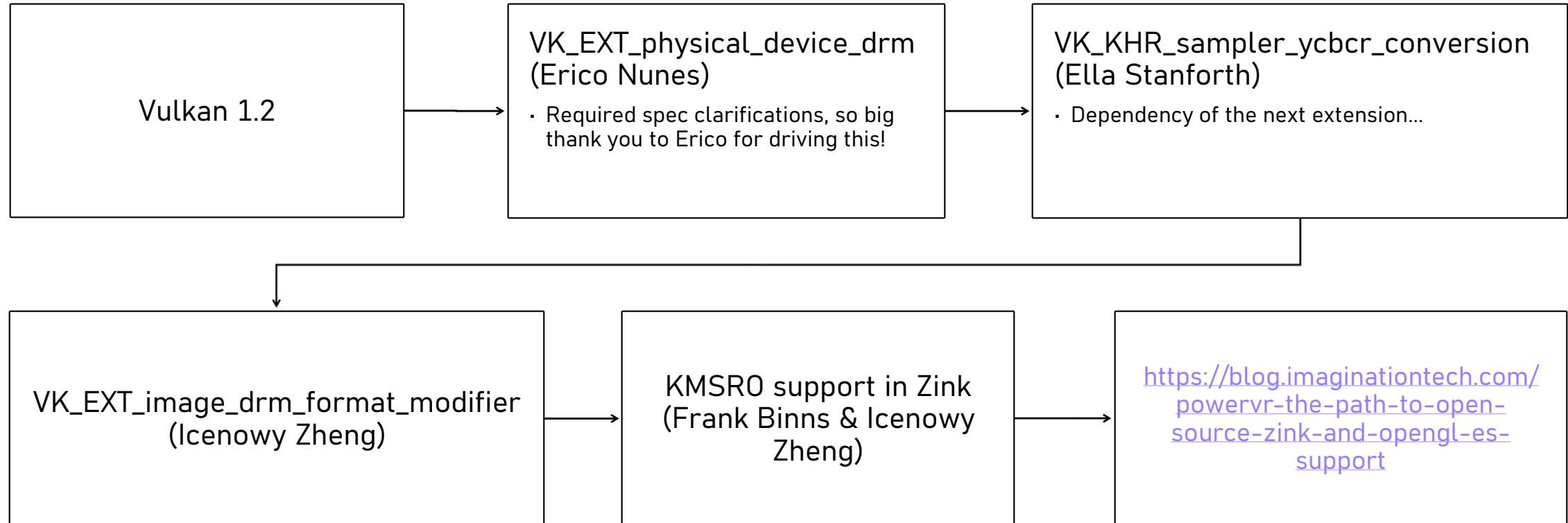
subgroupSizeControl
synchronization2
texelBufferAlignment
wideLines
workgroupMemoryExplicitLayout
workgroupMemoryExplicitLayoutScalarBlockLayout
VK_EXT_acquire_drm_display
VK_EXT_debug_marker
VK_EXT_device_memory_report
VK_EXT_direct_mode_display
VK_EXT_display_control
VK_EXT_display_surface_counter
VK_EXT_global_priority
VK_EXT_global_priority_query
VK_EXT_headless_surface
VK_EXT_image_drm_format_modifier
VK_EXT_index_type_uint8
VK_EXT_inline_uniform_block
VK_EXT_map_memory_placed
VK_EXT_non_seamless_cube_map
VK_EXT_primitive_topology_list_restart
VK_EXT_shader_replicated_composites
VK_EXT_shader_subgroup_ballot
VK_EXT_shader_subgroup_vote
VK_EXT_subgroup_size_control
VK_EXT_surface_maintenance1
VK_EXT_swapchain_colorspace
VK_EXT_texel_buffer_alignment
VK_EXT_zero_initialize_device_memory
VK_IMG_filter_linear_2d

VK_KHR_device_group
VK_KHR_device_group_creation
VK_KHR_dynamic_rendering
VK_KHR_format_feature_flags2
VK_KHR_global_priority
VK_KHR_incremental_present
VK_KHR_index_type_uint8
VK_KHR_maintenance4
VK_KHR_maintenance5
VK_KHR_maintenance7
VK_KHR_map_memory2
VK_KHR_pipeline_executable_properties
VK_KHR_present_id
VK_KHR_present_wait
VK_KHR_relaxed_block_layout
VK_KHR_sampler_mirror_clamp_to_edge
VK_KHR_sampler_ycbcr_conversion
VK_KHR_shader_expect_assume
VK_KHR_shader_fma
VK_KHR_shader_integer_dot_product
VK_KHR_shader_non_semantic_info
VK_KHR_shader_relaxed_extended_instruction
VK_KHR_shader_subgroup_rotate
VK_KHR_shader_subgroup_uniform_control_flow
VK_KHR_storage_buffer_storage_class
VK_KHR_surface_maintenance1
VK_KHR_swapchain_maintenance1
VK_KHR_synchronization2
VK_KHR_unified_image_layouts
VK_KHR_workgroup_memory_explicit_layout

Zink support



The path to Zink



Zink support



Status

GL ES 2.0 advertised since Mesa 26.1

GL ES 3.1 within touching distance...

- [!44385](#) adding support for VK_EXT_transform_feedback (Ashish Chauhan) is the last piece of the puzzle
- Looking to merge in time for 26.3

GL ES 3.1 CTS passing

- Final few fixes making their way upstream
- Conformance submission is imminent

Other highlights



More HW documentation (merged)

- Added control word and register descriptions
- Taken from original HW definitions
- <https://docs.mesa3d.org/drivers/powervr/csbaen.html>

GPU multi-arch support groundwork (merged)

- Worked on by Erik Faye-Lund and Ashish Chauhan
- PVR Vulkan driver has been re-architected to support multiple GPU architectures at run-time
- Paves the way for Volcanic architecture (and beyond)
- <https://docs.mesa3d.org/drivers/powervr.html#multi-architecture-support>
- <https://blog.imaginationtech.com/scaling-the-open-source-powervr-vulkan-driver-to-new-gpu-architectures>

vk-meta support (in-progress)

- Being worked on by Eric Smith

DRM shim support (merged)

PDSC (partially merged)



CI Update



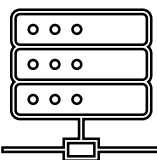
CI update

WE FINALLY HAVE IT!!!

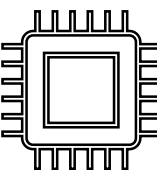
Farm hosted at the data center in the UK



Setup by Christian Gmeiner and
Imagination IT team

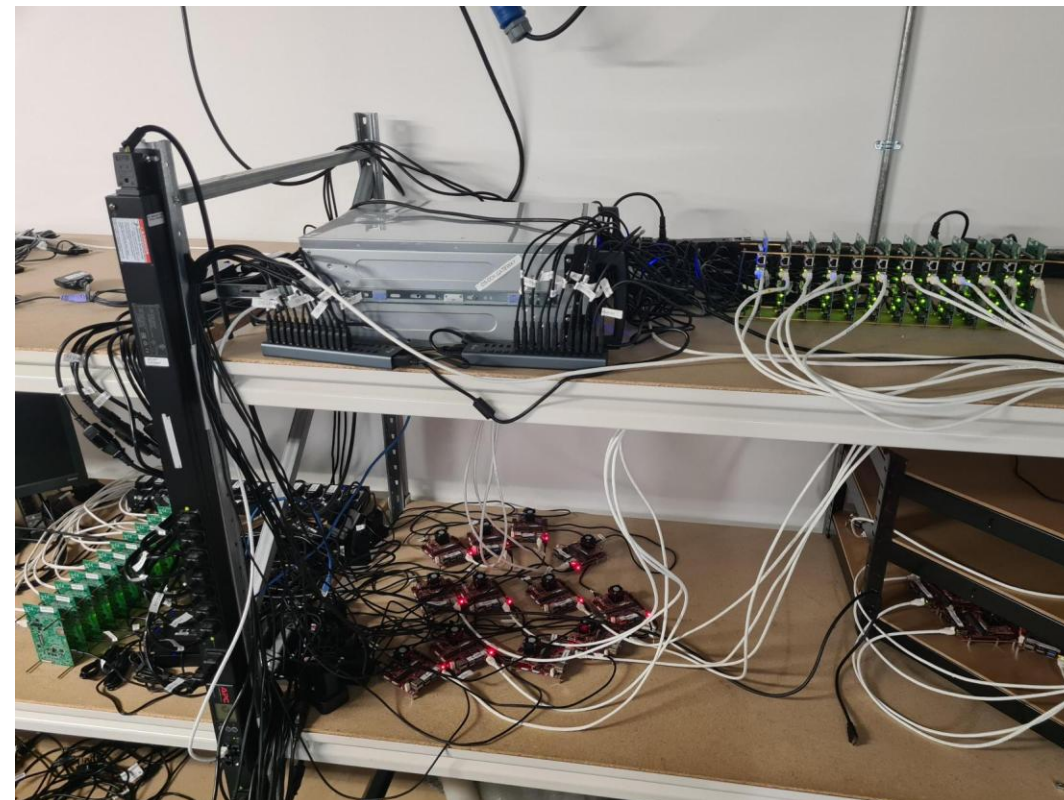


Runs CI-tron



Currently contains:

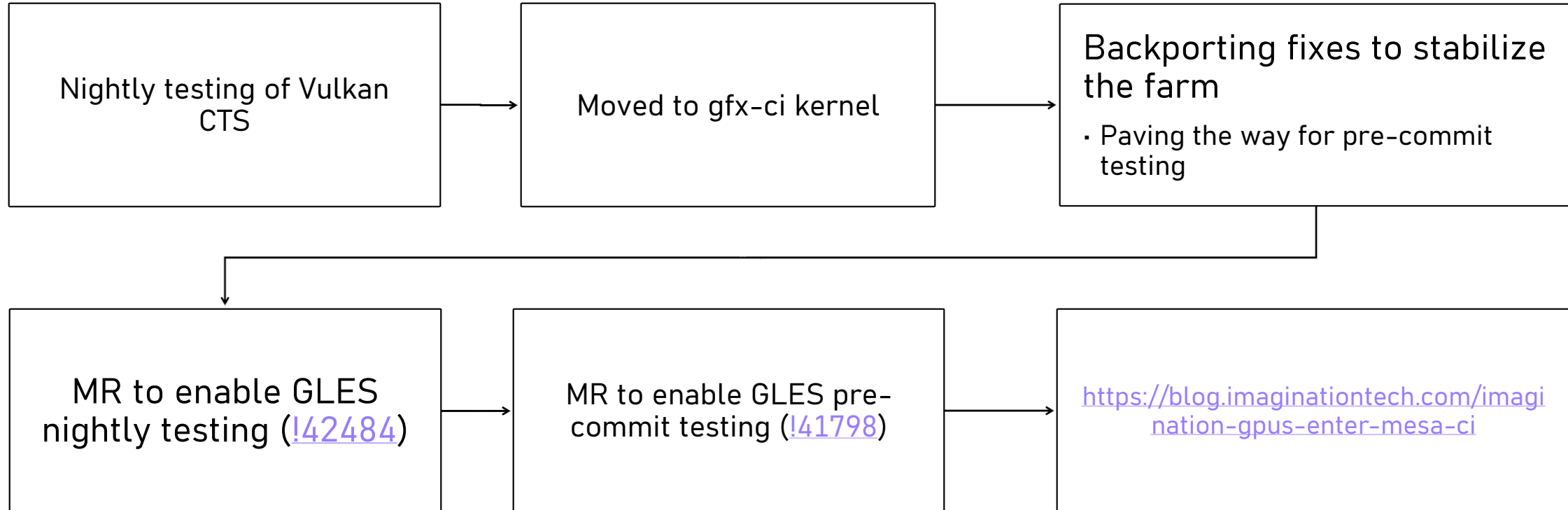
- 12x TI SK-AM62 boards (AXE-1-16M)
- 12x TI SK-AM68 boards (BXS-4-64)



CI update (cont.)



MESA CI





And in other news...



Other updates



No more IP reviews for Rogue based GPUs!



Fully featured version of PVRTune to be made publicly available

Replaces the "lite" version currently available on our website



GPU support for BeagleV-Ahead now works out of the box



Weston now works with the GLES renderer



BayLibre demonstrated basic Android 16 support

<https://baylibre.com/blog/baylibre-partners-with-spacemit-to-bring-android-16-to-risc-v/>



Experimental VisionFive2 GPU driver bring-up

Worked on by Dominique Belhachemi

https://github.com/domibel/visionfive2_gpu_bringup

Imagination

Looking ahead



What's next?



More Vulkan

- Will continue to work towards Vulkan 1.3 & 1.4 as time permits

Lots and lots of Volcanic architecture support

- Initially targeting AXM-8-256 & DXS-8-256

More CI

- Vulkan pre-commit testing
- More GPUs

Tooling

- PowerVR SDK: PVRTune & PVRCarbon
- Perfetto

Optimisations

Want to contribute?



<https://docs.mesa3d.org/drivers/powervr.html>

<https://docs.kernel.org/gpu/imagination/index.html>

BeaglePlay

CPU: arm64

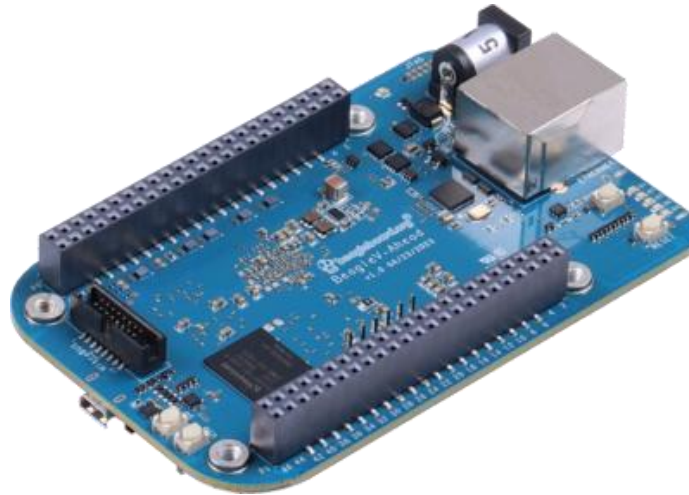
GPU: AXE-1-16M



BeagleV-Ahead

CPU: riscv64

GPU: BXM-4-64



BeagleY-AI

CPU: arm64

GPU: BXS-4-64



Source: beagleboard.org



Existing & upcoming SoCs

Allwinner A733

CPU: arm64

GPU: BXM-4-64 (Rogue architecture)

Boards: [Radxa Cubie A7S](#), [Radxa Cubie A7Z](#), [OrangePi Zero 3W](#), [OrangePi 4 Pro](#)

SpacemiT K3

CPU: riscv64

GPU: BXM-4-64 (Rogue architecture)

Boards: [Sipeed K3 Pico-ITX](#), [Milk-V Jupiter2](#), [Banana Pi BPI-SM10](#), [DeepComputing DC-ROMA RISC-V Mainboard III](#)

ESWIN EIC7700X

CPU: riscv64

GPU: AXM-8-256 (Volcanic architecture)

Boards: [SiFive HiFive Premier P550](#),

Renesas R-Car V4H

CPU: arm64

GPU: AXM-8-256 (Volcanic architecture)

Boards: [Retronix Sparrow Hawk](#), [Milk-V Megrez](#)

Texas Instruments TDA5

CPU: arm64

GPU: DXS-8-256 (Volcanic architecture)

Boards: TBC



PCO

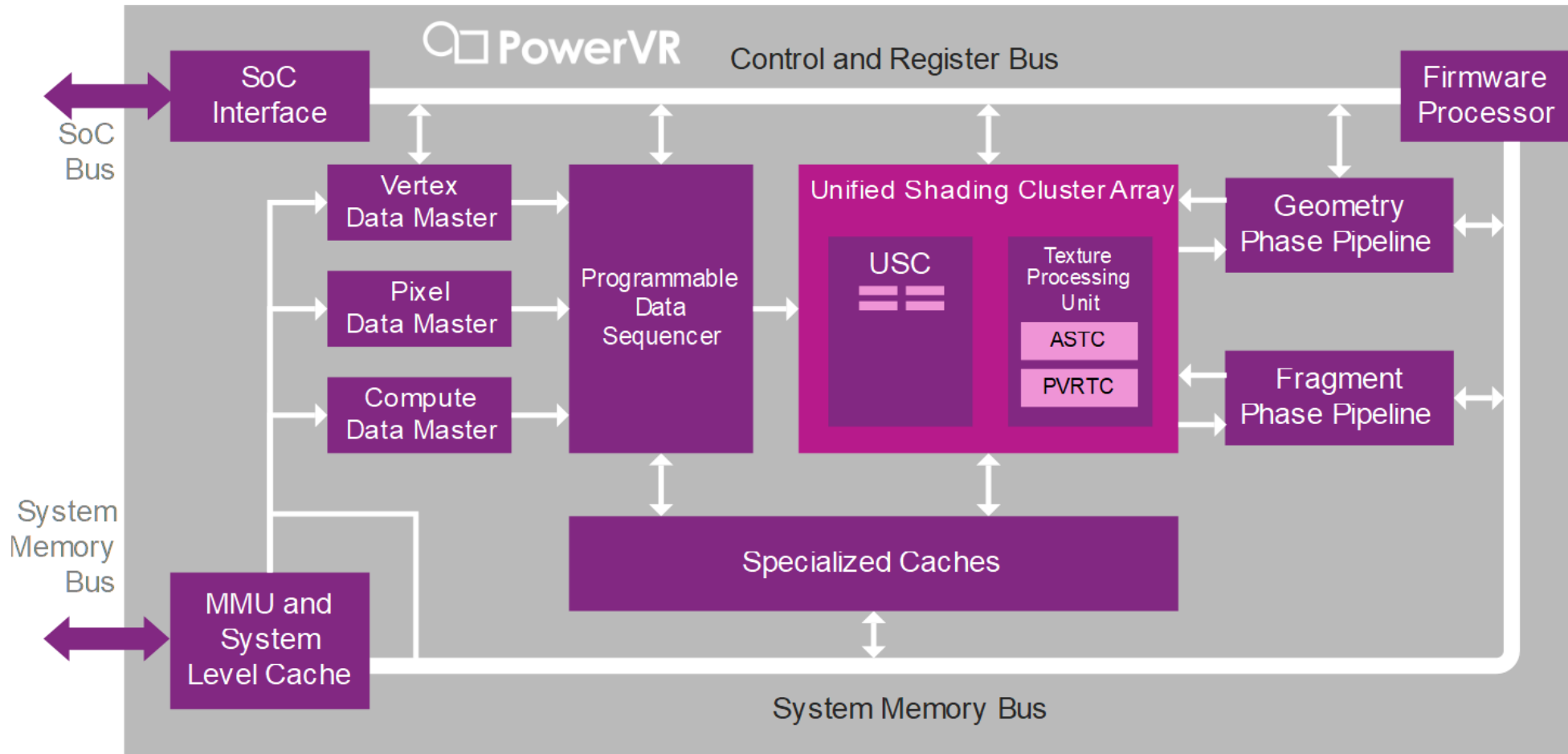




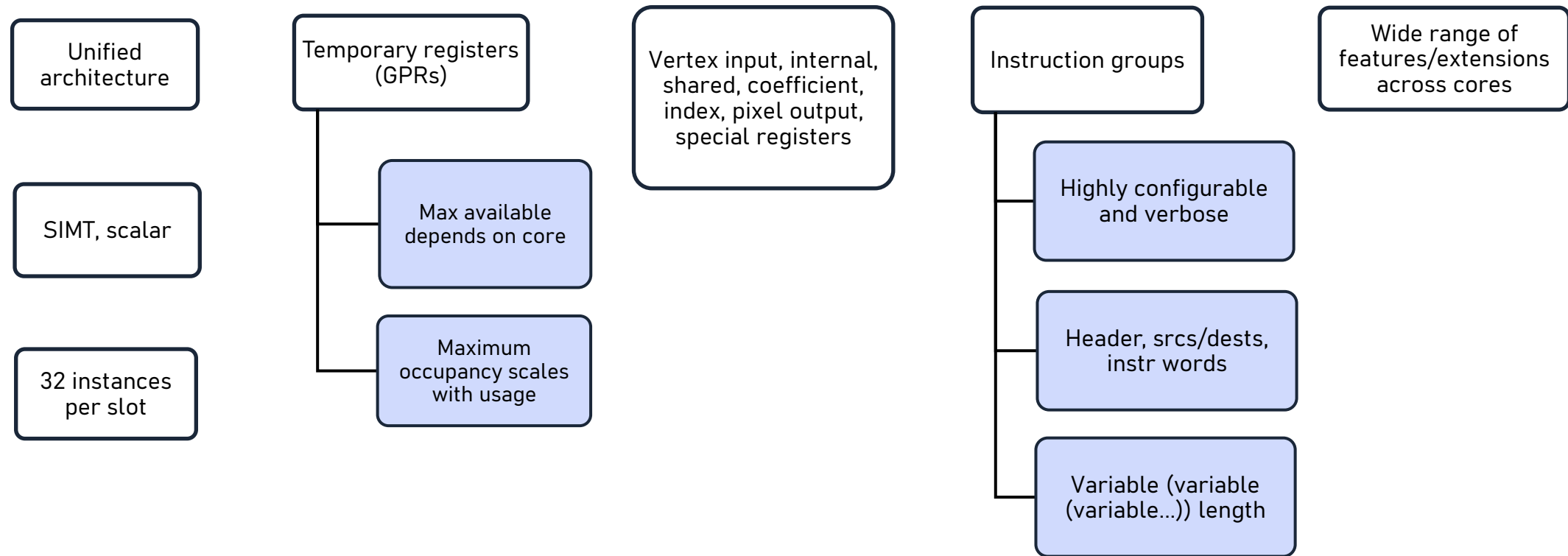
PVR Open-Source Compiler "History"

Experimental NIR to Uniflex translation layer (pre-2021)	"Experimental Rogue backend" (2021)	Rogue backend (late 2021-2024)	PCO (2025-present)
• Very early stages of project • Exercise in hooking up NIR to the proprietary DDK compiler • Not suitable beyond an experiment, but interesting nonetheless	• Extremely basic • Very simple SSA-based IR with blocks • Pre-baked ops • Internal shaders hardcoded • Exercise in learning NIR, tooling, and utilities within Mesa	• First "real" backend • Feature-rich (but heavy-handed...) IR • Some scheduling logic • I/O allocation within the compiler • Hard cap on architecture and core support • Performance not a consideration	• Full rewrite, fully upstreamed • New approach for IR • Targeting lowest common denominator for features • Pygen - encodings, mappings, validation rules • Rich disassembly and debug information • OpenCLC pre-compilation framework

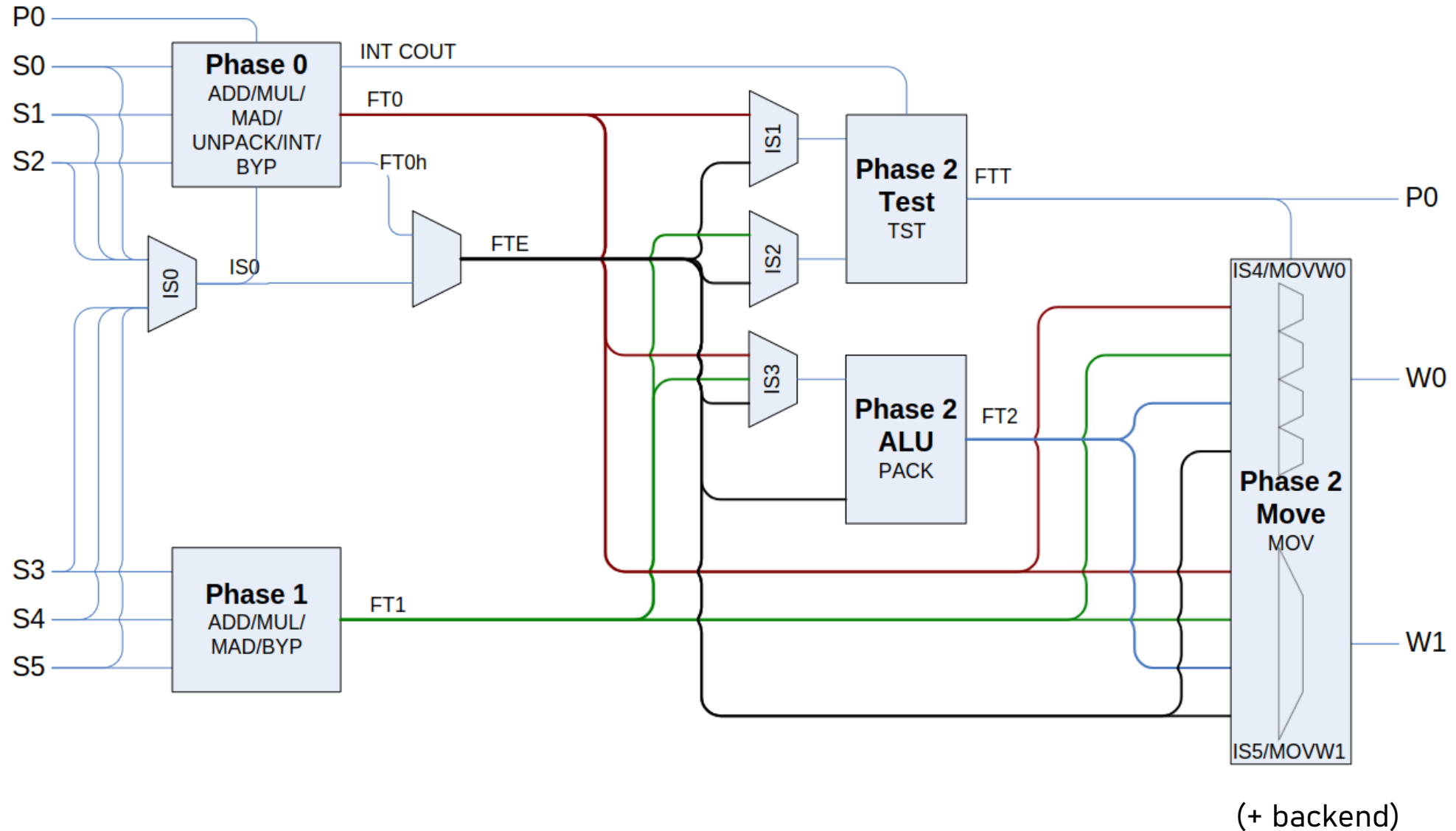
Background – Rogue Architecture



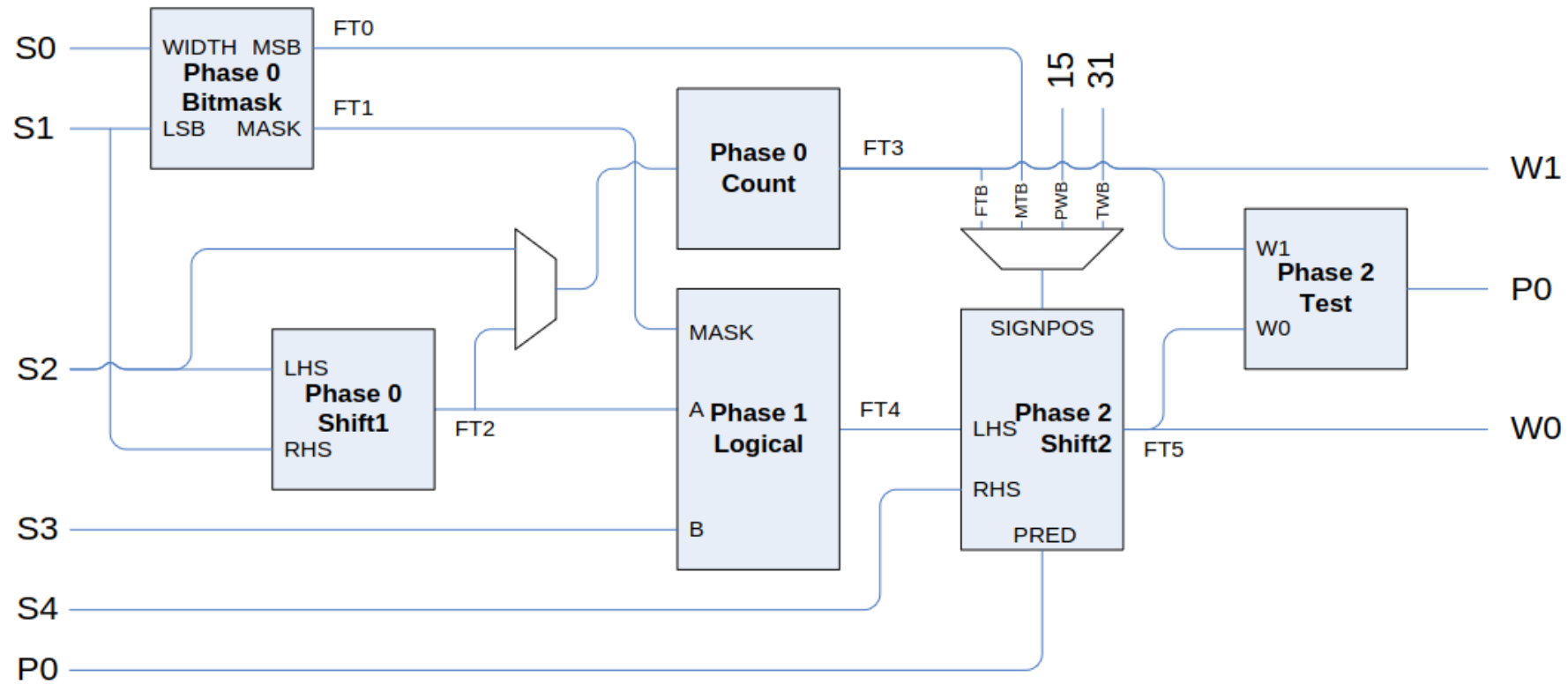
Background – Rogue Shader Architecture



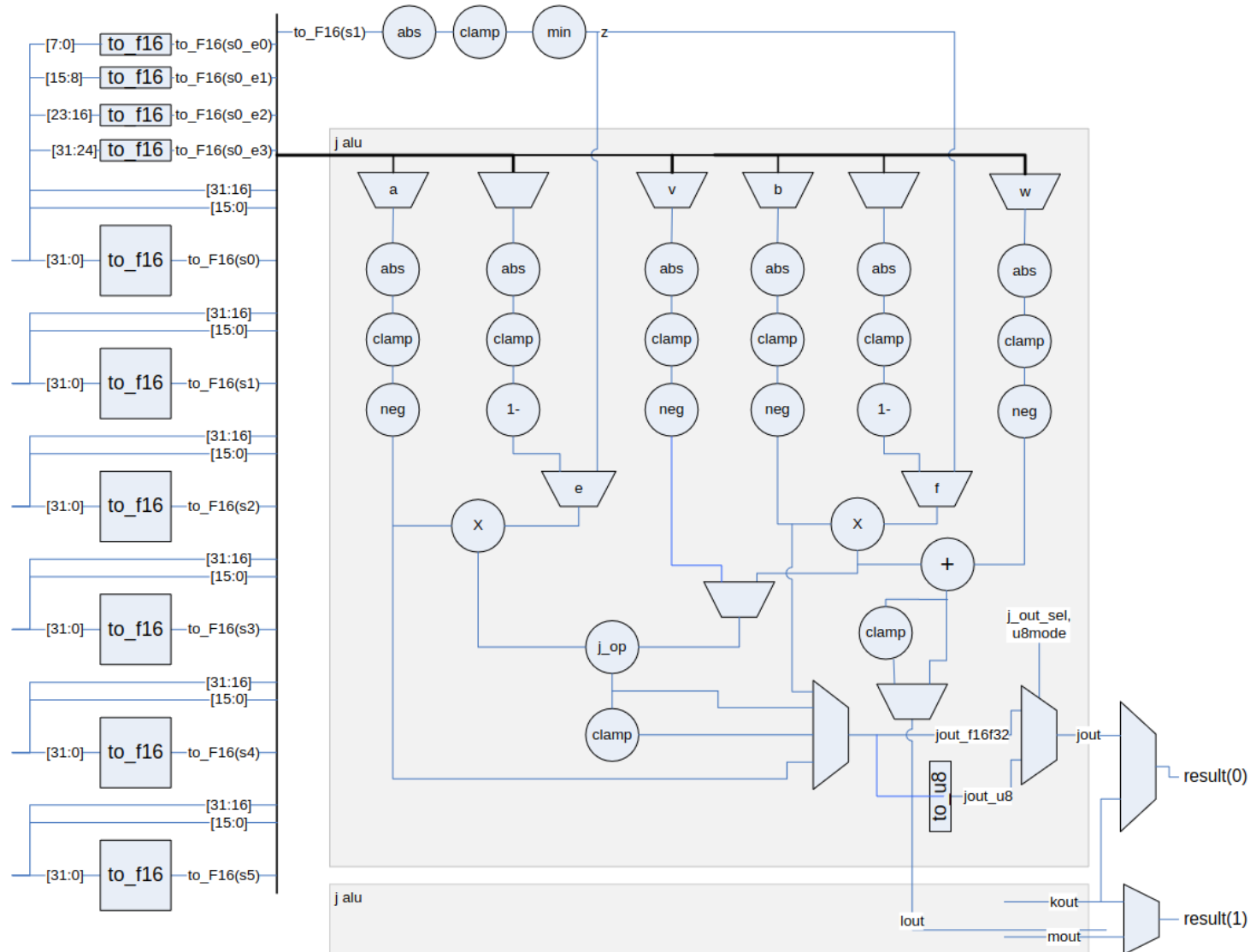
Background – Rogue Shader Architecture



Background – Rogue Shader Architecture



Background – Rogue Shader Architecture





PCO Design Considerations & Challenges

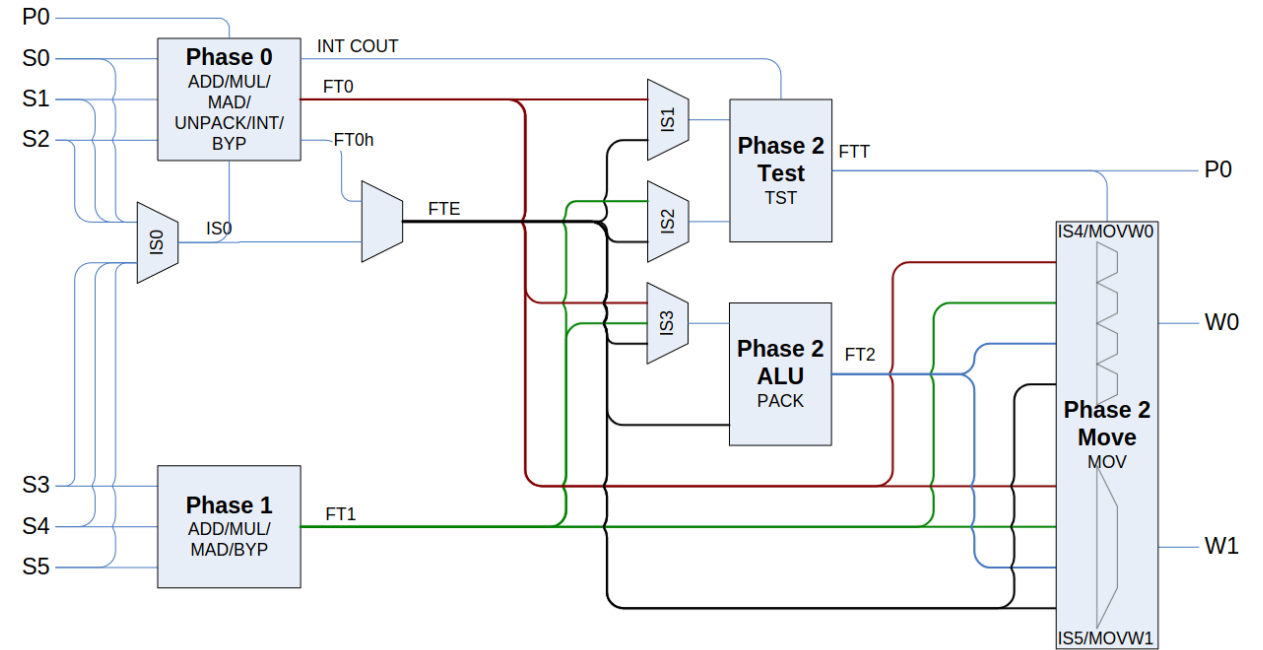
- Encodings and mappings
 - Representations, restrictions
 - Maintainability, documentation
- Instruction grouping
 - Static vs dynamic, performance
 - Future-proofing
- Core variants, new architectures
 - Extended instructions, restrictions
 - No instruction groups on Volcanic
- Internal shaders
 - Strict requirements
 - Pre-compilation, variants
- Control flow
 - EMC, jump opts
 - Predicated execution
- Subgroup support
 - (Ab)Using the LR for elect
 - Special branch conditions for vote

PC0 Design Considerations & Challenges



```
OM_ALU = [OM_OLCHK, OM_EXEC_CND, OM_END, OM_ATOM, OM_RPT]
O_FADD = hw_op('fadd', OM_ALU + [OM_SAT], 1, 2,
               [], [[RM_ABS, RM_NEG, RM_FLR], [RM_ABS]])
```

```
group_map(O_FADD,
  hdr=(I_IGRP_HDR_MAIN, [
    ('oporg', 'p0'),
    ('olchk', OM_OLCHK),
    ('wlp', False),
    ('w0p', True),
    ('cc', OM_EXEC_CND),
    ('end', OM_END),
    ('atom', OM_ATOM),
    ('rpt', OM_RPT)
  ]),
  enc_ops=[('0', O_FADD)],
  srcs=[
    ('s[0]', ('0', SRC(0)), 's0'),
    ('s[1]', ('0', SRC(1)), 's1')
  ],
  iss=[('is[4]', 'ft0')],
  dests=[('w[0]', ('0', DEST(0)), 'ft0')]
)
```



```
encode_map(O_FADD,
  encodings=[
    (I_FADD, [
      ('sat', OM_SAT),
      ('s0neg', (RM_NEG, SRC(0))),
      ('s0abs', (RM_ABS, SRC(0))),
      ('slabs', (RM_ABS, SRC(1))),
      ('s0flr', (RM_FLR, SRC(0)))
    ])
  ],
  op_ref_maps=[
    ('0', ['ft0'], ['s0', 's1']),
    ('1', ['ft1'], ['s3', 's4'])
  ]
)
```

PCO Design Considerations & Challenges



```
group_map(0_SCMP,
  hdr=(I_IGRP_HDR_MAIN, [
    ('qorg', 'p0 p1 p2'),
    ('olchk', OM_OLCHK),
    ('wlp', False),
    ('wop', True),
    ('cc', OM_EXEC_CND),
    ('end', OM_END),
    ('atom', OM_ATOM),
    ('rpt', OM_RPT)
  ]),
  enc_ops=[
    ('0', 0_MBYP, ['ft0'], [SRC(0)]),
    ('1', 0_MBYP, ['ft1'], [SRC(1)]),
    ('2_tst', 0_TST, ['ftt', '_'], ['is1', 'is2'],
      [(OM_TST_OP_MAIN, OM_TST_OP_MAIN), (OM_TST_TYPE_MAIN, 'f32')]),
    ('2_pck', 0_PCK, ['ft2'], ['_'], [(OM_PCK_FMT, 'one')]),
    ('2_mov', 0_MOVC, [DEST(0), '_'], ['ftt', 'ft2', 'is4', '_', '_'])
  ],
  srcs=[
    ('s[0]', ('0', SRC(0)), 's0'),
    ('s[1]', 'pco_zero'),
    ('s[3]', ('1', SRC(0)), 's3'),
  ],
  iss=[
    ('is[0]', 's1'),
    ('is[1]', 'ft0'),
    ('is[2]', 'ft1'),
    ('is[4]', 'fte'),
  ],
  dests=[
    ('w[0]', ('2_mov', DEST(0)), 'w0'),
  ]
)
```

```
binop("slt", tfloat, "", "(src0 < src1) ? 1.0f : 0.0f",
  valid_fp_math_ctrl = preserve_sz_inf_nan) # Set on Less Than
binop("sge", tfloat, "", "(src0 >= src1) ? 1.0f : 0.0f",
  valid_fp_math_ctrl = preserve_sz_inf_nan) # Set on Greater or Equal
binop("seq", tfloat, _2src_commutative, "(src0 == src1) ? 1.0f : 0.0f",
  valid_fp_math_ctrl = preserve_sz_inf_nan) # Set on Equal
binop("sne", tfloat, _2src_commutative, "(src0 != src1) ? 1.0f : 0.0f",
  valid_fp_math_ctrl = preserve_sz_inf_nan) # Set on Not Equal
```



30/09/2026

PCO Design Considerations & Challenges



```
shader ir before passes:
stage: MESA_SHADER_VERTEX
internal: false
func ENTRY main(void)
{
block B0:
  0000: movi32 %0, 0u; /* %0:(u32x1) */
  0001: rpt4 mov %1, vi0..3; /* %1:(f32x4) */
  0002: comp %5, %1, 0x0; /* %5:(32x1) */
  0003: comp %6, %1, 0x1; /* %6:(32x1) */
  0004: comp %7, %1, 0x2; /* %7:(32x1) */
  0005: comp %8, %1, 0x3; /* %8:(32x1) */
  0006: movi32 %2, 0u; /* %2:(u32x1) */
  0007: rpt4 uvsw.write %1, 0x0;
  0008: movi32 %3, 1.000000f; /* %3:(f32x1) */
  0009: movi32 %4, 0u; /* %4:(u32x1) */
  0010: uvsw.write %3, 0x4;
}
```

```
shader ir after pco_end:
stage: MESA_SHADER_VERTEX
internal: false
func ENTRY main(void)
{
block B0:
  0000: rpt4 uvsw.write vi0..3, 0x0;
  0002: uvsw.write.emit.endtask.end sc64, 0x4;
}
```

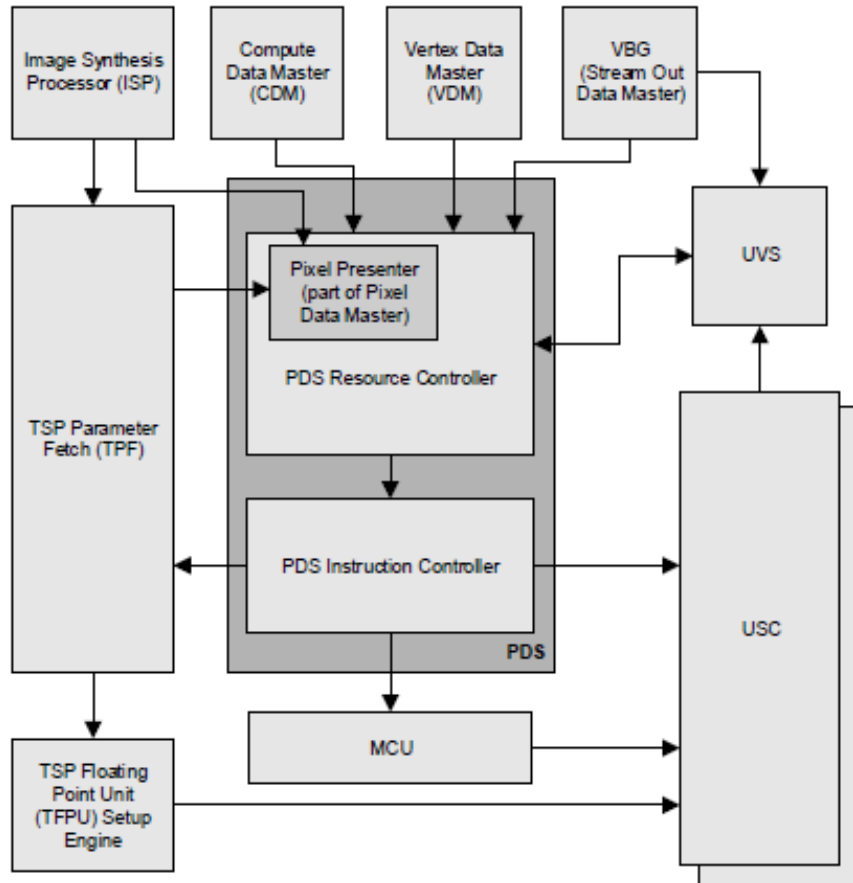
```
shader ir after passes:
stage: MESA_SHADER_VERTEX
internal: false
func ENTRY main(void) /* temps: 0 */ /* vtxins: 0 */
{
block B0:
  0000: { /* @ 0x00000000 [backend] len: 10, pad: 6, total: 16, da: 5 */
    type main /* hdr bytes: 3 @ +0 */
    repeat 4
    s0 = vi0..3 /* lo src bytes: 3 @ +5 */
    /* up src bytes: 1 @ +8 */
    is0 = s0, is4 = fte /* iss bytes: 1 @ +9 */
    backend: uvsw.write w0, 0x0; /* backend bytes: 2 @ +3 */
  }
  0001: { /* @ 0x00000010 [backend] len: 10, pad: 6, total: 16, da: 5 */
    type main /* hdr bytes: 3 @ +0 */
    s0 = sc64 /* lo src bytes: 3 @ +5 */
    /* up src bytes: 1 @ +8 */
    is0 = s0, is4 = fte /* iss bytes: 1 @ +9 */
    backend: uvsw.write.emit.endtask w0, 0x4; /* backend bytes: 2 @ +3 */
  }.end
}
```



PDSC

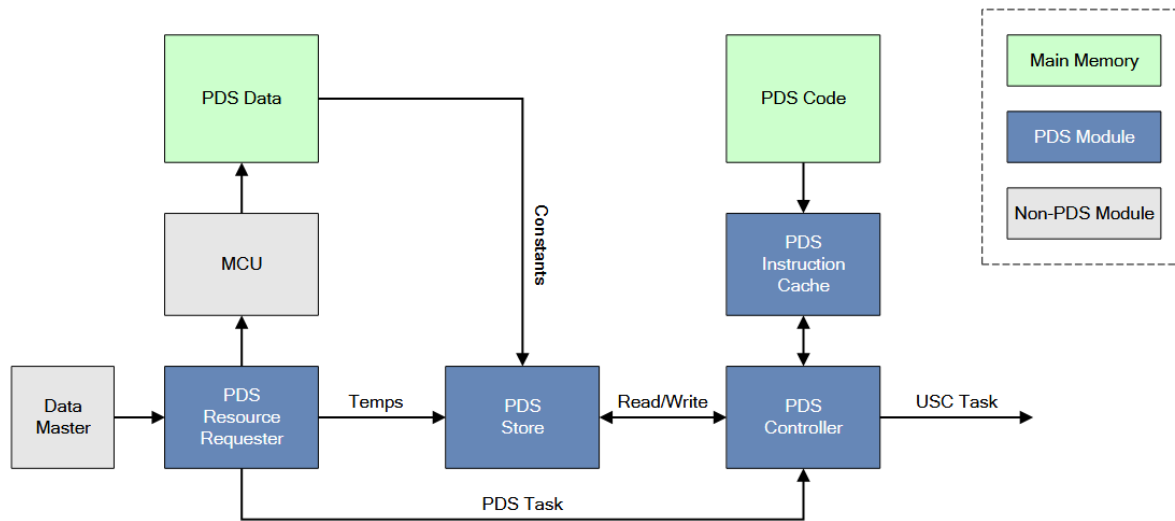


What is the PDS?



- Programmable Data Sequencer.
- Resource controller:
 - Takes task requests from DMs.
 - Packs data instances into tasks.
 - Allocates resources for tasks.
- Instruction controller:
 - Runs PDS programs.
 - Can perform DMA, ALU, sync ops.
 - Can create and send off USC tasks for execution.

How are PDS programs structured?

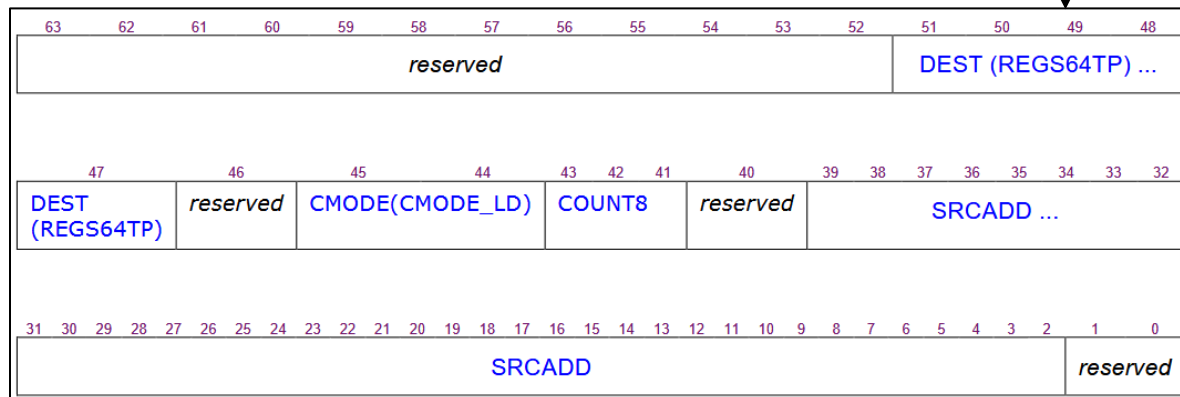
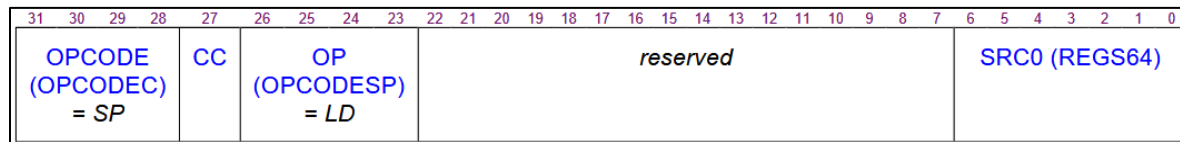
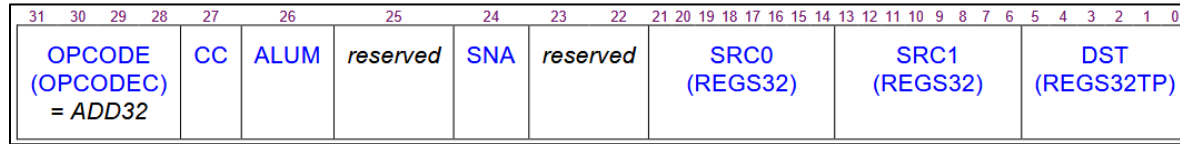


- Separate code/data segments.
- Register banks:
 - Temporary registers.
 - Some pre-loaded depending on the program.
 - Persistent temporary registers.
 - Constant registers.

Difficulties when developing PDS programs



Op parameters and disassembly



- ALU and control flow ops have simple parameters.
- Most other ops do not, parameters are “abstracted”.
 - Useful for PDS program generality.
 - Makes debugging a lot harder.



Example: Multiview view index init program

Existing program gen

```
uint32_t *pvr_pds_generate_view_index_init_program(
    struct pvr_pds_view_index_init_program *restrict program,
    uint32_t *restrict buffer,
    enum pvr_pds_generate_mode gen_mode)
{
    uint32_t next_temp = PVR_PDS_TEMPS_BLOCK_BASE;
    uint32_t temps_used = 0;
    uint32_t data_size = 0;
    uint32_t code_size = 0;

    if (gen_mode == PDS_GENERATE_CODE_SEGMENT ||
        gen_mode == PDS_GENERATE_SIZES) {
        const bool encode = (gen_mode == PDS_GENERATE_CODE_SEGMENT);
#define APPEND(X) \
        if (encode) { \
            *buffer = X; \
            buffer++; \
        } else { \
            code_size += sizeof(uint32_t); \
        }

        uint32_t view_index_temp = pvr_pds_get_temps(&next_temp, 1,
            &temps_used);
        uint32_t zero_temp = pvr_pds_get_temps(&next_temp, 1,
            &temps_used);

        /* Load the view index into a view_index_temp. */
        APPEND(
            pvr_pds_inst_encode_limm(0, view_index_temp, program-
            >view_index, 0));

        /* Load the zero into a zero_temp. */
        APPEND(pvr_pds_inst_encode_limm(0, zero_temp, 0, 0));

        /* Copy the temp into ptemp. */
        APPEND(pvr_pds_inst_encode_add32(
            0,
            0,
            0,
            view_index_temp,
            zero_temp,
            PVR_ROGUE_PDSINST_REGS32TP_PTEMP32_LOWER +
            PVR_PTEMP_VIEW_INDEX));

        APPEND(pvr_pds_inst_encode_halt(0));
#undef APPEND

        program->temps_used = temps_used;
    }

    if (gen_mode == PDS_GENERATE_SIZES) {
        program->code_size = code_size;
        program->data_size = data_size;
    }

    return buffer;
}
```

- Sets a ptemp to a value (program->view_index) and halts.
- Boilerplate-heavy.
- Poor generality, const segment unused.
- Explicitly using add32(view index, 0) due to op limitations.
- Generated at runtime.



Example: Multiview view index init program

PDSC program gen

```
static pdsc_program *pdsc_set_view_index_gen(void *mem_ctx,  
                                              enum pdsc_feature features)  
{  
    pdsc_program *p = pdsc_program_create(mem_ctx, features, "set_view_index");  
  
    ALLOC_CONST32(p, view_index_in, PVR_SET_VIEW_INDEX_ID_IN);  
    DECL_PTEMP32(p, view_index_out, PVR_VDM_PTEMP_VIEW_INDEX_OUT);  
  
    pdsc_mov32(p, view_index_out, view_index_in);  
    pdsc_halt(p);  
  
    return p;  
}
```



Example: Multiview view index init program

PDSC program specialization/patching

```
pdsc_program **p = &device->pdsc_precomp_programs.set_view_index;
*p = pdsc_precomp_set_view_index_variant(NULL, device->pdevice->pdsc_features);
pdsc_finalize_program(*p);

/* Upload code section (unchanging). */
pvr_gpu_upload_pds(..., pdsc_program_code(*p), pdsc_program_code_size(*p) / sizeof(uint32_t), ...);

/* Patch and upload data section for each view index. */
for (unsigned u = 0; u < PVR_MAX_MULTIVIEW; ++u) {
    pdsc_patch patch = { .patch_id = PVR_SET_VIEW_INDEX_ID_IN, .value = u };
    pdsc_patch_program(*p, 1, &patch, NULL);
    pvr_gpu_upload_pds(..., pdsc_program_data(*p), pdsc_program_data_size(*p) / sizeof(uint32_t), ...);
}
```

Example: Multiview view index init program



Debug output

```
ir before pdsc_encode_program:
"set_view_index" pds program:
features: (none)
const regs used: 2, required: 2
imm32_t c[0] view_index_in = (unassigned); (patch_id 0)
imm32_t c[1] const32_0 = 0x00000000;

ptemp regs referenced: 1
imm32_t pt[4] view_index_out;

2 instrs
000: add32.u view_index_out (pt[4]), view_index_in (c[0]), const32_0 (c[1]);
001: halt;
```

```
ir after pdsc_patch_program:
"set_view_index" pds program:
features: (none)
const regs used: 2, required: 2
imm32_t c[0] view_index_in = 0x00000005; (patch_id 0)
imm32_t c[1] const32_0 = 0x00000000;
```

State of PDSC



All PDS programs now ported to PDSC

- Old PDS compiler dropped
- Majority of PDS programs now generated at compile-time
- Program variants programmatically generated
- Many thanks to Ella Stanforth!
- Changes can be found in [!42260](#)

Compare **main** ▾ and **latest version** ▾



102 files **+12960** **-19578**





Demo time





Q & A



THANK YOU.

